aws

Guida per gli sviluppatori

Database Amazon Quantum Ledger (Amazon QLDB)

Database Amazon Quantum Ledger (Amazon QLDB): Guida per gli sviluppatori

Table of Contents

Cos'è Amazon QLDB?

Database Amazon Quantum Ledger (Amazon QLDB) è un database di libro mastro completamente gestito che fornisce un log delle transazioni appartenenti a un'autorità centrale attendibile in modo trasparente, immutabile, crittografato e verificabile. Amazon QLDB può essere utilizzato per tenere traccia di tutte le modifiche nel tempo. Per ulteriori informazioni sulla varietà di opzioni di database disponibili su Amazon Web Services, consulta Scegliere il database giusto per la tua organizzazione suAWS.

I registri vengono in genere utilizzati per registrare una cronologia delle attività economiche e finanziarie di un'organizzazione. Molte organizzazioni creano applicazioni con funzionalità simili a quelle dei registri perché desiderano mantenere una cronologia accurata dei dati delle proprie applicazioni. Ad esempio, potrebbero voler tenere traccia della cronologia dei crediti e degli addebiti nelle transazioni bancarie, verificare la provenienza dei dati di un sinistro assicurativo o tracciare il movimento di un articolo in una rete di catena di approvvigionamento. Le applicazioni di contabilità vengono spesso implementate utilizzando tabelle di controllo personalizzate o percorsi di controllo creati in database relazionali.

Amazon QLDB è una nuova classe di database che aiuta a eliminare la necessità di impegnarsi nel complesso sforzo di sviluppo di applicazioni simili a registri. Con QLDB, la cronologia delle modifiche ai dati è immutabile: non può essere sovrascritta o alterata in sede. Inoltre, utilizzando la crittografia, puoi verificare che non siano state apportate modifiche involontarie ai dati dell'applicazione. QLDB utilizza un registro transazionale immutabile, noto come journal. Il journal è solo per le appendici ed è composto da un set di blocchi sequenziati e concatenati di hash che contengono i dati salvati.

Video Amazon QLDB

Per una panoramica di Amazon QLDB e di come può esserti utile, guarda questo video introduttivo di QLDB su YouTube.

Prezzi di Amazon QLDB

I prezzi di Amazon QLDB vengono calcolati in base all'uso effettivo, senza tariffe minime o utilizzo obbligatorio del servizio. I costi sono calcolati solo in base alle risorse utilizzate dal database contabile e non è necessario provvedere in anticipo.

Per ulteriori informazioni, consulta Prezzi di Amazon QLDB.

Nozioni di base su QLDB

Consigliamo di iniziare leggendo gli argomenti seguenti:

- Panoramica di Amazon QLDB— Per avere una panoramica di alto livello di QLDB.
- Concetti e terminologia fondamentali in Amazon QLDB— Apprendere i concetti e la terminologia fondamentali del QLDB.
- Accesso ad Amazon QLDB— Per imparare ad accedere a QLDB utilizzandoAWS Management Console, API oAWS Command Line Interface (AWS CLI).
- Come funziona Amazon QLDB con IAM— Per imparare a controllare l'accesso a QLDB usandoAWS Identity and Access Management (IAM).

Per iniziare a utilizzare rapidamente la console QLDB, consultaNozioni di base sulla console Amazon QLDB.

Per informazioni sullo sviluppo con QLDB utilizzando un driverAWS fornito, vedereNozioni base sul driver Amazon QLDB.

Panoramica di Amazon QLDB

Nelle seguenti sezioni viene riportata una panoramica di alto livello dei componenti del servizio Amazon QLDB e di come interagiscono tra loro.

Argomenti

- Prima il diario
- Non modificabile
- Verificabile crittograficamente
- Simile a SQL e flessibile per i documenti
- Strumenti di sviluppo open source
- Senza server e a disponibilità elevata
- Grado aziendale

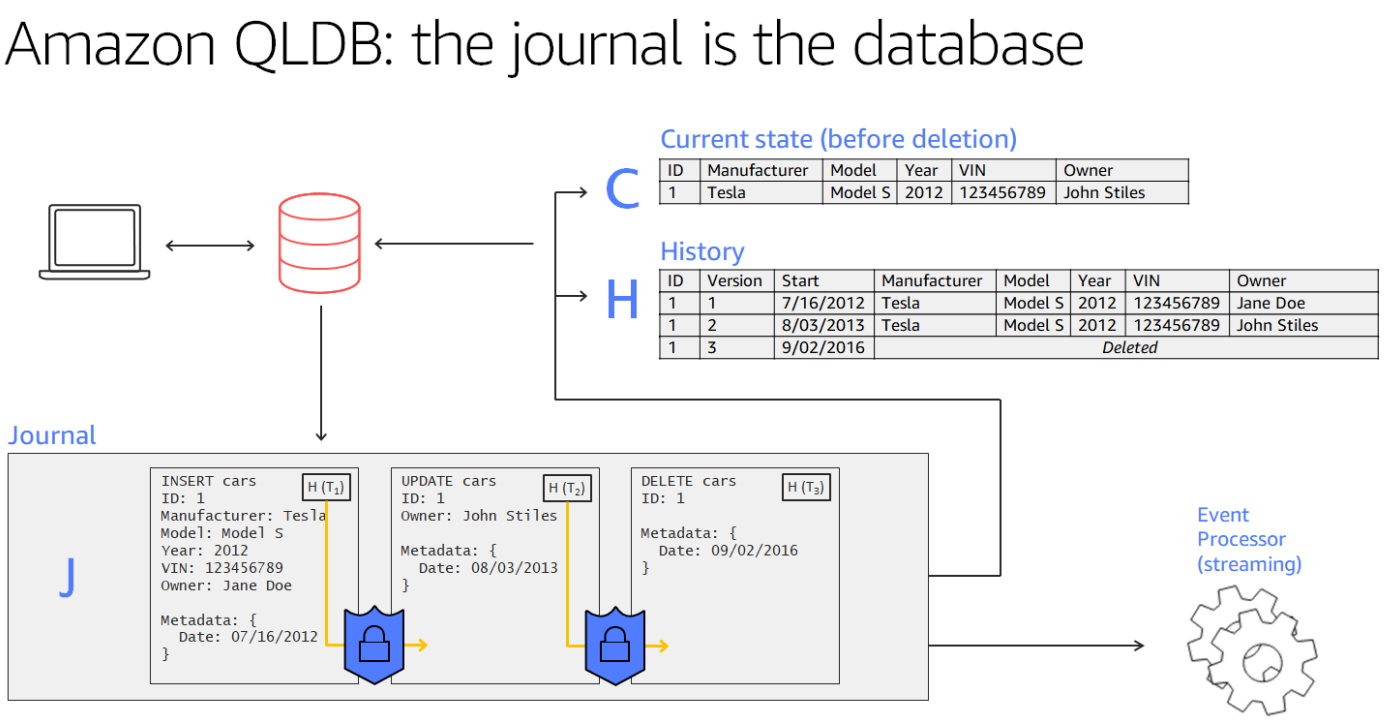
Prima il diario

Nell'architettura di database tradizionale, in genere si scrivono dati in tabelle come parte di una transazione. Un registro delle transazioni, in genere un'implementazione interna, registra tutte le transazioni e le modifiche apportate al database. Il registro delle transazioni è un componente fondamentale del database. È necessario il registro per riprodurre le transazioni in caso di guasto del sistema, ripristino di emergenza o replica dei dati. Tuttavia, i registri delle transazioni del database non sono immutabili e non sono progettati per fornire un accesso diretto e semplice agli utenti.

In Amazon QLDB, il journal è il cuore del database. Strutturalmente simile a un registro delle transazioni, il journal è una struttura dati immutabile e di sola aggiunta che memorizza i dati dell'applicazione insieme ai metadati associati. Tutte le transazioni di scrittura, inclusi gli aggiornamenti e le eliminazioni, vengono prima salvate nel giornale.

QLDB utilizza il journal per determinare lo stato corrente dei dati contabili materializzandoli in tabelle interrogabili e definite dall'utente. Queste tabelle forniscono anche una cronologia accessibile di tutti i dati delle transazioni, comprese le revisioni dei documenti e i metadati. Inoltre, il giornale gestisce la concorrenza, il sequenziamento, la verifica crittografica e la disponibilità dei dati contabili.

Il diagramma seguente illustra l'architettura del diario QLDB.



- In questo esempio, un'applicazione si connette a un libro mastro ed esegue transazioni che inseriscono, aggiornano ed eliminano un documento in una tabella denominata`cars`.
- I dati vengono prima scritti nel giornale in ordine sequenziale.
- Quindi i dati vengono materializzati nella tabella con viste integrate. Queste visualizzazioni consentono di interrogare sia lo stato corrente che la cronologia completa dell'auto, e a ciascuna revisione viene assegnato un numero di versione.
- Puoi anche esportare o trasmettere dati direttamente dal giornale.

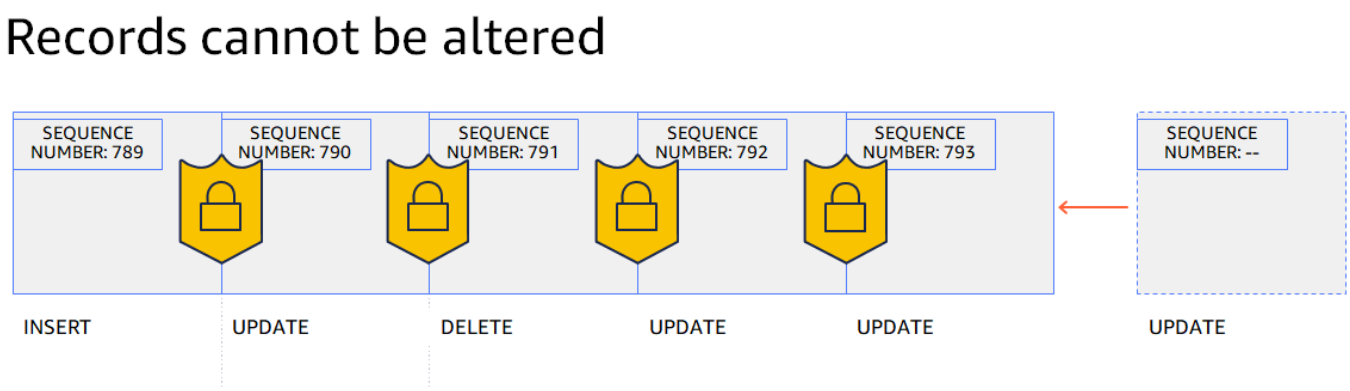
Non modificabile

Poiché il journal QLDB è solo per le aggiunte, mantiene un registro completo di tutte le modifiche ai dati che non possono essere modificate o sovrascritte. Non esistono API o altri metodi per modificare i dati salvati. Questa struttura contabile consente di accedere e interrogare la cronologia completa del libro contabile.

> **Note**
>
> L'unica eccezione all'immutabilità supportata da QLDB è la redazione dei dati. Con questa funzione, puoi rispettare gli statuti normativi come il Regolamento generale sulla protezione dei dati (GDPR) nell'Unione europea e il California Consumer Privacy Act (CCPA).
> QLDB fornisce un'operazione di redazione che consente di eliminare definitivamente le revisioni dei documenti inattivi nella cronologia di una tabella. Questa operazione elimina solo i dati utente nella revisione specificata e lascia invariati la sequenza del diario e i metadati del documento. Ciò mantiene l'integrità complessiva dei dati del tuo libro contabile. Per ulteriori informazioni, consulta Redazione delle revisioni dei documenti.

QLDB scrive un blocco nel journal in una transazione. Ogni blocco contiene oggetti di ingresso che rappresentano i documenti che inserisci, aggiorni ed elimini, insieme alle istruzioni che hai eseguito per confermarli. Questi blocchi sono sequenziati e concatenati di hash per garantire l'integrità dei dati.

Il diagramma seguente illustra questa struttura del diario.

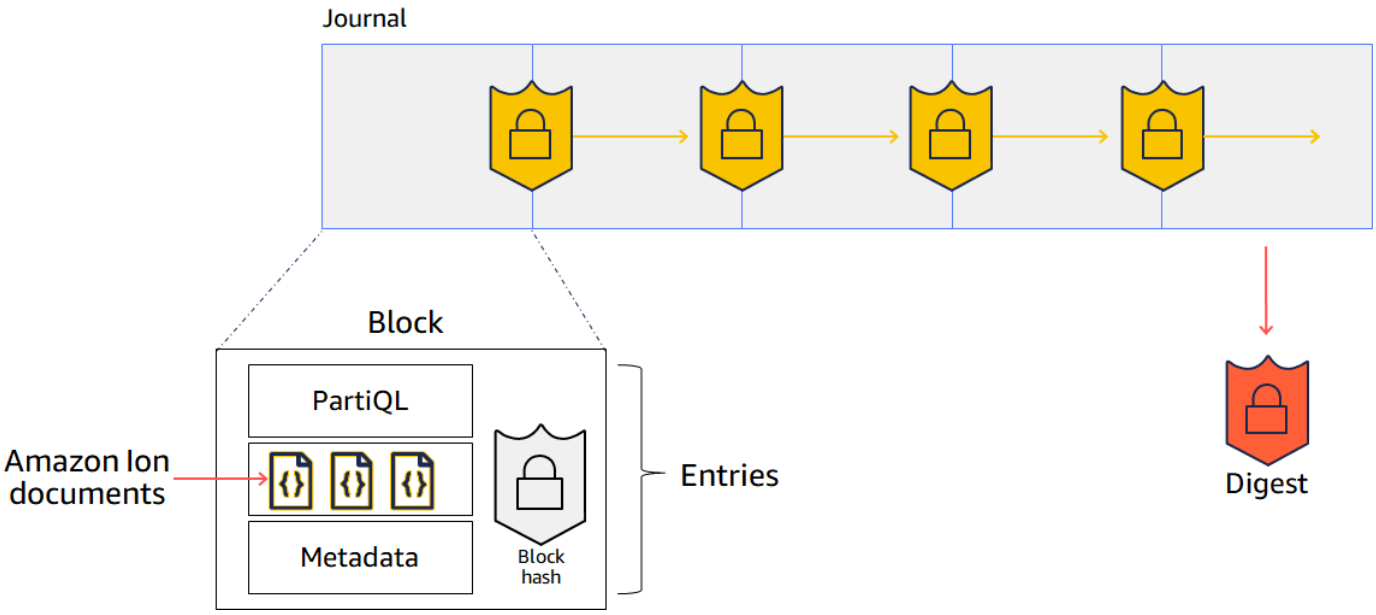


Il diagramma mostra che le transazioni vengono salvate nel giornale come blocchi concatenati di hash per la verifica. Ogni blocco ha un numero di sequenza per specificare il suo indirizzo.

Verificabile crittograficamente

I blocchi del journal sono sequenziati e concatenati con tecniche di hashing crittografico, simili alle blockchain. QLDB utilizza la catena di hash della rivista per fornire l'integrità dei dati transazionali utilizzando un metodo di verifica crittografica. Utilizzando un digest (un valore hash che rappresenta l'intera catena di hash di un giornale in un determinato momento) e una prova di controllo Merkle (un meccanismo che dimostra la validità di qualsiasi nodo all'interno di un albero hash binario), puoi verificare che non ci siano state modifiche involontarie ai tuoi dati in qualsiasi momento.

Il diagramma seguente mostra un riassunto che copre l'intera hash chain di un diario in un determinato momento.



In questo diagramma, i blocchi del journal vengono sottoposti a hashing utilizzando la funzione hash crittografica SHA-256 e vengono concatenati in sequenza ai blocchi successivi. Ogni blocco contiene voci che includono i documenti di dati, i metadati e le istruzioni PartiQL eseguite nella transazione.

Per ulteriori informazioni, consulta Verifica dei dati in Amazon QLDB.

Simile a SQL e flessibile per i documenti

QLDB utilizza PartiQL come linguaggio di interrogazione e Amazon Ion come modello di dati orientato ai documenti. PartiQL è un linguaggio di interrogazione open source e compatibile con SQL che è stato esteso per funzionare con Ion. Con PartiQL, puoi inserire, interrogare e gestire i tuoi dati con operatori SQL familiari. Quando si eseguono interrogazioni su documenti flat, la sintassi è la stessa dell'utilizzo di SQL per interrogare le tabelle relazionali. Per ulteriori informazioni sull'implementazione di PartiQLDB, consulta laDocumentazione di riferimento Amazon QLDB PartiQL.

Amazon Ion è un superset di JSON. Ion è un formato di dati open source basato su documenti che offre la flessibilità di archiviare ed elaborare dati strutturati, semistrutturati e annidati. Per ulteriori informazioni su Ion in QLDB, consulta laRiferimento al formato dei dati Amazon Ion in Amazon QLDB.

Per un confronto di alto livello dei componenti e delle funzionalità principali dei database relazionali tradizionali rispetto a QLDB, vedereDa relazionale a libro mastro.

Strumenti di sviluppo open source

Per semplificare lo sviluppo di applicazioni, QLDB fornisce driver open source in vari linguaggi di programmazione. È possibile utilizzare questi driver per interagire con l'API dei dati transazionali eseguendo istruzioni PartiQL su un libro mastro ed elaborando i risultati di tali dichiarazioni. Per informazioni e tutorial sulle lingue dei driver attualmente supportate, vedereNozioni base sul driver Amazon QLDB.

Amazon Ion fornisce anche librerie client che elaborano i dati Ion per te. Per guide per sviluppatori ed esempi di codice sull'elaborazione dei dati Ion, consulta la documentazione di Amazon Ion su GitHub.

Senza server e a disponibilità elevata

QLDB è completamente gestito, serverless e altamente disponibile. Il servizio è scalabile automaticamente per supportare le esigenze dell'applicazione e non è necessario fornire istanze o capacità. Più copie dei dati vengono replicate all'interno di una zona di disponibilità e tra le zone di disponibilità in unaRegione AWS.

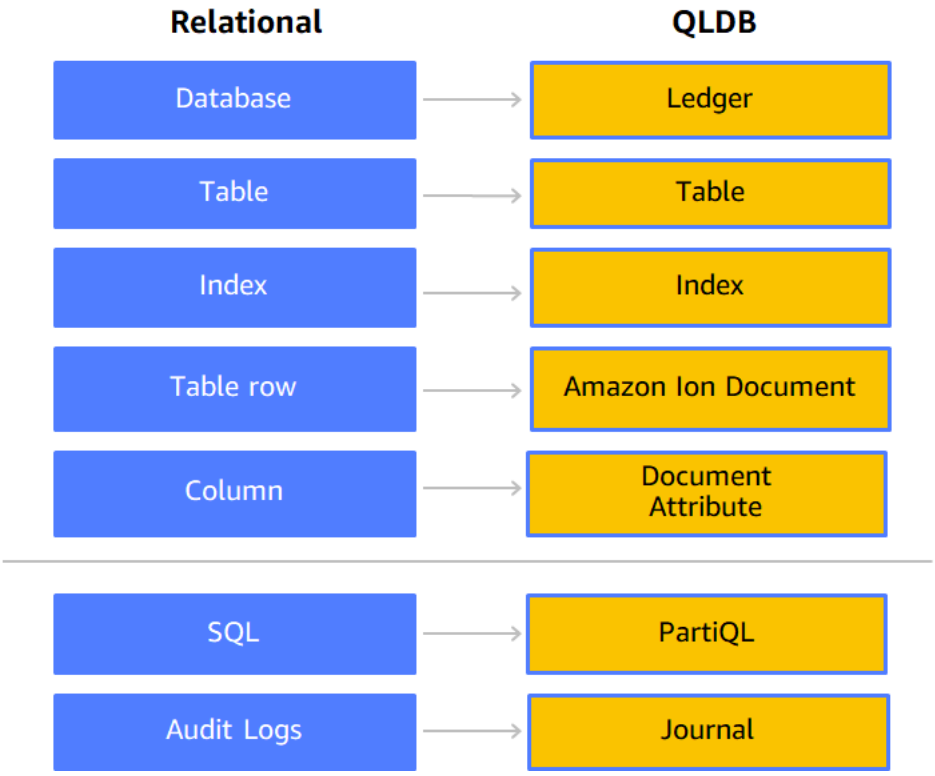
Grado aziendale

Le transazioni QLDB sono completamente conformi alle proprietà di atomicità, consistenza (ACID, Isolation and Durability). QLDB utilizza il controllo ottimistico della concorrenza (OCC) e le transazioni operano con serializzabilità completa, il massimo livello di isolamento. Ciò significa che non c'è il rischio di vedere letture fantasma, letture sporche, scritture distorte o altri problemi simili di concorrenza. Per ulteriori informazioni, consulta Modello di concorrenza Amazon QLDB.

Da relazionale a libro mastro

Se sei uno sviluppatore di applicazioni, potresti avere una certa esperienza nell'uso di un sistema di gestione di database relazionali (RDBMS) e di un linguaggio di interrogazione strutturato (SQL). Quando si inizia a lavorare con Amazon QLDB, è possibile che siano riscontrate molte somiglianze. Man mano che passi ad argomenti più avanzati, incontrerai anche nuove potenti funzionalità che QLDB ha costruito sulla base RDBMS. In questa sezione vengono descritti i componenti e confrontati con i loro equivalenti in QLDB.

Il diagramma seguente mostra i costrutti di mappatura dei componenti principali tra un RDBMS tradizionale e Amazon QLDB.



La tabella seguente mostra le principali somiglianze e differenze di alto livello delle funzionalità operative integrate tra un RDBMS tradizionale e un QLDB.

Operazioni	RDBMS	QLDB
Creazione di tabelle	`CREATE TABLE`dichiarazione che definisce tutti i nomi delle colonne e i tipi di dati	`CREATE TABLE`dichiarazione che non definisce alcun attributo o tipo di dati della tabella per consentire contenuti aperti e privi di schemi
Creazione di indici	Dichiarazione `CREATE INDEX`	`CREATE INDEX`dichiarazione per tutti i campi di primo livello di una tabella
Inserimento dei dati	`INSERT`dichiarazione che specifica i valori all'interno di una nuova riga o tupla che	`INSERT`dichiarazione che specifica i valori all'interno di un nuovo documento in

Operazioni	RDBMS	QLDB
	aderisce allo schema definito dalla tabella	qualsiasi formato Amazon Ion valido, indipendentemente dai documenti esistenti nella tabella
Query sui dati	Dichiarazione `SELECT-FROM-WHERE`	`SELECT-FROM-WHERE`dichiarazione con la stessa sintassi di SQL quando si interrogano documenti flat
Aggiornamento dei dati	Dichiarazione `UPDATE-SET-WHERE`	`UPDATE-SET-WHERE`dichiarazione con la stessa sintassi di SQL quando si aggiornano i documenti flat
Eliminazione di dati	Dichiarazione `DELETE-FROM-WHERE`	`DELETE-FROM-WHERE`dichiarazione con la stessa sintassi di SQL quando si eliminano documenti flat
Dati annidurati e semistrut turati	Solo file o tuple piatte	Documenti che possono contenere dati strutturati, semistrutturati o annidati, supportati dal formato dati Amazon Ion e dal linguaggio di interrogazione PartiQL
Interrogazione dei metadati	Nessun metadati integrato	`SELECT`dichiarazione che esegue l'interrogazione dalla visualizzazione integrata di una tabella
Query sulla cronologia delle modifiche	Nessuna cronologia di dati integrata	`SELECT`dichiarazione che interroga la funzione di cronologia integrata

Operazioni	RDBMS	QLDB
Verifica crittografica	Nessuna crittografia o immutabilità integrate	API che restituiscono un riassunto di un diario e una prova che verifica l'integrità di qualsiasi revisione del documento relativa a quel riassunto

Per una panoramica dei concetti e della terminologia principali in QLDB, vedereConcetti principali.

Per informazioni dettagliate sul processo di creazione, interrogazione e gestione dei dati in un libro mastro, vedereUtilizzo dei dati e della cronologia.

Concetti e terminologia fondamentali in Amazon QLDB

Questa sezione fornisce una panoramica dei concetti e della terminologia principali di Amazon QLDB, inclusa la struttura dei registri e il modo in cui un libro mastro gestisce i dati. In quanto database contabile, QLDB si differenzia dagli altri database orientati ai documenti per quanto riguarda i seguenti concetti chiave.

Argomenti

- Modello di oggetti dati QLDB
- Transazioni che iniziano con il diario
- Esecuzione di query sui dati dei dati
- Archiviazione dati
- Modello di API QLDB API QLDB API
- Fasi successive

Modello di oggetti dati QLDB

Il modello di oggetto dati fondamentale in Amazon QLDB è descritto come segue:

1. libro mastro

Il primo passo consiste nel creare un libro mastro, che è il tipo diAWS risorsa principale in QLDB. Per informazioni su come creare un libro mastro, consultaFase 1: creazione di un nuovo libro mastro la sezione Guida introduttiva alla console oOperazioni di base per i libri mastri Amazon QLDB.

Sia per la`ALLOW_ALL` modalità di autorizzazione che per`STANDARD` le autorizzazioni di un libro contabile, si creano politicheAWS Identity and Access Management (IAM) che concedono le autorizzazioni per eseguire operazioni API su questa risorsa contabile.

Formato ARN del contabilità contabilità Ledger ARN

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

2. Diario e tabelle

   Per iniziare a scrivere dati in un libro mastro QLDB, devi prima creare una tabella con unaCREATE TABLE dichiarazione di base. I dati contabili sono costituiti da revisioni di documenti che vengono inseriti nel giornale contabile. Le revisioni dei documenti vengono salvate nel registro nel contesto di tabelle definite dall'utente. In QLDB, una tabella rappresenta una vista materializzata di una raccolta di revisioni di documenti dal giornale.

   Nella modalità delle`STANDARD` autorizzazioni di un libro mastro, è necessario creare politiche IAM che concedano le autorizzazioni per eseguire istruzioni PartiQL su questa risorsa della tabella. Con le autorizzazioni su una risorsa della tabella, è possibile eseguire istruzioni che accedono allo stato corrente della tabella. Puoi anche interrogare la cronologia delle revisioni della tabella utilizzando la`history()` funzione integrata.

   Formato ARN della Tabella ARN della Tabella ARN

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

   Per ulteriori informazioni sulla concessione delle autorizzazioni su un libro contabilità contabilità contabilità contabilità e risorse associate, consultaCome funziona Amazon QLDB con IAM.

3. Documenti

Le tabelle sono costituite da revisioni diDocumentazione QLDB, che sono set di dati in`struct` formato Amazon Ion. Una revisione del documento rappresenta una singola versione di una sequenza di documenti identificati da un ID di documento univoco.

QLDB memorizza la cronologia completa delle modifiche dei documenti salvati. Una tabella consente di interrogare lo stato corrente dei documenti, mentre la`history()` funzione consente di interrogare l'intera cronologia delle revisioni dei documenti di una tabella. Per informazioni dettagliate sull'interrogazione e la scrittura delle revisioni, consultaUtilizzo dei dati e della cronologia.

4. Catalogo di sistema

   Ogni libro contabile fornisce anche una risorsa di catalogo definita dal sistema che è possibile interrogare per elencare tutte le tabelle e gli indici di un libro mastro. Nella modalità delle`STANDARD` autorizzazioni di un libro mastro, è necessaria l'`qldb:PartiQLSelect`autorizzazione su questa risorsa del catalogo per eseguire le seguenti operazioni:

   - Eseguì`SELECT` le istruzioni nella tabella del catalogo di sistema information_schema.user_tables.
   - Visualizza le informazioni sulla tabella e sull'indice nella pagina dei dettagli del libro contabile sulla console QLDB.
   - Visualizza l'elenco delle tabelle e degli indici nell'editor PartiQL sulla console QLDB.

   Formato ARN del catalogo ARN del catalogo ARN

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

Transazioni che iniziano con il diario

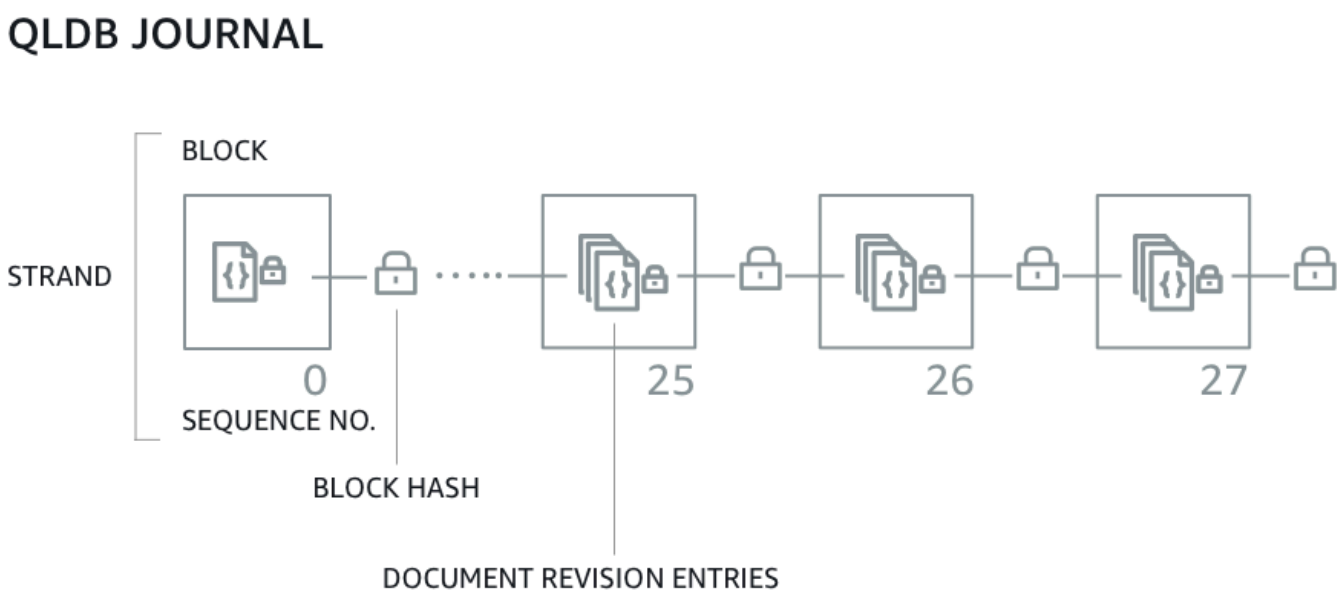
Quando un'applicazione legge o scrive dati in un registro QLDB, lo fa in una transazione di database. Tutte le transazioni sono soggette ai limiti definiti inQuote e limiti in Amazon QLDB. All'interno di una transazione, QLDB esegue i seguenti passaggi:

1. Leggi lo stato corrente dei dati del libro contabilità corrente dei dati del libro contabilità corrente dei dati del libro mastro.

2. Esegui le istruzioni fornite nella transazione, quindi verifica la presenza di eventuali conflitti utilizzando il controllo ottimistico della concorrenza (OCC) per garantire un isolamento completamente serializzabile.
3. Se non vengono rilevati conflitti OCC, restituisci i risultati della transazione come segue:
   - Per le letture, restituisci il set di risultati eSELECT salva le dichiarazioni nel diario in modo di sola aggiunta.
   - Per le scritture, salva gli aggiornamenti, le eliminazioni o i dati appena inseriti nel diario in modalità di sola aggiunta.

Il diario rappresenta una cronologia completa e immutabile di tutte le modifiche ai dati. QLDB scrive un blocco concatenato nel journal in una transazione. Ogni blocco contiene oggetti di immissione che rappresentano le revisioni del documento inserite, aggiornate ed eliminate, insieme alle istruzioni PartiQL che le hanno salvate.

Il diagramma seguente illustra questa struttura del giornale.



Il diagramma mostra che le transazioni vengono salvate nel giornale come blocchi contenenti le voci di revisione del documento. Ogni blocco viene sottoposto a hashing e concatenato ai blocchi successivi per la verifica. Ogni blocco ha un numero di sequenza per specificare il suo indirizzo all'interno del filamento.

> **Note**
>
> In Amazon QLDB, un filone è una partizione del giornale contabile. Attualmente QLDB supporta le riviste con un solo filamento.

Per informazioni sul contenuto dei dati in un blocco, vedereContenuto del diario in Amazon QLDB.

Esecuzione di query sui dati dei dati

QLDB è destinato a soddisfare le esigenze dei carichi di lavoro OLTP (Online Transaction Processing, elaborazione di transazioni online). Un libro mastro fornisce visualizzazioni tabellari interrogabili dei dati in base alle informazioni sulle transazioni inserite nel giornale. Una visualizzazione tabellare in QLDB è un sottoinsieme dei dati in una tabella. Le visualizzazioni vengono mantenute in tempo reale, in modo che siano sempre disponibili per le interrogazioni da parte delle applicazioni.

È possibile interrogare le seguenti viste definite dal sistema utilizzando`SELECT` istruzioni PartiQL:

- Utente: l'ultima revisione attiva dei soli dati che hai scritto nella tabella (ovvero lo stato corrente dei dati utente). Questa è la visualizzazione predefinita in QLDB.
- Impegnato: l'ultima revisione attiva dei dati utente e dei metadati generati dal sistema. Questa è la tabella completa definita dal sistema che corrisponde direttamente alla tabella utente.

Oltre a queste viste interrogabili, puoi interrogare la cronologia delle revisioni dei tuoi dati utilizzando la visualizzazione integrataFunzione di cronologia. La funzione di cronologia restituisce sia i dati utente che i metadati associati nello stesso schema della vista confermata.

Archiviazione dati

Esistono due tipi di archiviazione dati dati dati in QLDB:

- Archiviazione delle scritture contabili: lo spazio su disco utilizzato dal giornale contabile. Il diario è solo per le appendici e contiene la cronologia completa, immutabile e verificabile di tutte le modifiche ai dati.
- Archiviazione indicizzata: lo spazio su disco utilizzato dalle tabelle, dagli indici e dalla cronologia indicizzata di un libro contabile. Lo storage indicizzato è costituito da dati contabilità contabili che sono ottimizzati per le query ad alte prestazioni.

Dopo che i dati sono stati salvati nel giornale, vengono materializzati nelle tabelle che hai definito. Queste tabelle sono ottimizzate per eseguire interrogazioni più rapide ed efficienti. Quando un'applicazione utilizza l'API dei dati transazionali per leggere i dati, accede alle tabelle e agli indici archiviati nell'archivio indicizzato.

Modello di API QLDB API QLDB API

QLDB fornisce due tipi di API con cui il codice dell'applicazione può interagire:

- Amazon QLDB: l'API di gestione delle risorse QLDB (nota anche come piano di controllo). Questa API viene utilizzata solo per la gestione delle risorse contabili e per le operazioni sui dati non transazionali. È possibile utilizzare queste operazioni per creare, eliminare, elencare e aggiornare i libri contabili. Puoi anche verificare i dati crittograficamente ed esportare o trasmettere blocchi di journal.
- Amazon QLDB Session: l'API per i dati transazionali QLDB. È possibile utilizzare questa API per eseguire transazioni di dati su un libro mastro con istruzioni PartiQL.

> **Important**
>
> Invece di interagire direttamente con l'API di sessione QLDB, si consiglia di utilizzare il driver QLDB o la shell QLDB per eseguire transazioni di dati su un libro mastro.
>
> - Se stai lavorando con unAWS SDK, usa il driver QLDB. Il driver fornisce un livello di astrazione di alto livello sopra l'API dei dati della sessione QLDB e gestisce l'`SendCommand`operazione per te. Per informazioni e un elenco dei linguaggi di programmazione supportati, vedereNozioni base sul driver.
> - Se stai lavorando conAWS CLI, usa la shell QLDB. La shell è un'interfaccia a riga di comando che utilizza il driver QLDB per interagire con un libro mastro. Per informazioni, consulta Utilizzo della shell Amazon QLDB (solo API dati).

Per ulteriori informazioni su queste operazioni API, consulta ilDocumentazione di riferimento dell'API Amazon QLDB.

Fasi successive

Per imparare a utilizzare un libro mastro con i tuoi dati, consultaUtilizzo di dati e cronologia in Amazon QLDB e segui gli esempi che descrivono il processo di creazione di tabelle, inserimento di

dati ed esecuzione di interrogazioni di base. Questa guida spiega in modo approfondito come questi concetti funzionano, utilizzando dati di esempio ed esempi di interrogazioni per il contesto.

Per iniziare rapidamente con un tutorial applicativo di esempio utilizzando la console QLDB, consultaNozioni di base sulla console Amazon QLDB.

Per un elenco dei termini e delle definizioni chiave descritti in questa sezione, vedere ilGlossario Amazon QLDB.

Contenuto del diario in Amazon QLDB

In Amazon QLDB, il journal è il registro transazionale immutabile che memorizza la cronologia completa e verificabile di tutte le modifiche ai dati. Il journal è disponibile solo per le appendici ed è composto da un set di blocchi sequenziati e concatenati in hash che contengono i dati salvati e altri metadati di sistema. QLDB scrive un blocco concatenato nel journal in una transazione.

Questa sezione fornisce un esempio di blocco di diario con dati di esempio e descrive il contenuto di un blocco.

Argomenti

Esempio di blocco

Un blocco di diario contiene i metadati delle transazioni insieme a voci che rappresentano le revisioni dei documenti che sono state commesse nella transazione e le dichiarazioni di PartiQL che le hanno commesse.

Di seguito è riportato un esempio di blocco con dati di esempio.

> **Note**
>
> Questo esempio di blocco viene fornito solo a scopo informativo. Gli hash mostrati non sono valori hash reali calcolati.

```
{
  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"
      }
    }
  ]
}
```

Nel`revisions` campo, alcuni oggetti di revisione potrebbero contenere solo un`hash` valore e nessun altro attributo. Si tratta di revisioni del sistema solo interne che non contengono dati utente. Gli hash di queste revisioni fanno parte dell'intera catena di hash della rivista, necessaria per la verifica crittografica.

Blocca contenuti

Un blocco del diario ha i campi riportati di seguito:

`blockAddress`

La posizione del blocco nel diario. Un indirizzo è una struttura Amazon Ion con due campi:`strandId` e`sequenceNo`.

Ad esempio: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`

`transactionId`

L'ID univoco della transazione che ha commesso il blocco.

`blockTimestamp`

Il timestamp in cui il blocco è stato salvato nel journal.

`blockHash`

Il valore hash a 256 bit che rappresenta in modo univoco il blocco. Questo è l'hash della concatenazione di`entriesHash` e`previousBlockHash`.

`entriesHash`

L'hash che rappresenta tutte le voci all'interno del blocco, incluse le voci di sistema solo interne. Questo è l'hash principale dell'albero Merkle in cui i nodi delle foglie sono costituiti da tutti gli hash in cui sono costituiti`entriesHashList`.

`previousBlockHash`

L'hash del precedente blocco concatenato nel journal.

`entriesHashList`

L'elenco degli hash che rappresentano ogni voce all'interno del blocco. Questo elenco può includere gli hash della voce riportati di seguito:

- L'hash Ion che rappresenta`transactionInfo`. Questo valore viene calcolato prendendo l'hash Ion dell'intera`transactionInfo` struttura.
- L'hash della radice dell'albero Merkle in cui i nodi delle foglie sono costituiti da tutti gli hash in cui sono contenuti`revisions`.
- L'hash Ion che rappresenta`redactionInfo`. Questo hash esiste solo nei blocchi che sono stati commessi da una transazione di redazione. Il suo valore viene calcolato prendendo l'hash Ion dell'intera`redactionInfo` struttura.
- Hash che rappresentano metadati di sistema solo interni. Questi hash potrebbero non esistere in tutti i blocchi.

`transactionInfo`

Una struttura Amazon Ion che contiene informazioni sulle dichiarazioni della transazione che ha commesso il blocco. Questa struttura include i campi riportati di seguito:

- `statements`— L'elenco delle istruzioni PartiQL e il`startTime` momento in cui hanno iniziato a funzionare. Ogni istruzione ha un`statementDigest` hash, necessario per calcolare l'hash della`transactionInfo` struttura.

- `documents`— Gli ID dei documenti che sono stati aggiornati dalle dichiarazioni. Ogni documento include il`tableName` e `atableId` cui appartiene e l'indice di ogni dichiarazione che lo ha aggiornato.

`revisions`

L'elenco delle revisioni dei documenti che sono state eseguite nel blocco. Ogni struttura di revisione contiene tutti i campi della visualizzazione confermata della revisione.

Ciò può includere anche hash che rappresentano revisioni di sistema solo interne che fanno parte dell'intera catena di hash di un journal.

Revisioni redatte

In Amazon QLDB, un'`DELETE`istruzione elimina logicamente un documento solo creando una nuova revisione che lo contrassegna come eliminato. QLDB supporta anche un'operazione di redazione dei dati che consente di eliminare definitivamente le revisioni di documenti inattive nella cronologia di una tabella.

L'operazione di redazione elimina solo i dati dell'utente nella revisione specificata e lascia invariati la sequenza del diario e i metadati del documento. Ciò mantiene l'integrità complessiva dei dati del registro. Per ulteriori informazioni e un esempio di operazione di redazione, consultaRedazione delle revisioni dei documenti.

Esempio di revisione redatto

Considerate l'esempio di blocco precedente. In questo blocco, supponiamo di redigere la revisione che ha un ID del documento`HwVFkn8IMRa0xjze5xcgga` e un numero di versione di`0`.

Una volta completata la redazione, i dati utente nella revisione (rappresentati dalla`data` struttura) vengono sostituiti da un nuovo`dataHash` campo. Il valore di questo campo è l'hash Ion della`data` struttura rimossa. Di conseguenza, il registro mantiene l'integrità complessiva dei dati e rimane verificabile crittograficamente attraverso le operazioni dell'API di verifica esistenti.

Il seguente esempio di revisione mostra i risultati di questa redazione, con il nuovo`dataHash` campo evidenziato in *`corsivo rosso`*.

> **Note**
>
> Questo esempio di revisione viene fornito solo a scopo informativo. Gli hash mostrati non sono valori hash reali calcolati.

```
...
{
  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 aggiunge anche un nuovo blocco al giornale per la richiesta di redazione completata. Questo blocco include una`redactionInfo` voce aggiuntiva che contiene un elenco di revisioni che sono state cancellate nella transazione, come illustrato nell'esempio seguente.

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

```
...
```

Applicazione di esempio

Per un esempio di codice Java che convalida la catena hash di un journal utilizzando dati esportati, consulta il GitHub repository aws-samples/amazon-qldb-dmv-sample -java. Questa applicazione di esempio include i seguenti file di classe:

- ValidateQldbHashChain.java — Contiene un codice tutorial che esporta i blocchi di diario da un libro mastro e utilizza i dati esportati per convalidare la catena di hash tra i blocchi.
- JournalBlock.java — Contiene un metodo denominato`verifyBlockHash()` che dimostra come calcolare ogni singolo componente hash all'interno di un blocco. Questo metodo viene richiamato dal codice del tutorial in`ValidateQldbHashChain.java`.

Per istruzioni su come scaricare e installare questa applicazione di esempio completa, vedereInstallazione dell'applicazione di esempio Java Amazon QLDB. Prima di eseguire il codice del tutorial, assicurati di seguire i passaggi da 1Tutorial su Java a 3 per configurare un registro di esempio e caricarlo con dati di esempio.

Consulta anche

Per ulteriori informazioni sui giornali in QLDB, consulta i seguenti argomenti:

- Esportazione dei dati del diario da Amazon QLDB— Per scoprire come esportare i dati del diario in Amazon Simple Storage Service (Amazon S3).
- Streaming dei dati del diario da Amazon QLDB— Per scoprire come trasmettere i dati del diario in Amazon Kinesis Data Streams.
- Verifica dei dati in Amazon QLDB— Per conoscere la verifica crittografica dei dati del giornale.

Glossario Amazon QLDB

Di seguito sono riportate le definizioni dei termini chiave che potrebbero essere presenti durante l'utilizzo di Amazon QLDB.

blocco | digerire | documento | ID del documento | revisione del documento | ingresso | field | index | archiviazione indicizzata | diario | blocco diario | storage del diario | filone del diario | suggerimento sul diario | libro mastro | prova | revisione | session | filo | table | vista da tavolo | vista

blocco

Un oggetto che viene inserito nel giornale di registrazione in una transazione. Una singola transazione scrive un blocco nel giornale, quindi un blocco può essere associato a una sola transazione. Un blocco contiene le voci che rappresentano le revisioni dei documenti eseguite nella transazione, insieme alle dichiarazioni PartiQL che le hanno eseguite.

Ogni blocco ha anche un valore hash per la verifica. Un hash di blocco viene calcolato dagli hash di ingresso all'interno di quel blocco combinati con l'hash del blocco concatenato precedente.

digerire

Un valore hash a 256 bit che rappresenta in modo univoco l'intera cronologia delle revisioni dei documenti del registro in un determinato momento. Un digest hash viene calcolato a partire dall'intera catena di hash del diario a partire dall'ultimo blocco salvato nel journal in quel momento.

QLDB consente di generare un digest come file di output sicuro. Quindi, puoi utilizzare quel file di output per verificare l'integrità delle revisioni del documento rispetto a quell'hash.

documento

Un set di dati in `struct` formato Amazon Ion che può essere inserito, aggiornato ed eliminato in una tabella. Un documento QLDB può contenere dati strutturati, semistrutturati, annidati e senza schemi.

ID del documento

L'identificatore universale univoco (UID, Universally Unique Identifier) che QLDB assegna a ogni documento che si inserisca in una tabella. Questo ID è un numero a 128 bit rappresentato in una stringa alfanumerica codificata in Base62 con una lunghezza fissa di 22 caratteri.

revisione del documento

Una struttura a ioni che rappresenta una singola versione di una sequenza di documenti identificati da un ID di documento univoco. Una revisione include sia i dati utente (ovvero i dati che hai scritto nella tabella) sia i metadati generati dal sistema. Ogni revisione è associata a una tabella ed è identificata in modo univoco da una combinazione dell'ID del documento e di un numero di versione a base zero.

ingresso

Un oggetto contenuto in un blocco. Le voci rappresentano le revisioni dei documenti che vengono inserite, aggiornate ed eliminate in una transazione, insieme alle dichiarazioni di PartiQL che le hanno confermate.

Ogni voce ha anche un valore hash per la verifica. Un hash di ingresso viene calcolato dagli hash di revisione o dagli hash delle dichiarazioni all'interno di quella voce.

field

Una coppia nome-valore che costituisce ogni attributo di un documento QLDB. Il nome è un simbolo e il valore è illimitato.

index

Una struttura di dati che puoi creare su una tabella per ottimizzare le prestazioni delle operazioni di recupero dei dati. Per informazioni sugli indici in QLDB, consultaCREATE INDEX il riferimento Amazon QLDB PartiQL.

archiviazione indicizzata

Lo spazio su disco utilizzato dalle tabelle, dagli indici e dalla cronologia indicizzata di un libro mastro. L'archivio indicizzato è costituito da dati contabilità ottimizzati per query ad alte prestazioni.

diario

L'insieme concatenato di hash di tutti i blocchi salvati nel tuo registro. Il giornale è solo per le appendici e rappresenta una cronologia completa e immutabile di tutte le modifiche ai dati contabili.

blocco diario

Consultare blocco.

storage del diario

Lo spazio su disco utilizzato dal giornale contabile.

filone del diario

Consultare filo.

suggerimento sul diario

L'ultimo blocco di un diario in un determinato momento.

libro mastro

Un'istanza di una risorsa del database Amazon QLDB Ledger. Questo è il tipo diAWS risorsa principale in QLDB. Un libro mastro è costituito sia dall'archiviazione delle scritture contabili che

dall'archiviazione indicizzata. Dopo che i dati contabili sono stati salvati nel giornale, possono essere interrogati nelle tabelle delle revisioni dei documenti Amazon Ion.

prova

L'elenco ordinato di valori hash a 256 bit che QLDB restituisce per un determinato digest e revisione del documento. È costituito dagli hash richiesti da un modello ad albero Merkle per concatenare l'hash di revisione specificato all'hash digest. Utilizzi una prova per verificare l'integrità delle tue revisioni relative al riassunto. Per ulteriori informazioni, consulta Verifica dei dati in Amazon QLDB.

revisione

Consultare revisione del documento.

session

Un oggetto che gestisce le informazioni sulle richieste di transazioni di dati e sulle risposte da e verso un libro mastro. Una sessione attiva (una sessione che esegue attivamente una transazione) rappresenta una singola connessione a un libro mastro. QLDB supporta una transazione in esecuzione attiva per sessione.

filo

Una partizione di un diario. Attualmente QLDB supporta le riviste con un solo filamento.

table

Una visione materializzata di una raccolta non ordinata di revisioni dei documenti eseguite nel giornale contabile.

vista da tavolo

Un sottoinsieme interrogabile dei dati di una tabella, basato sulle transazioni registrate nel giornale. In un'istruzione PartiQL, una vista è indicata da un qualificatore di prefisso (che inizia con`_ql_`) per il nome di una tabella.

È possibile interrogare le seguenti viste definite dal sistema utilizzando`SELECT` le istruzioni:

- Utente: l'ultima revisione attiva dei soli dati che hai scritto nella tabella (ovvero lo stato corrente dei dati utente). Questa è la visualizzazione predefinita in QLDB.
- Impegnato: l'ultima revisione attiva dei dati utente e dei metadati generati dal sistema. Questa è la tabella completa definita dal sistema che corrisponde direttamente alla tabella utente. Ad esempio: `_ql_committed_`*`TableName`*.

vista

Consultare vista da tavolo.

Accesso ad Amazon QLDB

Puoi accedere ad Amazon QLDB utilizzando AWS Management Console l'API, AWS Command Line Interface the AWS CLI() o QLDB. Le seguenti sezioni descrivono come utilizzare queste opzioni e i prerequisiti per utilizzarle.

Prerequisiti

Prima di poter accedere a QLDB, devi configurarne Account AWS uno se non l'hai già fatto.

Argomenti

- Registrati per un Account AWS
- Crea un utente con accesso amministrativo
- Gestisci le autorizzazioni QLDB in IAM
- Concedere l'accesso programmatico (opzionale)

Registrati per un Account AWS

Se non ne hai uno Account AWS, completa i seguenti passaggi per crearne uno.

Per iscriverti a un Account AWS

1. Apri la pagina all'indirizzo https://portal.aws.amazon.com/billing/signup.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

   Quando ti iscrivi a un Account AWS, Utente root dell'account AWSviene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come procedura consigliata in materia di sicurezza, assegnate l'accesso amministrativo a un utente e utilizzate solo l'utente root per eseguire attività che richiedono l'accesso da parte dell'utente root.

AWS ti invia un'e-mail di conferma dopo il completamento della procedura di registrazione. È possibile visualizzare l'attività corrente dell'account e gestire l'account in qualsiasi momento accedendo all'indirizzo https://aws.amazon.com/ e selezionando Il mio account.

Crea un utente con accesso amministrativo

Dopo esserti registrato Account AWS, proteggi Utente root dell'account AWS AWS IAM Identity Center, abilita e crea un utente amministrativo in modo da non utilizzare l'utente root per le attività quotidiane.

Proteggi i tuoi Utente root dell'account AWS

1. Accedi AWS Management Consolecome proprietario dell'account scegliendo Utente root e inserendo il tuo indirizzo Account AWS email. Nella pagina successiva, inserisci la password.

   Per informazioni sull'accesso utilizzando un utente root, consulta la pagina Signing in as the root user della Guida per l'utente di Accedi ad AWS .
2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

   Per istruzioni, consulta Abilitare un dispositivo MFA virtuale per l'utente Account AWS root (console) nella Guida per l'utente IAM.

Crea un utente con accesso amministrativo

1. Abilita Centro identità IAM.

   Per istruzioni, consulta Abilitazione di AWS IAM Identity Center nella Guida per l'utente di AWS IAM Identity Center .
2. In IAM Identity Center, concedi l'accesso amministrativo a un utente.

   Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta Configurare l'accesso utente con le impostazioni predefinite IAM Identity Center directory nella Guida per l'AWS IAM Identity Center utente.

Accedi come utente con accesso amministrativo

- Per accedere con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

  Per informazioni sull'accesso utilizzando un utente IAM Identity Center, consulta AWS Accedere al portale di accesso nella Guida per l'Accedi ad AWS utente.

Assegna l'accesso ad altri utenti

1. In IAM Identity Center, crea un set di autorizzazioni che segua la migliore pratica di applicazione delle autorizzazioni con privilegi minimi.

   Per istruzioni, consulta Creare un set di autorizzazioni nella Guida per l'utente.AWS IAM Identity Center
2. Assegna gli utenti a un gruppo, quindi assegna l'accesso Single Sign-On al gruppo.

   Per istruzioni, consulta Aggiungere gruppi nella Guida per l'utente.AWS IAM Identity Center

Gestisci le autorizzazioni QLDB in IAM

Per informazioni sull'utilizzo di AWS Identity and Access Management (IAM) per gestire le autorizzazioni QLDB per gli utenti, vedere. Come funziona Amazon QLDB con IAM

Concedere l'accesso programmatico (opzionale)

Gli utenti necessitano dell'accesso programmatico se desiderano interagire con l' AWS esterno di. AWS Management Console Il modo per concedere l'accesso programmatico dipende dal tipo di utente che accede. AWS

Per fornire agli utenti l'accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell'accesso programmatico?	Per	Come
Identità della forza lavoro (Utenti gestiti nel centro identità IAM)	Utilizza credenziali temporane e per firmare le richieste programmatiche agli AWS CLI AWS SDK o alle API. AWS	Segui le istruzioni per l'interfa ccia che desideri utilizzare. • Per la AWS CLI, consulta Configurazione dell'uso AWS IAM Identity Center nella Guida AWS CLI per l'utente.AWS Command Line Interface • Per AWS SDK, strumenti e AWS API, consulta

Quale utente necessita dell'accesso programmatico?	Per	Come
		l'autenticazione IAM Identity Center nella Guida di riferimento agli AWS SDK e agli strumenti.
IAM	Utilizza credenziali temporane e per firmare le richieste programmatiche agli SDK o alle API AWS CLI. AWS AWS	Segui le istruzioni in Uso delle credenziali temporanee con AWS risorse nella Guida per l'utente IAM.
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare le richieste programmatiche agli AWS CLI AWS SDK o alle API. AWS	Segui le istruzioni per l'interfa ccia che desideri utilizzare. • Per la AWS CLI, consulta Autenticazione tramite credenziali utente IAM nella Guida per l'utente.AWS Command Line Interface • Per gli AWS SDK e gli strumenti, consulta Autenticazione tramite credenziali a lungo termine nella Guida di riferimen to agli SDK e agli AWS strumenti. • Per le AWS API, consulta Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM.

Come accedere ad Amazon QLDB

Dopo aver completato i prerequisiti per configurare un file Account AWS, consulta i seguenti argomenti per saperne di più su come accedere a QLDB:

- Utilizzo della console
- Utilizzando AWS CLI (solo l'API di gestione)
- Utilizzo della shell Amazon QLDB (solo API dati)
- Utilizzo dell'API

Accesso ad Amazon QLDB tramite la console

Puoi accedere a Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb. AWS Management Console

È possibile utilizzare la console per eseguire le seguenti operazioni in QLDB:

- Crea, elimina, descrivi ed elenca i registri.
- Esegui istruzioni PartiQL utilizzando l'editor PartiQL.
- Gestisci i tag per le risorse QLDB.
- Verifica crittograficamente i dati del diario.
- Esporta o trasmetti in streaming blocchi di journal.

Per informazioni su come creare un registro Amazon QLDB e configurarlo con dati applicativi di esempio, consulta. Nozioni di base sulla console Amazon QLDB

Guida rapida all'editor PartiQL

Amazon QLDB supporta un sottoinsieme di PartiQL come linguaggio di interrogazione e Amazon Ion come formato di dati orientato ai documenti. Per una guida completa e informazioni più dettagliate sull'implementazione QLDB di PartiQL, consulta. Documentazione di riferimento Amazon QLDB PartiQL

I seguenti argomenti forniscono una rapida panoramica di riferimento su come utilizzare PartiQL in QLDB.

Argomenti

- Suggerimenti rapidi su PartiQL in QLDB
- Comandi
- Viste definite dal sistema
- Regole di sintassi di base

- Scelte rapide da tastiera dell'editor PartiQL

Suggerimenti rapidi su PartiQL in QLDB

Di seguito è riportato un breve riepilogo di suggerimenti e best practice per lavorare con PartiQL in QLDB:

- Comprendi i limiti di concorrenza e transazione: tutte le dichiarazioni, comprese le `SELECT` interrogazioni, sono soggette a conflitti ottimistici di controllo della concorrenza (OCC) e a limiti di transazione, incluso un timeout della transazione di 30 secondi.
- Usa gli indici: utilizza indici ad alta cardinalità ed esegui query mirate per ottimizzare i rendiconti ed evitare scansioni complete delle tabelle. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.
- Usa i predicati di uguaglianza: le ricerche indicizzate richiedono un operatore di uguaglianza (o). `=` `IN` Gli operatori di disuguaglianza (<,, >`LIKE`,`BETWEEN`) non sono idonei per le ricerche indicizzate e generano scansioni complete della tabella.
- Usa solo i join interni: QLDB supporta solo i join interni. Come best practice, unisciti ai campi indicizzati per ogni tabella a cui ti stai unendo. Scegli indici ad alta cardinalità sia per i criteri di unione che per i predicati di uguaglianza.

Comandi

QLDB supporta i seguenti comandi PartiQL.

DDL (Data Definition Language)

Comando	Descrizione
CREATE INDEX	Crea un indice per un campo di documento di primo livello su una tabella.
CREATE TABLE	Crea una tabella.
DROP INDEX	Elimina un indice da una tabella.
DROP TABLE	Disattiva una tabella esistente.
TAVOLO UNDROP	Riattiva una tabella inattiva.

Linguaggio di manipolazione dei dati (DML)

Comando	Descrizione
DELETE	Contrassegna un documento attivo come eliminato creando una nuova revisione finale del documento.
DA (INSERISCI, RIMUOVI o IMPOSTA)	Semanticamente uguale a. UPDATE
INSERT	Aggiunge uno o più documenti a una tabella.
SELECT	Recupera i dati da una o più tabelle.
UPDATE	Aggiorna, inserisce o rimuove elementi specifici all'interno di un documento.

Esempi di istruzioni DML

INSERISCI

```
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
}
```

AGGIORNA-INSERISCI

```
UPDATE Vehicle AS v
INSERT INTO v VALUE 26500 AT 'Mileage'
WHERE v.VIN = '1N4AL11D75C109151'
```

AGGIORNA-RIMUOVI

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

SELECT — Sottoquery correlata

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

SELECT — Inner join

```
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 — Ottieni l'ID del documento utilizzando la clausola BY

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

Viste definite dal sistema

QLDB supporta le seguenti viste di una tabella definite dal sistema.

Vista	Descrizione
table_name	La visualizzazione utente predefinita di una tabella che include solo lo stato corrente dei dati utente.
_ql_committed_ *table_name*	La visualizzazione confermata completa definita dal sistema di una tabella che include lo stato corrente dei dati utente e dei metadati generati dal sistema, come l'ID di un documento.
history(*table_name*)	La funzione di cronologia integrata che restituisce la cronologia completa delle revisioni di una tabella.

Regole di sintassi di base

QLDB supporta le seguenti regole di sintassi di base per PartiQL.

Carattere	Descrizione
'	Le virgolette singole indicano valori di stringa o nomi di campo nelle strutture Amazon Ion.
"	Le virgolette doppie indicano identificatori tra virgolette, ad esempio una parola riservata utilizzata come nome di tabella.
`	I backtick indicano valori letterali Ion.
.	La notazione a punti accede ai nomi dei campi di una struttura principale.
[ ]	Le parentesi quadre definiscono uno ione `list` o indicano un numero ordinale a base zero per un elenco esistente.
{ }	Le parentesi graffe definiscono uno ione. `struct`
<< >>	Le parentesi a doppio angolo definiscono una borsa PartiQl, che è una raccolta non ordinata. Si utilizza una borsa per inserire più documenti in una tabella.
Distinzione tra lettere maiuscole e minuscole	Tutti i nomi degli oggetti del sistema QLDB, inclusi i nomi dei campi e i nomi delle tabelle, fanno distinzione tra maiuscole e minuscole.

Scelte rapide da tastiera dell'editor PartiQL

L'editor PartiQL sulla console QLDB supporta le seguenti scorciatoie da tastiera.

Azione	macOS	Windows
Esecuzione	Cmd+Return	Ctrl+Enter
Commento	Cmd+/	Ctrl+/

Azione	macOS	Windows
Annulla	Cmd+Shift+Delete	Ctrl+Shift+Delete

Accesso ad Amazon QLDB tramite (solo API AWS CLI di gestione)

È possibile utilizzare il AWS Command Line Interface (AWS CLI) per controllare più elementi Servizi AWS dalla riga di comando e automatizzarli tramite script. È possibile utilizzarlo AWS CLI per operazioni una tantum, se necessario. Puoi anche usarlo per incorporare le operazioni di Amazon QLDB negli script di utilità.

Per l'accesso alla CLI, sono necessari un ID chiave di accesso e una chiave di accesso segreta. Utilizza credenziali temporanee al posto delle chiavi di accesso a lungo termine quando possibile. Le credenziali temporanee includono un ID della chiave di accesso, una chiave di accesso segreta e un token di sicurezza che ne indica la scadenza. Per ulteriori informazioni, consulta Using temporary credenziali with AWS resources nella IAM User Guide.

Per un elenco completo ed esempi di utilizzo di tutti i comandi disponibili per QLDB in, vedere AWS CLI Command AWS CLI Reference.

> **Note**
>
> Supporta AWS CLI solo le operazioni dell'API di `qldb` gestione elencate in. Documentazione di riferimento dell'API Amazon QLDB Questa API viene utilizzata solo per la gestione delle risorse di registro e per le operazioni sui dati non transazionali.
> Per eseguire transazioni di dati con l'`qldb-session`API utilizzando un'interfaccia a riga di comando, consulta. Accesso ad Amazon QLDB tramite la shell QLDB (solo API dati)

Argomenti

- Installazione e configurazione di AWS CLI
- Utilizzo di AWS CLI con QLDB

Installazione e configurazione di AWS CLI

AWS CLI Funziona su Linux, macOS o Windows. Per installarlo e configurarlo, consulta le seguenti istruzioni nella Guida per l'AWS Command Line Interface utente:

1. Installazione o aggiornamento della versione più recente di AWS CLI
2. Configurazione rapida

Utilizzo di AWS CLI con QLDB

Il formato della riga di comando è costituito da un nome di operazione Amazon QLDB, seguito dai parametri per tale operazione. AWS CLI Supporta una sintassi abbreviata per i valori dei parametri, oltre a JSON.

Utilizzare `help` per elencare tutti i comandi disponibili in QLDB:

```
aws qldb help
```

Puoi anche usare `help` per descrivere un comando specifico e saperne di più sul suo utilizzo:

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

Ad esempio, per creare un libro mastro:

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

Accesso ad Amazon QLDB tramite la shell QLDB (solo API dati)

Amazon QLDB fornisce una shell a riga di comando per l'interazione con l'API dei dati transazionali. Con la shell QLDB, è possibile eseguire istruzioni PartiQL sui dati contabili.

L'ultima versione di questa shell è scritta in Rust ed è open source nel GitHub repository awslabs/amazon-qldb-shell sul`main` ramo predefinito. Anche la versione Python (v1) è ancora disponibile per l'uso nello stesso repository del`master` ramo.

> **Note**
>
> La shell Amazon QLDB supporta solo l'API dei dati`qldb-session` transazionali. Questa API viene utilizzata solo per eseguire istruzioni PartiQL su un registro QLDB.
> Per interagire con le operazioni dell'API di`qldb` gestione utilizzando un'interfaccia a riga di comando, vedereAccesso ad Amazon QLDB tramite (solo API AWS CLI di gestione).

Questo strumento non è destinato a essere incorporato in un'applicazione o adottato per scopi di produzione. L'obiettivo di questo strumento è permetterti di sperimentare rapidamente con QLDB e PartiQL.

Le sezioni seguenti descrivono come iniziare a usare la shell QLDB.

Argomenti

Prerequisiti

Prima di iniziare a usare la shell QLDB, bisogna eseguire le seguenti operazioni:

1. Segui le istruzioniAWS di configurazione inAccesso ad Amazon QLDB. Questo include gli output seguenti:

    1. Registrazione aAWS.
    2. Crea un utente con le autorizzazioni QLDB appropriate.
    3. Concessione dell'accesso programmatico per lo sviluppo.
2. ImpostaAWS le tue credenziali e quelle predefiniteRegione AWS. Per istruzioni, vedere Nozioni di base sulla configurazione nella Guida per l'AWS Command Line Interfaceutente.

    Per un elenco completo delle regioni disponibili, consulta gli endpoint e le quote di Amazon QLDB nel Riferimenti generali di AWS.
3. Per tutti i registri in modalità`STANDARD` autorizzazioni, crea policy IAM che ti concedano le autorizzazioni per eseguire istruzioni PartiQL sulle tabelle appropriate. Per informazioni su come

creare queste policy, consultaGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

Installa la shell

Per installare la versione più recente della shell QLDB, consulta il file README.md su GitHub. QLDB fornisce file binari predefiniti per Linux, macOS e Windows nella sezione Releases del GitHub repository.

Per macOS, la shell si integra con il rubinettoaws/tap Homebrew. Per installare la shell su macOS utilizzando Homebrew, eseguire i comandi seguenti.

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

Configurazione

Dopo l'installazione, la shell carica il file di configurazione predefinito che si trova$XDG_CONFIG_HOME/qldbshell/config.ion durante l'inizializzazione. Su Linux e macOS, questo file si trova in genere in~/.config/qldbshell/config.ion. Se tale file non esiste, la shell viene eseguita con le impostazioni predefinite.

È possibile creare unconfig.ion file manualmente dopo l'installazione. Questo file di configurazione utilizza il formato dati Amazon Ion. Di seguito è riportato un esempio diconfig.ion file minimo.

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

Sedefault_ledger non è impostato nel file di configurazione, il--ledger parametro è necessario quando si richiama la shell. Per un elenco completo delle opzioni di configurazione, consulta il file README.md su GitHub.

Invocazione della shell

Per richiamare la shell QLDB sul terminale della riga di comando per un registro specifico, esegui il comando seguente. Sostituisci *my-example-ledger*con il nome del tuo libro contabile.

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

Questo comando si collega all'impostazione predefinitaRegione AWS. Per specificare in modo esplicito la regione, è possibile eseguire il comando con il`--qldb-session-endpoint` parametro`--region` or, come descritto nella sezione seguente.

Dopo aver richiamato una sessione`qldb` shell, puoi inserire i seguenti tipi di input:

- Comandi Shell
- Dichiarazioni PartiQL singole in transazioni separate
- Più istruzioni PartiQL all'interno di una transazione

Parametri Shell

Per un elenco completo dei flag e delle opzioni disponibili per richiamare una shell, esegui il`qldb` comando con il`--help` flag, come segue.

```
$ qldb --help
```

Di seguito sono riportati alcuni contrassegni chiave e opzioni per il`qldb` comando. È possibile aggiungere questi parametri opzionali per sovrascrivere il profilo delle credenzialiRegione AWS, l'endpoint, il formato dei risultati e altre opzioni di configurazione.

Utilizzo

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

BANDIERE

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

Stampa le informazioni di aiuto.

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

Configura la verbosità della registrazione. Per impostazione predefinita, la shell registra solo gli errori. Per aumentare il livello di verbosità, ripeti questo argomento (ad esempio,`-vv`). Il livello più alto è quello`-vvv` che corrisponde alla`trace` verbosità.

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

Stampa le informazioni relative alla versione.

OPTIONS

`-l`,**`--ledger`** ***`NOME_REGISTRO`***

Il nome del libro mastro a cui connettersi. Questo è un parametro di shell obbligatorio se `default_ledger` non è impostato nel tuo `config.ion` file. In questo file, puoi impostare opzioni aggiuntive, come la regione.

`-c`,**`--config`** ***`FILE DI CONFIGURAZIONE`***

Il file in cui è possibile definire qualsiasi opzione di configurazione della shell. Per i dettagli sulla formattazione e un elenco completo delle opzioni di configurazione, consulta il file README.md su GitHub.

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

Il formato di output dei risultati della ricerca. Il valore predefinito è `ion`.

`-p`,**`--profile`** ***`PROFILO`***

La posizione del tuo profilo diAWS credenziali da utilizzare per l'autenticazione.

Se non viene fornito, la shell utilizza ilAWS profilo predefinito, che si trova in `~/.aws/credentials`.

`-r`,**`--region`** ***`CODICE REGIONALE`***

IlRegione AWS codice del registro QLDB a cui connettersi. Ad esempio: `us-east-1`.

Se non viene fornita, la shell si connette all'impostazione predefinitaRegione AWS come specificato nelAWS profilo.

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

L'endpoint `qldb-session` API a cui connettersi.

Per un elenco completo degli endpoint e delle Regioni QLDB disponibili, consulta Endpoint e quote di Amazon QLDB in Riferimenti generali di AWS.

Riferimento del comando

Dopo aver richiamato una`qldb` sessione, la shell supporta le seguenti chiavi e comandi del database:

Chiavi Shell

Chiave	Descrizione della funzione
Enter	Esegue la dichiarazione.
Escape+Enter (macOS, Linux) Shift+Enter (Windows)	Inizia una nuova riga per inserire un'istruzione che si estende su più righe. Puoi anche copiare il testo di input con più righe e incollarlo nella shell. Per istruzioni sulla configurazioneOption anzichéEscape come chiave Meta in macOS, consulta il sito OS X Daily.
Ctrl+C	Annulla il comando corrente.
Ctrl+D	Segnala la fine del file (EOF) ed esce dal livello corrente della shell. Se non è in una transazio ne attiva, esce dalla shell. In una transazione attiva, interrompe la transazione.

Comandi del database Shell

Comando	Descrizione della funzione
`help`	Visualizza le informazioni di aiuto.
`begin`	Inizia una transazione.
`start transaction`	
`commit`	Salva la transazione nel giornale contabile.

Comando	Descrizione della funzione
`abort`	Interrompe la transazione e rifiuta tutte le modifiche apportate.
`exit`	Esce dal guscio.
`quit`	

> **Note**
>
> Tutti i comandi shell QLDB non distinguono tra maiuscole e minuscole.

Esecuzione di dichiarazioni individuali

Ad eccezione dei comandi del database e dei meta comandi della shell elencati in README.md, la shell interpreta ogni comando immesso come un'istruzione PartiQL separata. Per impostazione predefinita, la shell abilita`auto-commit` la modalità. Questa modalità è configurabile.

Nella`auto-commit` modalità, la shell esegue implicitamente ogni istruzione nella propria transazione e la commette automaticamente se non vengono rilevati errori. Ciò significa che non è necessario eseguire`start transaction` (o`begin`) e`commit` manualmente ogni volta che si esegue un'istruzione.

Gestione delle transazioni

In alternativa, la shell QLDB consente di controllare manualmente le transazioni. È possibile eseguire più istruzioni all'interno di una transazione in modo interattivo o non interattivo raggruppando in batch comandi e istruzioni in sequenza.

Transazioni interattive

Per eseguire una transazione interattiva, completa la seguente procedura.

1. Per iniziare una transazione, inserisci il`begin` comando.

```
qldb> begin
```

Dopo aver iniziato una transazione, la shell visualizza il seguente prompt dei comandi.

```
qldb *>
```

2. Quindi, ogni dichiarazione inserita viene eseguita nella stessa transazione.
   - Ad esempio, puoi eseguire una singola istruzione come segue.

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

     Dopo aver premutoEnter, la shell visualizza i risultati dell'istruzione.
   - È inoltre possibile immettere più istruzioni o comandi separati da un delimitatore da punto e virgola (`;`) come segue.

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

3. Per terminare la transazione, inserisci uno dei comandi seguenti.
   - Inserisci il`commit` comando per salvare la transazione nel giornale contabile.

     ```
     qldb *> commit
     ```
   - Inserisci il`abort` comando per interrompere la transazione e rifiutare qualsiasi modifica apportata.

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

Limite di timeout delle transazioni

Una transazione interattiva rispetta il limite di timeout delle transazioni di QLDB. Se non confermi una transazione entro 30 secondi dall'avvio, QLDB fa scadere automaticamente la transazione e rifiuta qualsiasi modifica apportata durante la transazione.

Quindi, invece di visualizzare i risultati dell'istruzione, la shell visualizza un messaggio di errore di scadenza e torna al normale prompt dei comandi. Per riprovare, è necessario immettere nuovamente il`begin` comando per iniziare una nuova transazione.

```
transaction failed after 1 attempts, last error: communication failure:
 Transaction 2UMpiJ5hh7WLjVgEiMLOoO has expired
```

Transazioni non interattive

È possibile eseguire una transazione completa con più istruzioni raggruppando in batch comandi e istruzioni in sequenza come segue.

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

È necessario separare ogni comando e istruzione con un delimitatore da punto e virgola (`;`). Se una qualsiasi dichiarazione nella transazione non è valida, la shell rifiuta automaticamente la transazione. La shell non procede con le dichiarazioni successive che hai inserito.

Puoi anche impostare più transazioni.

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

Analogamente all'esempio precedente, se una transazione fallisce, la shell non procede con le transazioni o le dichiarazioni successive che hai inserito.

Se non si termina una transazione, la shell passa alla modalità interattiva e richiede il comando o l'istruzione successiva.

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

Uscire dal guscio

Per uscire dalla sessione corrente della`qldb` shell, inserisci il`quit` comando`exit` or o usa la scorciatoia da tastieraCtrl +D quando la shell non è in una transazione.

```
qldb> exit
$
```

```
qldb> quit
```

```
$
```

Esempio

Per informazioni sulla scrittura di istruzioni PartiQL in QLDB, vedere ilDocumentazione di riferimento Amazon QLDB PartiQL.

Example

Nell'esempio seguente viene illustrata una sequenza comune di comandi di base.

> **Note**
>
> La shell QLDB esegue ogni istruzione PartiQL in questo esempio nella propria transazione. Questo esempio presuppone che il libro mastro esista`test-ledger` già e sia attivo.

```
$ 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
```

Accesso ad Amazon QLDB tramite l'API

Puoi usare AWS Management Console and the AWS Command Line Interface (AWS CLI) per lavorare in modo interattivo con Amazon QLDB. Tuttavia, per ottenere il massimo da QLDB, puoi scrivere codice applicativo con un driver QLDB o AWS un SDK per interagire con il tuo libro mastro utilizzando le API.

Il driver consente all'applicazione di interagire con QLDB utilizzando l'API dei dati transazionali. L' AWS SDK supporta l'interazione con l'API di gestione delle risorse QLDB. Per ulteriori informazioni su queste API, consulta. Documentazione di riferimento dell'API Amazon QLDB

Il driver fornisce il supporto per QLDB in Java, .NET, Go , Node.js e Python. Per iniziare a utilizzare rapidamente questi linguaggi, vedi Nozioni base sul driver Amazon QLDB.

Prima di poter utilizzare un driver QLDB o AWS un SDK nell'applicazione, è necessario concedere l'accesso programmatico. Per ulteriori informazioni, consulta Concessione dell'accesso programmatico.

Nozioni di base sulla console Amazon QLDB

Questo tutorial illustra i passaggi per creare il tuo primo registro Amazon QLDB e compilarlo con tabelle e dati di esempio. Il registro di esempio creato in questo scenario è un database per un'applicazione del dipartimento dei veicoli a motore (DMV) che tiene traccia delle informazioni storiche complete sulle immatricolazioni dei veicoli.

La cronologia di un asset è un caso d'uso comune per QLDB perché implica una varietà di scenari e operazioni che evidenziano l'utilità di un database contabile. Con QLDB puoi accedere, interrogare e verificare direttamente la cronologia completa delle modifiche ai tuoi dati in un database orientato ai documenti che supporta funzionalità di interrogazione simili a SQL.

Mentre esegui questo tutorial, i seguenti argomenti spiegano come aggiungere le immatricolazioni dei veicoli, modificarle e visualizzare la cronologia delle modifiche a tali registrazioni. Questa guida mostra anche come verificare crittograficamente un documento di registrazione e si conclude ripulendo le risorse ed eliminando il registro di esempio.

Ogni passaggio del tutorial contiene istruzioni per l'utilizzo diAWS Management Console.

Argomenti

Prerequisiti e considerazioni

Prima di iniziare questo tutorial su Amazon QLDB, è necessario soddisfare i seguenti prerequisiti:

1. Segui le istruzioni diAWS configurazione inAccesso ad Amazon QLDB, se non l'hai già fatto. Questi passaggi includono la registrazioneAWS e la creazione di un utente amministrativo.

2. Segui le istruzioni inImpostazione delle autorizzazioni per impostare le autorizzazioni IAM per le tue risorse QLDB. Per completare tutti i passaggi di questo tutorial, è necessario l'accesso amministrativo completo alle risorse del libro mastro tramite ilAWS Management Console.

   > **Note**
   >
   > Se hai già effettuato l'accesso come utente con autorizzazioni diAWS gestione complete, puoi saltare questo passaggio.

3. (Facoltativo) QLDB crittografa i dati inattivi utilizzando una chiave inAWS Key Management Service (AWS KMS). Puoi scegliere uno dei seguenti tipi diAWS KMS keys:

   - AWSchiave KMS di proprietà: utilizza una chiave KMS di proprietà e gestita da per tuoAWS conto. Questa è l'opzione predefinita e non richiede alcuna configurazione aggiuntiva.
   - Chiave KMS gestita dal cliente: usa una chiave KMS KMS di crittografia nel tuo account che crei, possiedi e gestisci. QLDB non supporta chiavi asimmetriche.

     Questa opzione richiede la creazione di una chiave KMS o l'utilizzo di una chiave esistente nel tuo account. Per istruzioni sulla creazione di una chiave gestita dal cliente, consulta Creazione di chiavi KMS con crittografia simmetrica nella Guida per gliAWS Key Management Service sviluppatori.

     Puoi specificare una chiave KMS gestita dal cliente utilizzando un ID, un alias o un ARN (ARN). Per ulteriori informazioni, consulta Identificatori chiave (KeyId) nella Guida per gliAWS Key Management Service sviluppatori.

     > **Note**
     >
     > Le chiavi tra regioni non sono supportate. La chiave KMS specificata deve trovarsi nelloRegione AWS stesso libro mastro.

Impostazione delle autorizzazioni

In questo passaggio, configuri le autorizzazioni di accesso complete tramite la console per tutte le risorse QLDB del tuoAccount AWS. Per concedere rapidamente queste autorizzazioni, utilizza la politicaAWS gestita AmazonQLDBConsoleFullAccess.

Per fornire l'accesso, aggiungi autorizzazioni ai tuoi utenti, gruppi o ruoli:

- Utenti e gruppi in AWS IAM Identity Center:

  Crea un set di autorizzazioni. Segui le istruzioni riportate nella pagina Create a permission set (Creazione di un set di autorizzazioni) nella Guida per l'utente di AWS IAM Identity Center.

- Utenti gestiti in IAM tramite un provider di identità:

  Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate nella pagina Creating a role for a third-party identity provider (federation) (Creazione di un ruolo per un provider di identità di terze parti [federazione]) nella Guida per l'utente di IAM.

- Utenti IAM:
  - Crea un ruolo che l'utente possa assumere. Per istruzioni, consulta la pagina Creating a role for an IAM user (Creazione di un ruolo per un utente IAM) nella Guida per l'utente di IAM.
  - (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate nella pagina Aggiunta di autorizzazioni a un utente (console) nella Guida per l'utente di IAM.

> **Important**
>
> Ai fini di questo tutorial, ti concedi l'accesso amministrativo completo a tutte le risorse QLDB. Per i casi d'uso di produzione, tuttavia, segui le best practice di sicurezza di concedere il privilegio minimo o sulla concessione delle sole autorizzazioni richieste per eseguire un'attività. Per alcuni esempi, consulta Esempi di policy basate sull'identità per Amazon QLDB.

Per creare un libro contabile denominato`vehicle-registration`, procedere aFase 1: creazione di un nuovo libro mastro.

Fase 1: creazione di un nuovo libro mastro

In questo passaggio, crei un nuovo registro Amazon QLDB denominato`vehicle-registration`. Quindi, si conferma che lo stato del libro contabile è Attivo. Puoi anche verificare tutti i tag che hai aggiunto al libro mastro.

Quando si crea un libro mastro, la protezione dall'eliminazione è abilitata per impostazione predefinita. La protezione dall'eliminazione è una funzionalità di QLDB che impedisce a un utente

qualsiasi di eliminare i libri mastro. È possibile disabilitare la protezione dall'eliminazione quando si crea un libro contabile utilizzando l'API QLDB o ilAWS Command Line Interface (AWS CLI).

Per creare un nuovo libro mastro

1. Accedere allaAWS Management Console e aprire la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel riquadro di navigazione, selezionare Nozioni di base.
3. Nella scheda Crea il tuo primo libro contabile, scegli Crea libro contabile.
4. Nella pagina Create Ledger (Crea libro mastro), procedere come segue:

   - Informazioni sul libro contabile: il nome del libro contabile deve essere precompilato con**`vehicle-registration`**.
   - Modalità autorizzazioni: la modalità di autorizzazione da assegnare al libro mastro. Seleziona una delle seguenti opzioni:
     - Consenti tutto: una modalità di autorizzazione legacy che consente il controllo degli accessi con granularità a livello di API per i libri mastri.

       Questa modalità consente agli utenti che dispongono dell'autorizzazione API `SendCommand` per questo libro mastro per eseguire tutti i comandi PartiQL (quindi, `ALLOW_ALL`) su qualsiasi tabella nel libro mastro specificato. Questa modalità ignora tutte le policy di autorizzazione IAM a livello di tabella o di comando create per il libro mastro.
     - Standard: (impostazione consigliata) una modalità di autorizzazione che consente il controllo degli accessi con una granularità più fine per libri mastri mastri, tabelle e comandi PartiQL. Ti consigliamo di utilizzare questa modalità di autorizzazione per incrementare la sicurezza dei dati nel libro mastro.

       Per impostazione predefinita, questa modalità nega tutte le richieste di esecuzione di qualsiasi comando PartiQL su qualsiasi tabella in questo libro mastro. Per consentire i comandi PartiQL, è necessario creare le policy di autorizzazione IAM per risorse di tabelle e azioni PartiQL specifiche, oltre all'autorizzazione`SendCommand` API per il libro mastro. Per informazioni, consulta Guida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.
   - Crittografia dei dati a riposo: la chiave inAWS Key Management Service (AWS KMS) da usare per la crittografia dei dati a riposo. Seleziona una delle seguenti opzioni:

- Usa una chiave KMS diAWS proprietà: usa una chiave KMS di proprietà e gestita da per tuoAWS conto. Questa è l'opzione predefinita e non richiede alcuna configurazione aggiuntiva.
- Scegli unaAWS KMS chiave diversa: usa una chiave KMS di crittografia simmetrica nel tuo account che crei, possiedi e gestisci.

  Per creare una nuova chiave utilizzando laAWS KMS console, scegli Crea unaAWS KMS chiave. Per ulteriori informazioni, consulta Creazione di chiavi KMS di crittografia simmetrica nella Guida per gli sviluppatori di AWS Key Management Service.

  Per utilizzare una chiave KMS esistente, scegline una dall'elenco a discesa o specifica una chiave KMS ARN.

- Tag: (Facoltativo) Aggiungere metadati al libro mastro collegando i tag come coppie chiave-valore. Puoi aggiungere tag al libro mastro per facilitarne l'organizzazione e l'individuazione. Per ulteriori informazioni, consulta Assegnazione di tag alle risorse Amazon QLDB.

  Scegli Aggiungi tag, quindi inserisci le coppie chiave-valore appropriate.

5. Dopo aver selezionato tutte le impostazioni desiderate, scegliere Create (Crea libro mastro).

> **Note**
>
> Puoi accedere al tuo registro QLDB quando il suo stato diventa Attivo. Questo processo può richiedere diversi minuti.

6. Nell'elenco dei libri contabili, individua`vehicle-registration` e conferma che lo stato del libro contabile sia Attivo.
7. (Facoltativo) Scegliete il nome del`vehicle-registration` libro contabile. Nella pagina dei dettagli del registro di immatricolazione del veicolo, conferma che tutti i tag che hai aggiunto al libro contabile vengano visualizzati nella scheda Etichette. Puoi anche modificare i tag contabili utilizzando questa pagina della console.

Per creare tabelle nel`vehicle-registration` registro, procedere aFase 2: creare tabelle, indici e dati di esempio in un libro mastro.

Fase 2: creare tabelle, indici e dati di esempio in un libro mastro

Quando il tuo registro Amazon QLDB è attivo, puoi iniziare a creare tabelle con i dati sui veicoli, i loro proprietari e le loro informazioni di registrazione. Dopo aver creato le tabelle e gli indici, è possibile caricarli con i dati.

In questa fase, verranno create quattro tabelle nel`vehicle-registration` libro mastro:

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

È inoltre possibile creare i seguenti indici.

Nome tabella	Campo
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicensePlateNumber`
`DriversLicense`	`PersonId`

È possibile utilizzare la console QLDB per creare automaticamente queste tabelle con indici e caricarle con dati di esempio. In alternativa, è possibile utilizzare l'editor PartiQL sulla console per eseguire manualmente ogni istruzione PartiQL step-by-step.

Opzione automatica

Per creare tabelle, indici e dati di esempio

1. Apri la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.

2. Nel riquadro di navigazione, selezionare Nozioni di base.
3. Nell'opzione Automatico nella scheda dati dell'applicazione di esempio, sceglietevehicle-registration nell'elenco dei libri contabili.
4. Scegli Carica dati di esempio.

   Se l'operazione viene completata correttamente, la console visualizza il messaggio Dati di esempio caricati.

   Questo script esegue tutte le istruzioni in un'unica transazione. Se una parte della transazione fallisce, ogni dichiarazione viene ripristinata e viene visualizzato un messaggio di errore appropriato. È possibile riprovare l'operazione dopo aver risolto eventuali problemi.

   > **Note**
   >
   > - Una possibile causa di un errore di transazione è il tentativo di creare tabelle duplicate. La richiesta di caricamento dei dati di esempio avrà esito negativo se nel registro è già presente uno dei seguenti nomi di tabella:VehicleRegistrationVehicle,Person, eDriversLicense.
   >
   >   Prova invece a caricare questi dati di esempio in un registro vuoto.
   > - Questo script eseguеINSERT istruzioni parametrizzate. Quindi, queste istruzioni PartiQL vengono registrate nei blocchi del diario con parametri di associazione anziché con i dati letterali. Ad esempio, potresti vedere la seguente dichiarazione in un blocco di diario, in cui il punto interrogativo (?) è un segnaposto variabile per il contenuto del documento.
   >
   >   ```
   >   INSERT INTO Vehicle ?
   >   ```

Opzione manuale

I documenti vengono inseritiVehicleRegistration con unPrimaryOwner campo vuoto eDriversLicense con unPersonId campo vuoto. Successivamente, compili questi campi con il documento assegnato dal sistemaid dallaPerson tabella.

> **Tip**
>
> Come procedura consigliata, utilizza questo campo di`id` metadati del documento come chiave esterna. Per ulteriori informazioni, consulta Interrogazione dei metadati dei documenti.

Per creare tabelle, indici e dati di esempio

1. Apri la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel riquadro di navigazione, selezionare l'editor PartiQL.
3. Scegli il`vehicle-registration` libro mastro.
4. Inizia creando quattro tabelle. QLDB supporta contenuti aperti e non applica schemi, quindi non si specificano attributi o tipi di dati.

   Nella finestra dell'editor di query, immettete la seguente istruzione, quindi scegliete Esegui. Per eseguire l'istruzione, puoi anche usare la scorciatoia da tastieraCtrl +Enter per Windows oCmd +Return per macOS. Per altre scelte rapide da tastiera, consultaScelte rapide da tastiera dell'editor PartiQL.

   ```
   CREATE TABLE VehicleRegistration
   ```

   Ripetere questa fase per eseguire tutte le operazioni seguenti.

   ```
   CREATE TABLE Vehicle
   ```

   ```
   CREATE TABLE Person
   ```

   ```
   CREATE TABLE DriversLicense
   ```

5. Quindi, crea indici che ottimizzino le prestazioni delle query per ogni tabella.

   > **Important**
   >
   > QLDB richiede un indice per cercare in modo efficiente un documento. Senza un indice, QLDB deve eseguire una scansione completa della tabella durante la lettura dei documenti. Ciò può causare problemi di prestazioni su tabelle di grandi dimensioni, inclusi conflitti di concorrenza e timeout delle transazioni.

> Per evitare la scansione delle tabelle, è necessario eseguire istruzioni con una clausolaWHERE predicativa utilizzando un operatore di uguaglianza (=oIN) su un campo indicizzato o un ID di documento. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.

Nella finestra dell'editor di query, immettete la seguente istruzione, quindi scegliete Esegui.

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

Ripetere questa fase per eseguire le operazioni seguenti.

```
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. Dopo aver creato gli indici, puoi iniziare a caricare i dati nelle tabelle. In questo passaggio, inserisci i documenti nellaPerson tabella con le informazioni personali sui proprietari dei veicoli monitorati dal registro.

   Nella finestra dell'editor di query, immettete la seguente istruzione, quindi scegliete Esegui.

```
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. Quindi, compila la`DriversLicense` tabella con i documenti che includono le informazioni sulla patente di guida per ogni proprietario del veicolo.

   Nella finestra dell'editor di query, immettete la seguente istruzione, quindi scegliete Esegui.

```
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. Ora compila la`VehicleRegistration` tabella con i documenti di immatricolazione del veicolo. Questi documenti includono una`Owners` struttura annidata che memorizza i proprietari primari e secondari.

   Nella finestra dell'editor di query, immettete la seguente istruzione, quindi scegliete Esegui.

```
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. Infine, compila la`Vehicle` tabella con i documenti che descrivono i veicoli registrati nel tuo registro.

   Nella finestra dell'editor di query, immettete la seguente istruzione, quindi scegliete Esegui.

```
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'
} >>
```

Successivamente, è possibile utilizzare`SELECT` le dichiarazioni per leggere i dati dalle tabelle del`vehicle-registration` libro mastro. Continua con la Fase 3: esecuzione di query sulle tabelle in un libro mastro.

Fase 3: esecuzione di query sulle tabelle in un libro mastro

Dopo aver creato le tabelle in un registro Amazon QLDB e averle caricate con i dati, puoi eseguire interrogazioni per esaminare i dati di immatricolazione del veicolo che hai appena inserito. QLDB utilizza PartiQL come linguaggio di interrogazione e Amazon Ion come modello di dati orientato ai documenti.

PartiQL è un linguaggio di interrogazione open source e compatibile con SQL che è stato esteso per funzionare con Ion. Con PartiQL, puoi inserire, interrogare e gestire i tuoi dati con operatori SQL familiari. Amazon Ion è un superset di JSON. Ion è un formato di dati open source basato su documenti che offre la flessibilità di archiviare ed elaborare dati strutturati, semistrutturati e annidati.

In questo passaggio, si utilizzano i`SELECT` rendiconti per leggere i dati dalle tabelle del`vehicle-registration` libro mastro.

> **Warning**
>
> Quando si esegue una query in QLDB senza una ricerca indicizzata, viene richiamata una scansione completa della tabella. PartiQL supporta tali interrogazioni perché è compatibile con SQL. Tuttavia, non eseguite scansioni di tabelle per casi d'uso di produzione in QLDB. Le scansioni delle tabelle possono causare problemi di prestazioni su tabelle di grandi dimensioni, inclusi conflitti di concorrenza e timeout delle transazioni.
> Per evitare la scansione delle tabelle, è necessario eseguire istruzioni con una clausola`WHERE` predicativa utilizzando un operatore di uguaglianza su un campo indicizzato o un ID di documento; ad esempio,`WHERE indexedField = 123` o`WHERE indexedField IN (456, 789)`. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.

Per interrogare le tabelle

1. Apri la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel riquadro di navigazione, selezionare l'editor PartiQL.
3. Scegli il`vehicle-registration` libro mastro.
4. Nella finestra dell'editor delle interrogazioni, inserisci la seguente istruzione per interrogare la`Vehicle` tabella relativa a un particolare numero di identificazione del veicolo (VIN) che hai aggiunto al libro contabile, quindi scegli Esegui.

   Per eseguire l'istruzione, puoi anche usare la scorciatoia da tastieraCtrl +Enter per Windows oCmd +Return per macOS. Per altre scelte rapide da tastiera, consultaScelte rapide da tastiera dell'editor PartiQL.

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

5. Puoi scrivere interrogazioni interne. Questo esempio di interrogazione unisce`Vehicle`a`VehicleRegistration` e restituisce le informazioni di registrazione insieme agli attributi del veicolo registrato per un determinato veicolo`VIN`.

   Immettete la seguente istruzione, quindi scegliete Esegui.

   ```
   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
```

È inoltre possibile unire le`DriversLicense` tabelle`Person` e per visualizzare gli attributi relativi ai driver che sono stati aggiunti al libro mastro.

Ripetere questa fase per eseguire le operazioni seguenti.

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

Per informazioni sulla modifica dei documenti nelle tabelle del`vehicle-registration` libro mastro, vedereFase 4: modificare i documenti in un libro mastro.

Fase 4: modificare i documenti in un libro mastro

Ora che hai dati su cui lavorare, puoi iniziare a modificare i documenti nel`vehicle-registration` registro in Amazon QLDB. Prendiamo ad esempio l'Audi A5 con VIN`1N4AL11D75C109151`. Questa vettura è inizialmente di proprietà di un autista di nome Raul Lewis a Seattle, WA.

Supponiamo che Raul venda l'auto a un residente di Everett, WA di nome Brent Logan. Poi, Brent e Alexis Pena decidono di sposarsi. Brent vuole aggiungere Alexis come proprietario secondario alla registrazione. In questo passaggio, le seguenti istruzioni del linguaggio di manipolazione dei dati (DML) mostrano come apportare le modifiche appropriate nel registro per riflettere questi eventi.

> **Tip**
>
> Come procedura consigliata, utilizzare`id` come chiave esterna assegnata dal sistema di un documento. Sebbene sia possibile definire campi destinati a essere identificatori univoci (ad esempio, il VIN di un veicolo), il vero identificatore univoco di un documento è il suo`id`. Questo campo è incluso nei metadati del documento, che è possibile interrogare nella vista commit (la vista definita dal sistema di una tabella).
> Per ulteriori informazioni sulle viste in QLDB, consultaConcetti principali. Per ulteriori informazioni sui metadati, consultaInterrogazione dei metadati dei documenti.

Per modificare i documenti

1. Apri la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel riquadro di navigazione, selezionare l'editor PartiQL.
3. Scegli il`vehicle-registration` libro mastro.

> **Note**
>
> Se configuri il tuo libro contabile utilizzando la funzione Carica dati di esempio automatica della console, vai direttamente al passaggio 6.

4. Se hai eseguito manualmente`INSERT` le istruzioni per caricare i dati di esempio, continua con questi passaggi.

   Per registrare inizialmente Raul come proprietario di questo veicolo, inizia trovando il documento assegnato dal sistema`id` nella`Person` tabella. Questo campo è incluso nei metadati del documento, che possono essere interrogati nella vista definita dal sistema della tabella, denominata vista confermata.

   Nella finestra dell'editor di query, immettete la seguente istruzione, quindi scegliete Esegui.

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

   Il prefisso`_ql_committed_` è un prefisso riservato che indica che si desidera interrogare la vista confermata della`Person` tabella. In questa visualizzazione, i dati sono annidati nel`data` campo e i metadati sono annidati nel`metadata` campo.

5. Ora, usate`loid` in una`UPDATE` dichiarazione per modificare il documento appropriato nella`VehicleRegistration` tabella. Immettete la seguente istruzione, quindi scegliete Esegui.

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

   Conferma di aver modificato il`Owners` campo emettendo questa dichiarazione.

   ```
   SELECT r.Owners FROM VehicleRegistration AS r
   ```

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

6. Per trasferire la proprietà del veicolo a Brent nella città di Everett, trova innanzitutto la suaid dallaPerson tabella con la seguente dichiarazione.

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

Quindi, usaloid per aggiornarePrimaryOwner eCity nellaVehicleRegistration tabella.

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

Conferma di aver modificato iCity campiPrimaryOwner e emettendo questa dichiarazione.

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

7. Per aggiungere Alexis come proprietario secondario dell'auto, trovalaPerson id.

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

Quindi, inserisciloid nell'SecondaryOwnerselenco con la seguente istruzione DML FROM-INSERT.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners
    VALUE { 'PersonId' : '5Ufgdlnj06gF5CWcOIu64s' } --replace with your id
```

Conferma di aver modificatoSecondaryOwners emettendo questa dichiarazione.

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

Per esaminare queste modifiche nelvehicle-registration registro, vedereFase 5: visualizzazione della cronologia delle revisioni di un documento.

Fase 5: visualizzazione della cronologia delle revisioni di un documento

Dopo aver modificato i dati di registrazione dell'auto con VIN 1N4AL11D75C109151, è possibile interrogare la cronologia di tutti i proprietari registrati e qualsiasi altro campo aggiornato. Puoi vedere tutte le revisioni di un documento che hai inserito, aggiornato ed eliminato interrogando la funzionalità integrataFunzione di cronologia.

La funzione di cronologia restituisce le revisioni dalla vista salvata della tabella, che include sia i dati dell'applicazione che i metadati associati. I metadati mostrano esattamente quando è stata effettuata ogni revisione, in quale ordine e quale transazione è stata effettuata.

In questo passaggio, si interroga la cronologia delle revisioni di un documento nellaVehicleRegistration tabella delvehicle-registration libro mastro.

Per visualizzare la cronologia delle revisioni

1. Apri la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel riquadro di navigazione, selezionare l'editor PartiQL.
3. Scegli ilvehicle-registration libro mastro.
4. Per interrogare la cronologia di un documento, inizia trovandone l'univocoid. Oltre a interrogare la vista confermata, un altro modo per ottenere un documentoid consiste nell'utilizzare laBY parola chiave nella visualizzazione utente predefinita della tabella. Per ulteriori informazioni, consulta Utilizzo della clausola BY per interrogare l'ID del documento.

   Nella finestra dell'editor di query, immettete la seguente istruzione, quindi scegliete Esegui.

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

5. Successivamente, è possibile utilizzare questoid valore per interrogare la funzione di cronologia. Immettete la seguente istruzione, quindi scegliete Esegui. Sostituire ilid valore con l'ID documento, a seconda delle esigenze.

   ```
   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**
>
> Ai fini di questo tutorial, questa query cronologica restituisce tutte le revisioni dell'ID del documentoADR2LQq48kB9neZDupQrMm. Come procedura consigliata, tuttavia, qualifica un'interrogazione cronologica con un ID del documento e un intervallo di date (ora di inizio e ora di fine).
>
> In QLDB, ogniSELECT query viene elaborata in una transazione ed è soggetta a un limite di timeout della transazione. Le interrogazioni cronologiche che includono un'ora di inizio e un'ora di fine ottengono il vantaggio della qualificazione dell'intervallo di date. Per ulteriori informazioni, consulta Funzione di cronologia.

La funzione di cronologia restituisce documenti nello stesso schema della visualizzazione. Questo esempio proietta i dati di immatricolazione del veicolo modificati. L'output visualizzato dovrebbe essere simile al seguente:

VIN	Città	Proprietari
"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**
>
> L'interrogazione della cronologia potrebbe non restituire sempre le revisioni dei documenti in ordine sequenziale.

Rivedi l'output e conferma che le modifiche riflettano ciò che hai fattoFase 4: modificare i documenti in un libro mastro.

6. Quindi, puoi controllare i metadati del documento di ogni revisione. Immettete la seguente istruzione, quindi scegliete Esegui. Ancora una volta, assicurati di sostituire il`id` valore con l'ID del tuo documento, a seconda dei casi.

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

L'output visualizzato dovrebbe essere simile al seguente:

versio	id	Ora TX	ID 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"

Questi campi di metadati forniscono dettagli su quando ogni articolo è stato modificato e con quale transazione. Da questi dati, puoi dedurre quanto segue:

- Il documento è identificato in modo univoco dal sistema assegnato`id`:`ADR2LQq48kB9neZDupQrMm`. Si tratta di un identificatore univoco universale (UUID) rappresentato in una stringa codificata in Base62.
- `txTime`Mostra che la revisione iniziale del documento (versione`0`) è stata creata in`2019-05-23T19:20:360d-3Z`.
- Ogni transazione successiva crea una nuova revisione con lo stesso documento`id`, un numero di versione incrementato e un`txId` e aggiornato`txTime`.

Per verificare crittograficamente la revisione di un documento nel`vehicle-registration` libro mastro, procedere aFase 6: Verificare un documento in un libro mastro.

Fase 6: Verificare un documento in un libro mastro

Con Amazon QLDB, puoi verificare in modo efficiente l'integrità di un documento nel giornale contabile utilizzando l'hashing crittografico con SHA-256. In questo esempio, Alexis e Brent decidono di passare a un nuovo modello scambiando il veicolo con VIN`1N4AL11D75C109151` presso un concessionario di automobili. La concessionaria avvia il processo verificando la proprietà del veicolo presso l'ufficio di immatricolazione.

Per ulteriori informazioni su come funzionano la verifica e l'hashing crittografico in QLDB, consultaVerifica dei dati in Amazon QLDB.

In questo passaggio, si verifica una revisione del documento nel`vehicle-registration` registro. Innanzitutto, richiedi un riassunto, che viene restituito come file di output e funge da firma dell'intera cronologia delle modifiche del tuo libro contabile. Quindi, richiedi una prova della revisione relativa a quel riassunto. Utilizzando questa prova, l'integrità della revisione viene verificata se tutti i controlli di convalida vengono superati.

Per richiedere un riassunto

1. Apri la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel riquadro di navigazione, selezionare Ledgers (libro mastro).
3. Nell'elenco dei libri contabili, seleziona`vehicle-registration`.
4. Scegli Get digest. La finestra di dialogo Ottieni riassunto mostra i seguenti dettagli del riassunto:

   - Digest: il valore hash SHA-256 del digest richiesto.

- Indirizzo suggerito per il riassunto: l'ultima posizione del blocco nel diario a cui si riferisce il riassunto richiesto. Un indirizzo ha i seguenti due campi:
  - `strandId`— L'ID univoco del filone del journal che contiene il blocco.
  - `sequenceNo`— Il numero di indice che specifica la posizione del blocco all'interno del filamento.
- Libro contabile: il nome del libro contabile per il quale è stato richiesto un riassunto.
- Data: il timestamp in cui hai richiesto il digest.

5. Verificare le informazioni del riassunto. Quindi scegli Save (Salva). Puoi mantenere il nome del file predefinito o immettere un nuovo nome del file.

   Questo passaggio salva un file di testo in chiaro con contenuti in formato Amazon Ion. Il file ha l'estensione del nome di file `.ion.txt` e contiene tutte le informazioni del riassunto elencate nella finestra di dialogo precedente. Di seguito è riportato un esempio del contenuto di un file digest. L'ordine dei campi può variare a seconda del browser.

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

6. Salva questo file dove potrai accedervi in seguito. Nei passaggi seguenti, si utilizza questo file per verificare la revisione di un documento.

Dopo aver salvato un riassunto contabile, puoi iniziare il processo di verifica della revisione di un documento rispetto a quel riepilogo.

> **Note**
>
> In un caso d'uso di produzione per la verifica, si utilizza un riassunto precedentemente salvato anziché eseguire le due attività consecutivamente. Come buona prassi, richiedi e salva il riassunto non appena una revisione che desideri verificare in seguito viene scritta nel giornale.

Per verificare la revisione di un documento

1. Per prima cosa, interroga il`id` libro mastro per la fine`blockAddress` della revisione del documento che desideri verificare. Questi campi sono inclusi nei metadati del documento, che puoi interrogare nella vista confermata.

   Il documento`id` è una stringa ID univoca assegnata dal sistema. `blockAddress`È una struttura a ioni che specifica la posizione del blocco in cui è stata eseguita la revisione.

   Nel riquadro di navigazione della console QLDB, selezionare l'editor PartiQL (PartiQL).
2. Scegli il`vehicle-registration` libro mastro.
3. Nella finestra dell'editor di query, immettete la seguente istruzione, quindi scegliete Esegui.

   ```
   SELECT r.metadata.id, r.blockAddress
   FROM _ql_committed_VehicleRegistration AS r
   WHERE r.data.VIN = '1N4AL11D75C109151'
   ```
4. Copia e salva i`blockAddress` valori`id` e restituiti dall'interrogazione. Assicurati di omettere le virgolette doppie per il`id` campo. In Amazon Ion, i tipi di dati di tipo stringa sono delimitati da virgolette doppie.
5. Ora che hai selezionato una revisione del documento, puoi iniziare il processo di verifica.

   Nel riquadro di navigazione, selezionare Verificare.
6. Nel modulo Verifica documento, in Specifica il documento che desideri verificare, inserisci i seguenti parametri di input:

   - Ledger: sceglil`vehicle-registration`.
   - Indirizzo di blocco: il`blockAddress` valore restituito dall'interrogazione nel passaggio 3.
   - ID documento: il`id` valore restituito dall'interrogazione nel passaggio 3.
7. In Specifica il riassunto da utilizzare per la verifica, seleziona il riassunto che hai salvato in precedenza scegliendo Scegli riassunto. Se il file è valido, compila automaticamente tutti i campi del riepilogo sulla tua console. In alternativa, puoi copiare e incollare manualmente i seguenti valori direttamente dal tuo file di digest:

   - Digest: il`digest` valore del file di riepilogo.
   - Indirizzo del suggerimento del digest: il`digestTipAddress` valore del file del digest.
8. Controlla il documento e riepiloga i parametri di input, quindi scegli Verifica.

La console automatizza due passaggi per te:

a. Richiedi una prova a QLDB per il documento specificato.

b. Usa la bozza restituita da QLDB per chiamare un'API lato client, che verifica la revisione del documento rispetto al digest fornito.

La console visualizza i risultati della richiesta nella scheda dei risultati della verifica. Per ulteriori informazioni, consulta Risultati della verifica.

9. Per testare la logica di verifica, ripeti i passaggi da 6 a 8 in Per verificare una revisione del documento, ma modifica un singolo carattere nella stringa di input Digest. Ciò dovrebbe far sì che la tua richiesta di verifica fallisca con un messaggio di errore appropriato.

Se non è più necessario utilizzare il`vehicle-registration` libro mastro, procedere aFase 7 (opzionale): eliminazione delle risorse.

Fase 7 (opzionale): eliminazione delle risorse

Puoi continuare a usare il`vehicle-registration` libro mastro. Tuttavia, se non è più necessario, è necessario eliminarlo.

Se la protezione dall'eliminazione è abilitata per il libro mastro, è necessario innanzitutto disabilitarla prima di poter eliminare la contabilità utilizzando l'API QLDB,AWS Command Line Interface (AWS CLI) o la console QLDB.

Per eliminare il libro mastro

1. Apri la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel riquadro di navigazione, selezionare Ledgers (libro mastro).
3. Se la protezione dall'eliminazione è abilitata per questo libro mastro, è necessario innanzitutto disabilitarlo.

   Nell'elenco dei libri contabili`vehicle-registration`, selezionare e quindi scegliere Modifica libro contabile.
4. Nella pagina Modifica libro contabile, disattiva la protezione da eliminazione e quindi scegli Conferma modifiche.

5. Nell'elenco dei libri contabili, selezionare`vehicle-registration` nuovamente, quindi scegliere Elimina.
6. Confermalo inserendo**`delete vehicle-registration`** nel campo fornito.

Per ulteriori informazioni sull'utilizzo di libri mastro in QLDB, consultaNozioni di base su Amazon QLDB: fasi successive.

Nozioni di base su Amazon QLDB: fasi successive

Per ulteriori informazioni sull'utilizzo di Amazon QLDB, consulta i seguenti argomenti:

- Utilizzo di dati e cronologia in Amazon QLDB
- Nozioni base sul driver Amazon QLDB
- Modello di concorrenza Amazon QLDB
- Esportazione dei dati del diario da Amazon QLDB
- Streaming dei dati del diario da Amazon QLDB
- Verifica dei dati in Amazon QLDB
- Documentazione di riferimento Amazon QLDB PartiQL

Nozioni base sul driver Amazon QLDB

Questo capitolo contiene tutorial pratici per aiutarti a saperne di più sullo sviluppo con Amazon QLDB usando il driver QLDB. Il driver è basato sull'AWSSDK, che supporta l'interazione con l'API QLDB.

Astrazione della sessione QLDB

Il driver fornisce un livello di astrazione di alto livello sopra l'API dei dati transazionali (sessione QLDB). Semplifica il processo di esecuzione delle dichiarazioni PartiQL sui dati contabili gestendo le chiamate SendCommandAPI. Queste chiamate API richiedono diversi parametri che il driver gestisce per te, inclusa la gestione delle sessioni, delle transazioni e la politica dei tentativi in caso di errori. Il driver dispone anche di ottimizzazioni delle prestazioni e applica le migliori pratiche per interagire con QLDB.

> **Note**
>
> Per interagire con le operazioni API di gestione delle risorse elencate nel riferimento all'API Amazon QLDB, si utilizza direttamente l'AWSSDK anziché il driver. L'API di gestione viene utilizzata solo per la gestione delle risorse contabili e per operazioni sui dati non transazionali, come l'esportazione, lo streaming e la verifica dei dati.

Supporto Amazon Ion

Inoltre, il driver utilizza le librerie Amazon Ion per fornire supporto per la gestione dei dati Ion durante l'esecuzione delle transazioni. Queste librerie si occupano anche del calcolo dell'hash dei valori Ion. QLDB richiede questi hash Ion per verificare l'integrità delle richieste di transazione di dati.

Terminologia relativa ai conducenti

Questo strumento è chiamato driver perché è paragonabile ad altri driver di database che forniscono interfacce intuitive per gli sviluppatori. Allo stesso modo, questi driver incapsulano una logica che converte un set standard di comandi e funzioni in chiamate specifiche richieste dall'API di basso livello del servizio.

Il driver è open source GitHub ed è disponibile per i seguenti linguaggi di programmazione:

- driver Java

- .NET driver
- Vai, autista
- Driver Node.js
- driver Python

Per informazioni generali sui driver per tutti i linguaggi di programmazione supportati e tutorial aggiuntivi, consulta i seguenti argomenti:

- Gestione delle sessioni con il conducente
- Consigli per i
- Policy per riprovare un driver
- Errori comuni
- Tutorial di un'applicazione di esempio
- Uso di Amazon Ion
- Ottenere le statistiche delle dichiarazioni PartiQL

Driver Amazon QLDB per Java

Per lavorare con i dati contenuti nel tuo registro, puoi connetterti ad Amazon QLDB dalla tua applicazione Java utilizzando un driverAWS fornito. I seguenti argomenti descrivono come iniziare a usare il driver QLDB per Java.

Argomenti

- Risorse per i conducenti
- Prerequisiti
- Impostazione delleAWS credenziali e della regione predefinite
- Installazione
- Driver Amazon QLDB per Java — Tutorial di avvio rapido
- Driver Amazon QLDB per Java — Riferimento al ricettario

Risorse per i conducenti

Per ulteriori informazioni sulle funzionalità supportate dal driver Java, vedere le risorse seguenti:

- Riferimento API: 2.x, 1.x
- Codice sorgente del driver (GitHub)
- Codice sorgente dell'applicazione di esempio (GitHub)
- Struttura Ledger Loader (GitHub)
- Esempi di codice Amazon Ion

Prerequisiti

Prima di iniziare a utilizzare il driver QLDB per Java, occorre:

1. Segui le istruzioniAWS di configurazione inAccesso ad Amazon QLDB. Questo include gli output seguenti:
    1. Per registrarti e ottenere un accountAWS.
    2. Crea un utente con le autorizzazioni QLDB appropriate.
    3. Concessione dell'accesso programmatico per lo sviluppo.
2. Configura un ambiente di sviluppo Java scaricando e installando quanto segue:
    1. Java SE Development Kit 8, ad esempio Amazon Corretto 8.
    2. (Facoltativo) Ambiente di sviluppo integrato Java (IDE) di tua scelta, come Eclipse o IntelliJ.
3. Configura subito il tuo ambienteAWS SDK for Java di sviluppoImpostazione delleAWS credenziali e della regione predefinite.

Successivamente, puoi scaricare l'applicazione di esempio del tutorial completo oppure puoi installare solo il driver in un progetto Java ed eseguire brevi esempi di codice.

- Per installare il driver QLDB e il driverAWS SDK for Java in un progetto esistente, procedere aInstallazione.
- Per configurare un progetto ed eseguire brevi esempi di codice che dimostrino le transazioni di dati di base su un libro mastro, consulta ilGuida di avvio rapido.
- Per eseguire esempi più approfonditi di operazioni relative ai dati e alle API di gestione nell'applicazione di esempio del tutorial completo, consulta ilTutorial su Java.

Impostazione delleAWS credenziali e della regione predefinite

Il driver QLDB e il sottostante driver AWS SDK for Javarichiedono che siano forniteAWS le credenziali all'applicazione durante il runtime. Gli esempi di codice in questa guida presuppongono che stia usando un file diAWS credenziali, come descritto in Configurazione delle credenziali e della regione predefinite nella Guida per gliAWS SDK for Java 2.x sviluppatori di.

Come parte di questi passaggi, è necessario impostare anche il valore predefinitoRegione AWS per determinare l'endpoint QLDB predefinito. Gli esempi di codice si connettono a QLDB come impostazione predefinitaRegione AWS. Per un elenco completo delle regioni in cui QLDB è disponibile, consulta gli endpoint e le quote di Amazon QLDB nella Riferimenti generali di AWS.

Di seguito, è riportato un esempio di un file di credenziali AWS, denominato `~/.aws/credentials`, dove il segno della tilde (~) rappresenta la tua directory home.

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

Sostituisci i valoriAWS delle tue credenziali con i valori *`your_access_key_id`* e *`your_secret_access_key`*.

Installazione

QLDB supporta le seguenti versioni dei driver Java e le relative dipendenzeAWS SDK.

Versione driver	AWS SDK	Stato	Data di rilascio più recente
1.x	AWS SDK for Java1.x	Versione di produzione	20 marzo 2020
2.x	AWS SDK for Java 2.x	Versione di produzione	4 giugno 2021

Per installare il driver QLDB, si consiglia di utilizzare un sistema di gestione delle dipendenze, come Gradle o Maven. Ad esempio, aggiungi il seguente artefatto come dipendenza nel proprio progetto Java.

2.x

Gradle

Aggiungi questa dipendenza nel tuo file`build.gradle` di configurazione.

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

Maven

Aggiungi questa dipendenza nel tuo file`pom.xml` di configurazione.

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

Questo artefatto include automaticamente il moduloAWS SDK for Java 2.x principale, le librerie Amazon Ion e altre dipendenze richieste.

1.x

Gradle

Aggiungi questa dipendenza nel tuo file`build.gradle` di configurazione.

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

Maven

Aggiungi questa dipendenza nel tuo file`pom.xml` di configurazione.

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

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

Questo artefatto include automaticamente il moduloAWS SDK for Java principale, le librerie Amazon Ion e altre dipendenze richieste.

> **Important**
>
> Spazio dei nomi Amazon Ion: quando si importano le classi Amazon Ion nell'applicazione, è necessario utilizzare il pacchetto che si trova sotto lo spazio dei nomi`com.amazon.ion`. AWS SDK for JavaDipende da un altro pacchetto Ion nel namespace`software.amazon.ion`, ma questo è un pacchetto legacy che non è compatibile con il driver QLDB.

Per brevi esempi di codice su come eseguire transazioni di dati di base su un libro mastro, vedere ilDocumentazione di riferimento del libro.

Altre librerie opzionali

Eventualmente, è anche possibile aggiungere le seguenti librerie utili al proprio progetto. Questi artefatti sono dipendenze obbligatorie nell'applicazioneTutorial su Java di esempio.

1. aws-java-sdk-qldb— Il modulo QLDB delAWS SDK for Java. La versione minima supportata da QLDB è`1.11.785`.

   Utilizza questo modulo nella tua applicazione per interagire direttamente con le operazioni dell'API di gestione elencate inDocumentazione di riferimento dell'API Amazon QLDB.
2. jackson-dataformat-ion— Modulo di formato dati Jackson di FasterXML per Ion. L'applicazione di esempio richiede una versione`2.10.0` o successiva.

Gradle

Aggiungi queste dipendenze nel tuo file`build.gradle` di configurazione.

```
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

Aggiungi queste dipendenze nel tuo filepom.xml di configurazione.

```
<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>
```

Driver Amazon QLDB per Java — Tutorial di avvio rapido

In questo tutorial, imparerai come configurare una semplice applicazione utilizzando la versione più recente del driver Amazon QLDB per Java. Questa guida include i passaggi per l'installazione del driver ed esempi di creazione, lettura, aggiornamento ed eliminazione (CRUD). Per esempi più approfonditi che dimostrano queste operazioni in un'applicazione di esempio completa, consulta ilTutorial su Java.

Argomenti

- Esecuzione dell'applicazione completa

Prerequisiti

Prima di iniziare, assicurati di completare quanto segue:

1. Completa il driverPrerequisiti for the Java, se non l'hai già fatto. Ciò include la registrazioneAWS, la concessione dell'accesso programmatico per lo sviluppo e l'installazione di un ambiente di sviluppo integrato (IDE) Java.
2. Crea un libro contabile denominato`quick-start`.

   Per informazioni su come creare un libro mastro, consultaOperazioni di base per i libri mastri Amazon QLDB oFase 1: creazione di un nuovo libro mastro in Guida introduttiva alla console.

Fase 1: Configurazione del progetto

Per prima cosa, configura il tuo progetto Java. Consigliamo di utilizzare il sistema di gestione delle dipendenze Maven per questo tutorial.

> **Note**
>
> Se utilizzi un IDE con funzionalità per automatizzare questi passaggi di configurazione, puoi passare direttamente aFase 2: Inizializzazione del driver.

1. Crea una cartella per la tua applicazione.

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

2. Inserisci il comando seguente per inizializzazione del progetto da un modello Maven. Sostituisci *`project-package`*, *`project-name`* e *`maven-template`* con i tuoi valori, a seconda dei casi.

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

Per *Maven-template*, puoi usare il modello Maven di base:`maven-archetype-quickstart`

3. Per aggiungere il driver QLDB per Java come dipendenza del progetto, vai al`pom.xml` file appena creato e aggiungi il seguente artefatto.

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

Questo artefatto include automaticamente il modulo AWS SDK for Java 2.xprincipale, le librerie Amazon Ion e altre dipendenze richieste. Al momento, il`pom.xml` file si presenta in maniera simile a quanto riportato di seguito.

```
<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. Apri il file `App.java`.

Quindi, aggiungi in modo incrementale gli esempi di codice nei passaggi seguenti per provare alcune operazioni CRUD di base. In alternativa, puoi saltare il step-by-step tutorial ed eseguire invece l'applicazione completa.

Fase 2: Inizializzazione del driver

Inizializza un'istanza del driver che si connette al registro denominato`quick-start`. Aggiungi il seguente codice al`App.java` file.

```
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();
    }
}
```

Fase 3: Crea una tabella e un indice

L'esempio di codice seguente mostra come eseguire`CREATE TABLE` le`CREATE INDEX` istruzioni.

Nel`main` metodo, aggiungi il codice seguente che crea una tabella denominata`People` e un indice per il`lastName` campo di quella tabella. Gli indici sono necessari per ottimizzare le prestazioni delle query e contribuire a limitare le eccezioni di conflitto OCC (Optimistic Concurrency Control).

```
// 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)");
});
```

Fase 4: Inserimento di un documento

L'esempio di codice seguente mostra come eseguire un'`INSERT`istruzione. QLDB supporta il linguaggio di interrogazione PartiQL (compatibile con SQL) e il formato dati Amazon Ion (superset di JSON).

Aggiungete il seguente codice che inserisce un documento nella`People` tabella.

```
// 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);
});
```

Questo esempio utilizza un punto interrogativo (?) come segnaposto variabile per passare le informazioni del documento alla dichiarazione. Quando si utilizzano segnaposti, è necessario passare un valore di tipo`IonValue`.

> **Tip**
>
> Per inserire più documenti utilizzando una singolaINSERT istruzione, è possibile passare un parametro di tipo IonList(esplicitamente espresso come un`IonValue`) all'istruzione come segue.
>
> ```
> // people is an IonList explicitly cast as an IonValue
> txn.execute("INSERT INTO People ?", (IonValue) people);
> ```
>
> Non si racchiude il segnaposto variabile (?) tra parentesi a doppio angolo (`<<...>>`) quando si passa un`IonList`. Nelle istruzioni PartiQL, le parentesi a doppio angolo indicano una raccolta non ordinata nota come borsa.

Fase 5: esegui query sul documento

L'esempio di codice seguente mostra come eseguire un'`SELECT`istruzione.

Aggiungete il codice seguente per interrogare un documento dalla`People` tabella.

```
// 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
});
```

Fase 6: Aggiornamento del documento

L'esempio di codice seguente mostra come eseguire un'`UPDATE`istruzione.

1. Aggiungi il codice seguente che aggiorna un documento nella`People` tabella eseguendo l'aggiornamento`age a 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. Interroga nuovamente il documento per visualizzare il valore aggiornato.

   ```
   // 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. Usa Maven o il tuo IDE per compilare ed eseguire il`App.java` file.

Esecuzione dell'applicazione completa

Il seguente esempio di codice è la versione completa dell'`App.java`applicazione. Invece di eseguire i passaggi precedenti singolarmente, puoi anche copiare ed eseguire questo esempio di codice dall'inizio alla fine. Questa applicazione dimostra alcune operazioni CRUD di base sul registro denominato`quick-start`.

> **Note**
>
> Prima di eseguire questo codice, assicuratevi di non avere già una tabella attiva denominata`People` nel`quick-start` libro mastro.
> Nella prima riga, sostituisci *`project-package`* con il`groupId` valore che hai usato per il comando Maven inFase 1: Configurazione del progetto .

```
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
        });
    }
}
```

Usa Maven o il tuo IDE per compilare ed eseguire il`App.java` file.

Driver Amazon QLDB per Java — Riferimento al ricettario

Questa guida di riferimento mostra i casi d'uso più comuni del driver Amazon QLDB per Java. Fornisce esempi di codice Java che illustra come eseguire operazioni di creazione, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, aggiornamento ed eliminazione (CRUD) Include anche esempi di codice per l'elaborazione dei dati di Amazon Ion. Inoltre, questa guida evidenzia le migliori pratiche per rendere le transazioni idempotenti e implementare i vincoli di unicità.

> **Note**
>
> Ove applicabile, alcuni casi d'uso hanno esempi di codice diversi per ciascuna versione principale supportata del driver QLDB per Java.

Indice

Importazione del driver

Il seguente esempio di codice importa il driver, il client di sessione QLDB, i pacchetti Amazon Ion e altre dipendenze correlate.

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;
```

Documentazione di creazione del driver

L'esempio di codice seguente crea un'istanza driver che si connette a un nome contabile specificato e utilizza una logica di ripetizione specificata con un limite di tentativi personalizzato.

> **Note**
>
> Questo esempio crea anche un'istanza di un oggetto di sistema Amazon Ion (`IonSystem`). È necessario questo oggetto per elaborare i dati Ion quando si eseguono alcune operazioni sui dati in questo riferimento. Per ulteriori informazioni, consulta Come lavorare con 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();
```

Operazioni CRUD

QLDB esegue operazioni di creazione, lettura, lettura, lettura, aggiornamento ed eliminazione (CRUD) come parte di una transazione.

> **Warning**
>
> Come best practice, cerca di rendere le tue transazioni di scrittura strettamente idempotenti.

Rendere le transazioni idempotenti

Ti consigliamo di effettuare transazioni di scrittura idempotenti per evitare effetti collaterali imprevisti in caso di nuovi tentativi. Una transazione è idempotente se può essere eseguita più volte e produrre risultati identici ogni volta.

Ad esempio, si consideri una transazione che inserisce un documento in una tabella denominata`Person`. La transazione deve innanzitutto verificare se il documento esiste già o meno nella tabella. Senza questo controllo, la tabella potrebbe contenere documenti duplicati.

Supponiamo che QLDB esegua correttamente la transazione sul lato server, ma che il client scada in attesa di una risposta. Se la transazione non è idempotente, lo stesso documento potrebbe essere inserito più di una volta in caso di nuovo tentativo.

Utilizzo degli indici per evitare la scansione completa delle tabelle

Si consiglia inoltre di eseguire istruzioni con una clausola`WHERE` predicativa utilizzando un operatore di uguaglianza su un campo indicizzato o un ID di documento, ad esempio`WHERE indexedField = 123` o`WHERE indexedField IN (456, 789)`. Senza questa ricerca indicizzata, QLDB deve eseguire una scansione della tabella, che può portare a timeout delle transazioni o conflitti OCC (Optimistic Concurrency Control).

Per ulteriori informazioni su OCC, consulta[Modello di concorrenza Amazon QLDB](concurrency.html).

Transazioni create implicitamente

Il [QldbDrivermetodo.execute](qldb/latest/developerguide/driver.html#driver.execute) accetta una funzione lambda che riceve un'istanza di [Executor](qldb/latest/developerguide/driver.html#driver.executor), che è possibile utilizzare per eseguire istruzioni. L'`Executor`istanza racchiude una transazione creata implicitamente.

È possibile eseguire istruzioni all'interno della funzione lambda utilizzando il`Executor.execute` metodo. Il driver esegue implicitamente la transazione quando la funzione lambda ritorna.

Le sezioni seguenti mostrano come eseguire operazioni CRUD di base, specificare una logica di ripetizione personalizzata e implementare vincoli di unicità.

> **Note**
>
> Ove applicabile, queste sezioni forniscono esempi di codice di elaborazione dei dati Amazon Ion utilizzando sia la libreria Ion integrata che la libreria di mappe Jackson Ion. Per ulteriori informazioni, consulta Come lavorare con Amazon Ion.

Indice

Creazione di tabelle

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

Creazione di indici di creazione di indici

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

Lettura di documenti

```
// 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
});
```

Utilizzo dei parametri di interrogazione

L'esempio di codice seguente utilizza un parametro di interrogazione di tipo 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
});
```

Il seguente esempio di codice utilizza più parametri di interrogazione.

```
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'esempio di codice seguente utilizza un elenco di parametri di interrogazione.

```
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
```

```
});
```

Utilizzo del mappatore 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();
    }
});
```

Utilizzo dei parametri di interrogazione

L'esempio di codice seguente utilizza un parametro di interrogazione di tipo 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();
    }
});
```

Il seguente esempio di codice utilizza più parametri di interrogazione.

```
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'esempio di codice seguente utilizza un elenco di parametri di interrogazione.

```
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**
>
> Quando si esegue una query senza una ricerca indicizzata, viene richiamata una scansione completa della tabella. In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le query possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Inserimento di documenti di creazione di

Il seguente esempio di codice inserisce i tipi di dati 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);
    }
});
```

Utilizzo del mappatore Jackson

Il seguente esempio di codice inserisce i tipi di dati 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();
    }
});
```

Questa transazione inserisce un documento nella`Person` tabella. Prima dell'inserimento, controlla innanzitutto se il documento esiste già nella tabella. Questo controllo rende la transazione di natura idempotente. Anche se esegui questa transazione più volte, non causerà effetti collaterali indesiderati.

> **Note**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Inserimento di più documenti in un'unica dichiarazione

Per inserire più documenti utilizzando una singolaINSERT istruzione, è possibile passare un parametro di tipo IonList(esplicitamente espresso come un`IonValue`) all'istruzione come segue.

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

Non si racchiude il segnaposto variabile (?) tra parentesi a doppio angolo (`<<...>>`) quando si passa un`IonList`. Nelle istruzioni PartiQL, le parentesi a doppio angolo indicano una raccolta non ordinata nota come borsa.

Perché è richiesto il cast esplicito?

Il TransactionExecutormetodo.execute è sovraccarico. Accetta un numero variabile di`IonValue` argomenti (varargs) o un singolo`List<IonValue>` argomento. In ion-java,`IonList` è implementato come un`List<IonValue>`.

L'impostazione predefinita di Java è l'implementazione del metodo più specifica quando si chiama un metodo sovraccaricato. In questo caso, quando si passa un`IonList` parametro, per impostazione predefinita viene utilizzato il metodo che richiede un`List<IonValue>`. Quando viene invocata, questa implementazione del metodo passa gli`IonValue` elementi dell'elenco come valori distinti. Quindi, per richiamare invece il metodo varargs, devi inserire esplicitamente un`IonList` parametro come un`IonValue`.

Aggiornamento dei documenti

```
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);
```

```
});
```

Utilizzo del mappatore 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**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Eliminazione di documenti

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

Utilizzo del mappatore Jackson

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

> **Note**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Esecuzione di più rendiconti in una transazione

```
// 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;
    });
}
```

Logica di ripetizione dei tentativi

Il`execute` metodo del driver dispone di un meccanismo di ripetizione integrato che riprova la transazione se si verifica un'eccezione ripetibile (ad esempio timeout o conflitti OCC).

2.x

Il numero massimo di tentativi di ripetizione e la strategia di backoff sono configurabili.

Il limite di tentativi predefinito è4 e la strategia di backoff predefinita è DefaultQldbTransactionBackoffStrategy. È possibile impostare la configurazione dei tentativi per istanza del driver e anche per transazione utilizzando un'istanza di RetryPolicy.

L'esempio di codice seguente specifica la logica dei tentativi con un limite di tentativi personalizzato e una strategia di backoff personalizzata per un'istanza del driver.

```
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'esempio di codice seguente specifica la logica dei tentativi con un limite di tentativi personalizzato e una strategia di backoff personalizzata per una determinata transazione. Questa configurazione per`execute` sovrascrive la logica dei tentativi impostata per l'istanza del driver.

```
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

Il numero massimo di tentativi di ripetizione è configurabile. È possibile configurare il limite di tentativi impostando la`retryLimit` proprietà durante l'inizializzazione`PooledQldbDriver`.

Il limite di tentativi predefinito è`4`.

Implementazione dei vincoli di unicità di

QLDB non supporta indici univoci, ma puoi implementare questo comportamento nella tua applicazione.

Supponiamo di voler implementare un vincolo di unicità sul`GovId` campo della`Person` tabella. Per fare ciò, è possibile scrivere una transazione che esegue le seguenti operazioni di creazione di questa operazione che esegue le seguenti operazioni di creazione di questa operazione

1. Asserisci che la tabella non ha documenti esistenti con un valore specificato`GovId`.
2. Inserisci il documento se l'asserzione è valida.

Se una transazione concorrente supera contemporaneamente l'asserzione, solo una delle transazioni verrà confermata con successo. L'altra transazione avrà esito negativo con un'eccezione di conflitto OCC.

L'esempio di codice seguente mostra come implementare questa logica di creazione di vincoli di unicità.

```
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**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Come lavorare con Amazon Ion

Esistono diversi modi per elaborare i dati di Amazon Ion in QLDB. È possibile utilizzare i metodi integrati della libreria Ion per creare e modificare i documenti in modo flessibile in base alle esigenze. Oppure, puoi utilizzare il modulo di formato dati Jackson di FasterXML per Ion per mappare i documenti Ion su semplici vecchi modelli di oggetti Java (POJO).

Le sezioni seguenti forniscono esempi di codice di elaborazione dei dati Ion utilizzando entrambe le tecniche.

Indice

Importazione dei pacchetti Ion

Aggiungi l'artefatto ion-java come dipendenza nel tuo progetto 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>
```

Importa i seguenti pacchetti Ion.

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

Utilizzo del mappatore Jackson

Aggiungi l'artefatto jackson-dataformat-ioncome dipendenza nel tuo progetto Java. QLDB richiede una versione`2.10.0` o successiva.

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>
```

Importa i seguenti pacchetti Ion.

```
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;
```

Inizializzazione di Ion

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

Utilizzo del mappatore Jackson

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

Creazione di oggetti Ion

Il seguente esempio di codice crea un oggetto Ion utilizzando l'`IonStruct`interfaccia e i relativi metodi incorporati.

```
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
```

Utilizzo del mappatore Jackson

Supponiamo di avere una classe di modello mappata in JSON denominata`Person` come segue.

```
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;
    }
}
```

Il seguente esempio di codice crea un`IonStruct` oggetto da un'istanza di`Person`.

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

lettura di oggetti Ion Ion

Il seguente esempio di codice stampa ogni campo dell'`ionStruct`istanza.

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

Utilizzo del mappatore Jackson

Il seguente esempio di codice legge un`IonStruct` oggetto e lo associa a un'istanza di`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
```

Per ulteriori informazioni sull'utilizzo di Ion, consulta la documentazione di Amazon Ion su GitHub. Per altri esempi di codice sull'utilizzo di Ion in QLDB, vedereUtilizzo dei tipi di dati Amazon Ion in Amazon QLDB.

Driver Amazon QLDB per .NET

Per lavorare con i dati contenuti nel tuo registro, puoi connetterti ad Amazon QLDB dalla tua applicazione Microsoft.NET utilizzando un driverAWS fornito. Il driver è indirizzato.NET Standard 2.0. Più specificamente, supporta .NET Core (LTS) 2.1+ e.NET Framework 4.5.2+. Per informazioni sulla compatibilità, vedere .NET Standard nel sito Microsoft Docs.

Consigliamo vivamente di utilizzare il mappatore di oggetti Ion per aggirare completamente la necessità di convertire manualmente tra i tipi Amazon Ion e i tipi C# nativi.

I seguenti argomenti descrivono come iniziare a utilizzare il driver QLDB per .NET.

Argomenti

Risorse per i conducenti

Per ulteriori informazioni sulle funzionalità supportate dal driver.NET, consulta le risorse seguenti:

- Documentazione di riferimento dell'API
- Codice sorgente del driver (GitHub)
- Codice sorgente dell'applicazione di esempio (GitHub)
- Ricettario Amazon Ion
- Mappatore di oggetti ionici (GitHub)

Prerequisiti

Prima di poter utilizzare il driver QLDB per .NET, devi completare le seguenti operazioni:

1. Segui le istruzioniAWS di configurazione inAccesso ad Amazon QLDB. Questo include gli output seguenti:

    1. Per poter utilizzare un a a aAWS a:

   2. Creazione di un utente con le autorizzazioni QLDB appropriate.
   3. Concessione dell'accesso programmatico per lo sviluppo.
2. Scaricare e installare la versione 2.1 o successiva dell'SDK .NET Core dal sito di download di Microsoft.NET.
3. (Facoltativo) Installa un ambiente di sviluppo integrato (IDE) di tua scelta, come Visual Studio, Visual Studio per Mac o Visual Studio Code. Puoi scaricarli dal sito Microsoft Visual Studio.
4. Configura il tuo ambiente di sviluppo per AWS SDK for .NET:

   1. ConfiguraAWS le tue credenziali. Si consiglia di creare un file delle credenziali condiviso.

      Per istruzioni, consulta ConfigurazioneAWS delle credenziali utilizzando un file di credenziali nella Guida per gliAWS SDK for .NET sviluppatori.
   2. Imposta il tuo valore predefinitoRegione AWS. Per sapere come, consulta la sezione Regione AWSselezione.

      Per un elenco completo delle regioni disponibili, consulta gli endpoint e le quote di Amazon QLDB nel Riferimenti generali di AWS.

Successivamente, è possibile configurare un'applicazione di esempio di base ed eseguire brevi esempi di codice oppure installare il driver in un progetto .NET esistente.

- Per installare il driver QLDB e il driverAWS SDK for .NET in un progetto esistente, procedere aInstallazione.
- Per configurare un progetto ed eseguire brevi esempi di codice che dimostrino le transazioni di dati di base su un libro mastro, consulta ilGuida introduttiva rapida al tutorial.

Installazione

Usa il gestore di NuGet pacchetti per installare il driver QLDB per .NET. Ti consigliamo di utilizzare Visual Studio o un IDE di tua scelta per aggiungere le dipendenze del progetto. Il nome del pacchetto driver è Amazon.QLDB.driver.

Ad esempio, in Visual Studio, apri la console di gestioneNuGet Package nel menu Strumenti. Quindi, immetti il seguente comando alPM> prompt.

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

L'installazione del driver installa anche le relative dipendenze, inclusi i pacchetti Amazon IonAWS SDK for .NET e Amazon.

Installa il mappatore di oggetti Ion

La versione 1.3.0 del driver QLDB per .NET introduce il supporto per l'accettazione e la restituzione di tipi di dati C# nativi senza la necessità di lavorare con Amazon Ion. Per utilizzare questa funzionalità, aggiungi il seguente pacchetto al tuo progetto.

- Amazon.QLDB.driver.Serialization — Una libreria in grado di mappare i valori Ion su semplici vecchi oggetti CLR (POCO) in C# e viceversa. Questo mappatore di oggetti Ion consente all'applicazione di interagire direttamente con i tipi di dati C# nativi senza la necessità di lavorare con Ion. Per una breve guida su come usare questa libreria, consultate il file Serialization.md nel GitHub repository`awslabs/amazon-qldb-driver-dotnet`.

Per poter installare questo pacchetto, immetti il seguente comando.

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

Per brevi esempi di codice su come eseguire transazioni di dati di base su un libro mastro, vedere ilDocumentazione di riferimento del libro.

Driver Amazon QLDB per .NET — Tutorial di avvio rapido

In questo tutorial, imparerai come configurare una semplice applicazione utilizzando il driver Amazon QLDB per .NET. Questa guida include procedura per installare il driver ed esempi di creazione, lettura, aggiornamento ed eliminazione CRUD.

Argomenti

- Prerequisiti
- Fase 1: Configurazione del progetto
- Fase 2: Inizializzazione del driver: Inizializzazione del driver: Inizializzazione
- Fase 3: Creare una tabella e un indice: Creare una tabella e un indice: Creazione
- Fase 4: Inserimento di un documento: inserisci un documento
- Fase 5: esegui esegui query sul documento: esegui
- Fase 6: Aggiornare il documento: Aggiornamento del documento: Aggiornamento
- Esecuzione dell'applicazione completa

Prerequisiti

Prima di iniziare, assicurati di completare quanto segue procedura seguente procedura:

1. Se non è stato ancora fatto. Se non è stato ancora fatto.Prerequisiti Ciò include la registrazioneAWS, la concessione dell'accesso programmatico per lo sviluppo e l'installazione dell'SDK.NET Core.
2. Crea un libro contabile denominato`quick-start`.

   Per informazioni su come creare un libro mastro, consultaOperazioni di base per i libri mastri Amazon QLDB oFase 1: creazione di un nuovo libro mastro in Guida introduttiva alla console.

Fase 1: Configurazione del progetto

In primo luogo, configura il tuo progetto .NET.

1. Per creare ed eseguire un'applicazione modello, inserisci i seguenti`dotnet` comandi su un terminale come bash o Command Prompt. PowerShell

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

   Questo modello crea una cartella denominata`Amazon.QLDB.QuickStartGuide`. In quella cartella, crea un progetto con lo stesso nome e un file denominato`Program.cs`. Il programma contiene codice che visualizza l'output`Hello World!`.
2. Usa il gestore di NuGet pacchetti per installare il driver QLDB per .NET. Ti consigliamo di utilizzare Visual Studio o un IDE di tua scelta per aggiungere dipendenze al tuo progetto. Il nome del pacchetto driver è Amazon.QLDB.driver.
   - Ad esempio, in Visual Studio, apri la console di gestioneNuGet Package nel menu Strumenti. Quindi, inserisci il seguente comando seguente comando al seguente comando seguente comando al seguente comando al seguente`PM>` comando al seguente comando

     ```
     PM> Install-Package Amazon.QLDB.Driver
     ```
   - In alternativa, è possibile eseguire i seguenti comandi sul seguente comando, è possibile eseguire i seguenti comandi sul seguente comando sul seguente comando

     ```
     $ cd Amazon.QLDB.QuickStartGuide
     ```

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

L'installazione del driver installa anche le relative dipendenze, incluse le librerie Amazon Ion AWS SDK for .NETe Amazon.

3. Installa la libreria di serializzazione del driver.

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

4. Apri il file `Program.cs`.

   Quindi, aggiungi in modo incrementale gli esempi di codice nei passaggi seguenti per provare alcune operazioni CRUD di base. In alternativa, puoi saltare il step-by-step tutorial ed eseguire invece l'applicazione completa.

> **Note**
>
> - Scelta tra API sincrone e asincrone: il driver fornisce API sincrone e asincrone. Per le applicazioni ad alta richiesta che gestiscono più richieste senza blocchi, consigliamo di utilizzare le API asincrone per migliorare le prestazioni. Il driver offre API sincrone come ulteriore comodità per le basi di codice esistenti scritte in modo sincrono.
>
>   Questo tutorial include esempi di codice sincrono e asincrono. Per ulteriori informazioni sulle API, consulta leAsyncQldbDriver interfacce IQldbDriver e I nella documentazione delle API.
> - Elaborazione dei dati Amazon Ion: questo tutorial fornisce esempi di codice di elaborazione dei dati Amazon Ion utilizzando la mappatrice di oggetti Ion per impostazione predefinita. QLDB ha introdotto il mappatore di oggetti Ion nella versione 1.3.0 del driver.NET. Ove applicabile, questo tutorial fornisce anche esempi di codice utilizzando la libreria Ion standard come alternativa. Per ulteriori informazioni, consulta Uso di Amazon Ion.

Fase 2: Inizializzazione del driver: Inizializzazione del driver: Inizializzazione

Inizializza un'istanza del driver che si connette al registro denominato`quick-start`. Aggiungi il seguente codice seguente al seguente codice seguente al seguente codice seguente al`Program.cs` seguente codice seguente

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();
        }
    }
}
```

Utilizzo della libreria 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();
        }
    }
}
```

Fase 3: Creare una tabella e un indice: Creare una tabella e un indice: Creazione

Per il resto di questo tutorial fino al passaggio 6, è necessario aggiungere i seguenti esempi di codice all'esempio di codice precedente.

In questo passaggio, il codice seguente mostra come eseguire`CREATE TABLE` le`CREATE INDEX` istruzioni. Crea una tabella denominata`Person` e un indice per il`firstName` campo su quella tabella.

Gli indici sono necessari per ottimizzare le prestazioni delle query e contribuire a limitare le eccezioni di conflitto OCC (Optimistic Concurrency Control).

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)");
});
```

Fase 4: Inserimento di un documento: inserisci un documento

L'seguente esempio di codice seguente seguente seguente mostra come eseguire un'istruzione seguente seguente seguente seguente mostra come eseguire un'`INSERT`istruzione seguente QLDB supporta il linguaggio di interrogazione PartiQL (compatibile con SQL) e il formato dati Amazon Ion (superset di JSON).

Aggiungete il seguente codice che inserisce un documento nella`Person` tabella.

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);
});
```

Utilizzo della libreria 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

Per inserire più documenti utilizzando una singolaINSERT istruzione, è possibile passare un parametro di tipo Ion list all'istruzione come segue.

```
// people is an Ion list
txn.Execute("INSERT INTO Person ?", people);
```

Non si racchiude il segnaposto variabile (?) tra parentesi a doppio angolo (<<...>>) quando si passa un elenco Ion. Nelle istruzioni PartiQL, le parentesi a doppio angolo indicano una raccolta non ordinata nota come borsa.

Fase 5: esegui esegui query sul documento: esegui

L'seguente esempio di codice seguente seguente seguente mostra come eseguire un'istruzione seguente seguente seguente seguente mostra come eseguire un'`SELECT`istruzione seguente

Aggiungete il codice seguente per interrogare un documento dalla`Person` tabella.

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
```

```
}
```

Utilizzo della libreria 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);
}
```

Questo esempio utilizza un punto interrogativo (?) come segnaposto variabile per passare le informazioni del documento alla dichiarazione. Quando si utilizzano segnaposti, è necessario passare un valore di tipo`IonValue`.

Fase 6: Aggiornare il documento: Aggiornamento del documento: Aggiornamento

L'seguente esempio di codice seguente seguente seguente mostra come eseguire un'istruzione seguente seguente seguente seguente mostra come eseguire un'`UPDATE`istruzione seguente

1. Aggiungi il codice seguente che aggiorna un documento nella`Person` tabella eseguendo l'aggiornamento`age` a 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. Interroga nuovamente il documento per visualizzare il valore aggiornato.

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. Per eseguire l'applicazione, inserisci il seguente comando dalla directory principale della directory del tuo`Amazon.QLDB.QuickStartGuide` progetto.

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

Utilizzo della libreria Ion

1. Aggiungi il codice seguente che aggiorna un documento nella`Person` tabella eseguendo l'aggiornamento`age` a 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. Interroga nuovamente il documento per visualizzare il valore aggiornato.

   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. Per eseguire l'applicazione, inserisci il seguente comando dalla directory principale della directory del tuo`Amazon.QLDB.QuickStartGuide` progetto.

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

Esecuzione dell'applicazione completa

Il seguente esempio di codice è la versione completa dell'`Program.cs`applicazione. Invece di eseguire i passaggi precedenti singolarmente, puoi anche copiare ed eseguire questo esempio di codice dall'inizio alla fine. Questa applicazione dimostra alcune operazioni CRUD di base sul registro denominato`quick-start`.

> **Note**
>
> Prima di eseguire questo codice, assicuratevi di non avere già una tabella attiva denominata`Person` nel`quick-start` libro mastro.

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
            }

        }
    }
}
```

Utilizzo della libreria 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);
            }
        }
    }
}
```

Per eseguire l'applicazione completa, inserisci il seguente comando dalla directory principale della directory del`Amazon.QLDB.QuickStartGuide` progetto.

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

Driver Amazon QLDB per .NET — Riferimento al ricettario

Questa guida di riferimento mostra i casi d'uso più comuni del driver Amazon QLDB per .NET. Fornisce esempi di codice C# che illustra come utilizzare il driver per eseguire operazioni di creazione, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, lettura, aggiornamento ed eliminazione (CRUD) Include anche esempi di codice per l'elaborazione dei dati di Amazon Ion. Inoltre, questa guida evidenzia le migliori pratiche per rendere le transazioni idempotenti e implementare i vincoli di unicità.

> **Note**
>
> Questo argomento fornisce esempi di codice di elaborazione dei dati Amazon Ion utilizzando la mappatrice di oggetti Ion per impostazione predefinita. QLDB ha introdotto il mappatore di oggetti Ion nella versione 1.3.0 del driver.NET. Ove applicabile, questo argomento fornisce anche esempi di codice che utilizzano la libreria Ion standard come alternativa. Per ulteriori informazioni, consulta Uso di Amazon Ion.

Indice

Importazione del driver

Il seguente esempio di codice importa il driver.

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

Utilizzo della libreria Ion

```
using Amazon.QLDB.Driver;
using Amazon.IonDotnet.Builders;
```

Istanziazione del driver

L'esempio di codice seguente crea un'istanza del driver che si connette a un nome di registro specificato utilizzando le impostazioni predefinite.

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();
```

Utilizzo della libreria Ion

Async

```
IAsyncQldbDriver driver = AsyncQldbDriver.Builder().WithLedger("vehicle-
registration").Build();
```

Sync

```
IQldbDriver driver = QldbDriver.Builder().WithLedger("vehicle-
registration").Build();
```

Operazioni CRUD

QLDB esegue operazioni di creazione, lettura, aggiornamento ed eliminazione (CRUD) come parte di una transazione.

> **Warning**
>
> Come best practice, cerca di rendere le tue transazioni di scrittura rigorosamente idempotenti.

Rendere le transazioni idempotenti

Ti consigliamo di effettuare transazioni di scrittura idempotenti per evitare effetti collaterali imprevisti in caso di nuovi tentativi. Una transazione è idempotente se può essere eseguita più volte e produrre risultati identici ogni volta.

Ad esempio, si consideri una transazione che inserisce un documento in una tabella denominata`Person`. La transazione deve innanzitutto verificare se il documento esiste già o meno nella tabella. Senza questo controllo, la tabella potrebbe contenere documenti duplicati.

Supponiamo che QLDB esegua correttamente la transazione sul lato server, ma che il client scada in attesa di una risposta. Se la transazione non è idempotente, lo stesso documento potrebbe essere inserito più di una volta in caso di nuovo tentativo.

Utilizzo degli indici per evitare la scansione completa delle tabelle

Si consiglia inoltre di eseguire istruzioni con una clausola`WHERE` predicativa utilizzando un operatore di uguaglianza su un campo indicizzato o un ID di documento, ad esempio`WHERE indexedField = 123` o`WHERE indexedField IN (456, 789)`. Senza questa ricerca indicizzata, QLDB deve eseguire una scansione della tabella, che può portare a timeout delle transazioni o conflitti OCC (Optimistic Concurrency Control).

Per ulteriori informazioni su OCC, consultaModello di concorrenza Amazon QLDB.

Transazioni create implicitamente

Il metodo Amazon.QLDB.driver.iQldbDriver .Execute accetta una funzione lambda che riceve un'istanza di Amazon.QLDB.driver. TransactionExecutor, che è possibile utilizzare per eseguire istruzioni. L'istanza di`TransactionExecutor` avvolge una transazione creata implicitamente.

È possibile eseguire istruzioni all'interno della funzione lambda utilizzando il`Execute` metodo dell'esecutore della transazione. Il driver esegue implicitamente la transazione quando la funzione lambda ritorna.

Le sezioni seguenti mostrano come eseguire operazioni CRUD di base, specificare una logica di ripetizione personalizzata e implementare vincoli di unicità.

Indice

Creazione di tabelle

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 }
}
```

Utilizzo della libreria 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"
    // }
}
```

Creazione di indici

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 }
}
```

Utilizzo della libreria 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"
    // }
}
```

Lettura di documenti

```
// 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**
>
> Quando si esegue una query senza una ricerca indicizzata, viene richiamata una scansione completa della tabella. In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le query possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Utilizzo dei parametri della query

L'esempio di codice seguente utilizza un parametro di interrogazione di tipo 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'esempio di codice seguente utilizza più parametri di interrogazione di tipo 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'esempio di codice seguente utilizza una matrice di parametri di query di tipo 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'esempio di codice seguente utilizza un elenco C# come valore.

```
// 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
    //     },
    //     ]
    // }
}
```

Utilizzo della libreria 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**
>
> Quando si esegue una query senza una ricerca indicizzata, viene richiamata una scansione completa della tabella. In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le query possono avere una

maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

L'esempio di codice seguente utilizza un parametro di interrogazione di tipo 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.
}
```

Il seguente esempio di codice utilizza più parametri di interrogazione.

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'esempio di codice seguente utilizza un elenco di parametri di interrogazione.

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'esempio di codice seguente utilizza un elenco Ion come valore. Per ulteriori informazioni sull'utilizzo dei diversi tipi di Ion, consultaUtilizzo dei tipi di dati Amazon Ion in 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"
    //     }
    //     ]
    // }
}
```

Inserimento dei documenti

Il seguente esempio di codice inserisce i tipi di dati 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));
});
```

Utilizzo della libreria 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);
});
```

Questa transazione inserisce un documento nella`Person` tabella. Prima dell'inserimento, controlla innanzitutto se il documento esiste già nella tabella. Questo controllo rende la transazione di natura idempotente. Anche se esegui questa transazione più volte, non causerà effetti collaterali indesiderati.

> **Note**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Inserimento di più documenti in un'unica dichiarazione

Per inserire più documenti utilizzando una singolaINSERT istruzione, è possibile passare un`List` parametro C# all'istruzione come segue.

```
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 }
}
```

Utilizzo della libreria Ion

Per inserire più documenti utilizzando una singolaINSERT istruzione, è possibile passare un parametro di tipo Ion list all'istruzione come segue.

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"
    // }
}
```

Non si racchiude il segnaposto variabile (?) tra parentesi a doppio angolo (<<...>>) quando si passa un elenco di Ion. Nelle istruzioni PartiQL, le parentesi a doppio angolo indicano una raccolta non ordinata nota come borsa.

Aggiornamento dei documenti

```
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 }
}
```

Utilizzo della libreria 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**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Eliminazione di documenti

```
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 }
}
```

Utilizzo della libreria 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**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Esecuzione di più rendiconti in una transazione

```
// 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;
    });
}
```

Utilizzo della libreria 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;
    });
}
```

Logica di ripetizione dei tentativi

Per informazioni sulla logica di ripetizione integrata nel driver, vedereInformazioni sulla politica di riprova con il driver in Amazon QLDB.

Implementazione dei vincoli di unicità

QLDB non supporta indici univoci, ma puoi implementare questo comportamento nella tua applicazione.

Si supponga di voler implementare un vincolo di unicità sul`GovId` campo della`Person` tabella. Questa operazione può essere eseguita in modo da scrivere una transazione che esegue le seguenti operazioni:

1. Asserisci che la tabella non ha documenti esistenti con un valore specificato`GovId`.
2. Inserisci il documento se l'asserzione è valida.

Se una transazione concorrente supera contemporaneamente l'asserzione, solo una delle transazioni verrà confermata con successo. L'altra transazione avrà esito negativo con un'eccezione di conflitto OCC.

L'esempio di codice seguente mostra come implementare questa logica di vincolo di unicità.

```
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));
});
```

Utilizzo della libreria 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**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Uso di Amazon Ion

Esistono diversi modi per elaborare i dati Amazon Ion in QLDB. È possibile utilizzare la libreria Ion per creare e modificare i valori Ion. In alternativa, è possibile utilizzare il mappatore di oggetti Ion per mappare vecchi oggetti CLR (POCO) semplici in C# da e verso i valori Ion. La versione 1.3.0 del driver QLDB per .NET introduce il supporto per il mappatore di oggetti Ion.

Le sezioni seguenti forniscono esempi di codice di elaborazione dei dati Ion utilizzando entrambe le tecniche.

Indice

- Creazione di tipi di ioni
- Ottenere un dump binario Ion
- Ottenere un dump di testo Ion

Importazione del modulo Ion

```
using Amazon.IonObjectMapper;
```

Utilizzo della libreria Ion

```
using Amazon.IonDotnet.Builders;
```

Creazione di tipi di ioni

L'esempio di codice seguente mostra come eseguire operazioni di creazione di oggetti C# utilizzando la mappatrice di oggetti Ion.

```
// 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);
```

Utilizzo della libreria Ion

I seguenti esempi di codice mostrano i due modi per creare valori Ion utilizzando la libreria Ion.

Uso di **`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);
```

Uso di **`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);
```

Ottenere un dump binario 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()));
```

Utilizzo della libreria 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()));
```

Ottenere un dump di testo 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()));
```

Utilizzo della libreria 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());
```

Per ulteriori informazioni sull'utilizzo di Ion, consulta la documentazione di Amazon Ion su GitHub. Per altri esempi di codice sull'utilizzo di Ion in QLDB, vedereUtilizzo dei tipi di dati Amazon Ion in Amazon QLDB.

Driver Amazon QLDB per Go

Per lavorare con i dati contenuti nel tuo registro, puoi connetterti ad Amazon QLDB dall'applicazione Go utilizzando un driverAWS fornito. Gli argomenti seguenti descrivono come iniziare a utilizzare il driver QLDB per Go.

Argomenti

- Risorse per i conducenti
- Prerequisiti
- Installazione
- Driver Amazon QLDB per Go — Tutorial di avvio rapido
- Driver Amazon QLDB per Go — Riferimento al ricettario

Risorse per i conducenti

Per ulteriori informazioni sulle funzionalità supportate dal driver Go, consulta le risorse seguenti:

- Riferimento API: 3.x, 2.x, 1.x
- Codice sorgente del driver (GitHub)

- Ricettario Amazon Ion

Prerequisiti

Per poter utilizzare il driver QLDB per Go, occorre:

1. Segui le istruzioniAWS di configurazione inAccesso ad Amazon QLDB. Questo include gli output seguenti:
    1. Registrazione adAWS.
    2. Crea un utente con le autorizzazioni QLDB appropriate.
    3. Concessione dell'accesso programmatico per lo sviluppo.
2. (Facoltativo) Installa un ambiente di sviluppo integrato (IDE) di tua scelta. Per un elenco degli IDE più usati per Go, consulta i plugin e gli IDE dell'editor sul sito Web Go.
3. Scarica e installa una delle seguenti versioni di Go dal sito di download di Go:
    - 1.15 o versione successiva — driver QLDB per Go v3
    - 1.14 — driver QLDB per Go v1 o v2
4. Configura il tuo ambiente di sviluppo per AWS SDK for Go:
    1. ConfiguraAWS le tue credenziali. Si consiglia: creare un file delle credenziali condiviso.

        Per istruzioni, consulta Specificare le credenziali nella Guida per gliAWS SDK for Go sviluppatori.
    2. Imposta il tuo valore predefinitoRegione AWS. Per sapere come, consulta Specificare ilRegione AWS.

        Per un elenco completo delle regioni disponibili, consulta gli endpoint e le quote di Amazon QLDB nel Riferimenti generali di AWS.

Successivamente, puoi configurare un'applicazione di esempio di base ed eseguire brevi esempi di codice oppure puoi installare il driver in un progetto Go esistente.

- Per installare il driver QLDB e il driverAWS SDK for Go in un progetto esistente, procedere aInstallazione.
- Per configurare un progetto ed eseguire brevi esempi di codice che dimostrino le transazioni di dati di base su un libro mastro, consulta ilFase 4 di Quick Start.

Installazione

Il driver QLDB per Go è open source nel GitHub repository awslabs/amazon-qldb-driver-go. QLDB supporta le seguenti versioni dei driver e le relative dipendenze Go.

Versione driver	Versione Go	Stato	Data di rilascio:
1.x	1.14 o versione successiva	Versione di produzione	16 giugno 2021
2.x	1.14 o versione successiva	Versione di produzione	21 luglio 2021
3.x	1.15 o versioni successive	Versione di produzione	10 novembre 2022

Come installare il driver

1. Assicurati che il tuo progetto utilizzi i moduli Go per installare le dipendenze del progetto.
2. Nella cartella del progetto, inserisci il seguentego `get` comando.

   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'installazione del driver installa anche le relative dipendenze, inclusi i pacchetti AWS SDK for Goor AWS SDK for Gov2 e Amazon Ion.

Per brevi esempi di codice su come eseguire transazioni di dati di base su un libro mastro, vedere ilRiferimento del libro di cucina.

Driver Amazon QLDB per Go — Tutorial di avvio rapido

In questo tutorial, imparerai come configurare una semplice applicazione usando l'ultima versione del driver Amazon QLDB for Go. Questa guida include passaggi per installare il driver ed esempi in codice breve di creazione, lettura, aggiornamento ed eliminazione (CRUD).

Argomenti

Prerequisiti

Prima di iniziare, eseguire quanto segue:

1. FasePrerequisiti 6, se non l'hai già fatto. Ciò include la registrazioneAWS, la concessione dell'accesso programmatico per lo sviluppo e l'installazione di Go.
2. Crea un libro contabile denominato`quick-start`.

   Per informazioni su come creare un libro mastro, consultaOperazioni di base per i libri mastri Amazon QLDB oFase 1: creazione di un nuovo libro mastro in Guida introduttiva alla console.

Fase 1: Installare il driver del driver

Assicurati che il tuo progetto utilizzi i moduli Go per installare le dipendenze del progetto.

Nella cartella del progetto, inserisci il seguentego `get` comando.

```
$ go get -u github.com/awslabs/amazon-qldb-driver-go/v3/qldbdriver
```

L'installazione del driver installa anche le relative dipendenze, inclusi i pacchetti AWS SDK for Gov2 e Amazon Ion.

Fase 2: Importazione dei pacchetti dei pacchetti

Importa i seguentiAWS pacchetti.

```
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"
)
```

Fase 3: inizializzare il driver del driver

Inizializza un'istanza del driver che si connette al registro denominato`quick-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**
>
> In questo esempio di codice, sostituisci *us-east-1* con ilRegione AWS luogo in cui hai creato il tuo libro mastro.

Fase 4: Crea una tabella e un indice

L'esempio di codice seguente mostra come eseguireCREATE INDEX istruzioniCREATE TABLE and.

```
_, 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)
}
```

Questo codice crea una tabella denominataPeople e gli indici per iage campifirstName e di quella tabella. Gli indici sono necessari per ottimizzare le prestazioni delle query e contribuire a limitare le eccezioni di conflitto OCC (Optimistic Concurrency Control).

Fase 5: Inserire un documento

Gli esempi di codice seguenti mostrano come eseguire un'`INSERT`istruzione. QLDB supporta il linguaggio di interrogazione PartiQL (compatibile con SQL) e il formato dati Amazon Ion (superset di JSON).

Utilizzo di PartiQL letterale

Il codice seguente inserisce un documento nella`People` tabella utilizzando un'istruzione partiQL letterale a stringa.

```
_, 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)
}
```

Utilizzo dei tipi di dati Ion

Analogamente al pacchetto JSON integrato di Go, puoi suddividere e deselezionare i tipi di dati Go da e verso Ion.

1. Si supponga di avere la seguente struttura Go`Person`.

   ```
   type Person struct {
       FirstName string `ion:"firstName"`
       LastName  string `ion:"lastName"`
       Age       int    `ion:"age"`
   }
   ```

2. Creare un'istanza di `Person`.

   ```
   person := Person{"John", "Doe", 54}
   ```

   L'autista crea`person` per te una rappresentazione testuale con codifica ION.

> **Important**
>
> Affinché marshal e unmarshal funzionino correttamente, è necessario esportare i nomi dei campi della struttura dati Go (prima lettera maiuscola).

3. Passa l'`person`istanza al`Execute` metodo della transazione.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    return txn.Execute("INSERT INTO People ?", person)
})
if err != nil {
    panic(err)
}
```

Questo esempio utilizza un punto interrogativo (?) come segnaposto variabile per passare le informazioni del documento alla dichiarazione. Quando si utilizzano segnaposti, è necessario passare un valore di testo con codifica ION.

> **Tip**
>
> Per inserire più documenti utilizzando una singolaINSERT istruzione, è possibile passare un elenco di parametri di tipo all'istruzione come segue.
>
> ```
> // people is a list
> txn.Execute("INSERT INTO People ?", people)
> ```
>
> Non si racchiude la variabile placeholder (?) tra parentesi a doppio angolo (`<<...>>`) quando si passa un elenco. Nelle istruzioni PartiQL, le parentesi a doppio angolo indicano una raccolta non ordinata nota come borsa.

Fase 6: Esecuzione della query del documento

L'esempio di codice seguente mostra come eseguire un'`SELECT`istruzione.

```
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")
}
```

Questo esempio interroga il documento dalla`People` tabella, presuppone che il set di risultati non sia vuoto e restituisce il documento dal risultato.

Fase 7: aggiornamento del documento

L'esempio di codice seguente mostra come eseguire un'`UPDATE`istruzione.

```
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)
}
```

Fase 8: Interrogare il documento aggiornato

L'esempio di codice seguente interroga la`People` tabella`firstName` e restituisce tutti i documenti del set di risultati.

```
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")
}
```

Fase 9: Eliminazione del tavolo.

L'esempio di codice seguente mostra come eseguire un'`DROP TABLE`istruzione.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    return txn.Execute("DROP TABLE People")
})
if err != nil {
    panic(err)
}
```

Esecuzione dell'applicazione completa

Il seguente esempio di codice è la versione completa dell'applicazione. Invece di eseguire i passaggi precedenti singolarmente, puoi anche copiare ed eseguire questo esempio di codice dall'inizio alla fine. Questa applicazione dimostra alcune operazioni CRUD di base sul registro denominato`quick-start`.

> **Note**
>
> Prima di eseguire questo codice, assicuratevi di non avere già una tabella attiva denominata`People` nel`quick-start` libro mastro.

```
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)
    }
}
```

Driver Amazon QLDB per Go — Riferimento al ricettario

Questa guida di riferimento mostra i casi d'uso più comuni del driver Amazon QLDB per Go. Fornisce esempi di codice Go che mostrano come utilizzare il driver per eseguire operazioni di creazione, lettura Include anche esempi di codice per l'elaborazione dei dati di Amazon Ion. Inoltre, questa guida evidenzia le migliori pratiche per rendere le transazioni idempotenti e implementare i vincoli di unicità.

> **Note**
>
> Ove applicabile, alcuni casi d'uso presentano esempi di codice diversi per ciascuna versione principale supportata del driver QLDB per Go.

Indice

Importazione del driver

Il seguente esempio di codice importa il driver e altriAWS pacchetti richiesti.

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**
>
> Questo esempio importa anche il pacchetto Amazon Ion (`amzn/ion-go/ion`). È necessario questo pacchetto per elaborare i dati Ion quando si eseguono alcune operazioni sui dati in questo riferimento. Per ulteriori informazioni, consulta Come lavorare con Amazon Ion.

Esempio di creazione del driver

L'esempio di codice seguente crea un'istanza del driver che si connette a un nome di registro specificato in un datoRegione 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)
}
```

Operazioni CRUD

QLDB esegue operazioni di creazione, lettura, lettura, lettura, lettura, aggiornamento ed eliminazione (CRUD) come parte di una transazione.

> **Warning**
>
> Come best practice, cerca di rendere le tue transazioni di scrittura rigorosamente idempotenti.

Rendere le transazioni idempotenti

Ti consigliamo di effettuare transazioni di scrittura idempotenti per evitare effetti collaterali imprevisti in caso di nuovi tentativi. Una transazione è idempotente se può essere eseguita più volte e produrre risultati identici ogni volta.

Ad esempio, si consideri una transazione che inserisce un documento in una tabella denominata`Person`. La transazione deve innanzitutto verificare se il documento esiste già o meno nella tabella. Senza questo controllo, la tabella potrebbe contenere documenti duplicati.

Supponiamo che QLDB esegua correttamente la transazione sul lato server, ma che il client scada in attesa di una risposta. Se la transazione non è idempotente, lo stesso documento potrebbe essere inserito più di una volta in caso di nuovo tentativo.

Utilizzo degli indici per evitare la scansione completa delle tabelle

Si consiglia inoltre di eseguire istruzioni con una clausola`WHERE` predicativa utilizzando un operatore di uguaglianza su un campo indicizzato o un ID di documento, ad esempio`WHERE indexedField = 123` o`WHERE indexedField IN (456, 789)`. Senza questa ricerca indicizzata, QLDB deve eseguire una scansione della tabella, che può portare a timeout delle transazioni o conflitti OCC (Optimistic Concurrency Control).

Per ulteriori informazioni su OCC, consultaModello di concorrenza Amazon QLDB.

Transazioni create implicitamente

La funzione QLDBDriver.execute accetta una funzione lambda che riceve un'istanza di Transaction, che è possibile utilizzare per eseguire istruzioni. L'istanza di`Transaction` avvolge una transazione creata implicitamente.

È possibile eseguire istruzioni all'interno della funzione lambda utilizzando la`Transaction.Execute` funzione. Il driver esegue implicitamente la transazione quando la funzione lambda ritorna.

Le sezioni seguenti mostrano come eseguire operazioni CRUD di base, specificare una logica di ripetizione personalizzata e implementare vincoli di unicità.

Indice

Creazione di tabelle

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("CREATE TABLE Person")
})
```

Creazione di indici

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("CREATE INDEX ON Person(GovId)")
})
```

Lettura di documenti

```
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)
}
```

Utilizzo dei parametri della query

L'esempio di codice seguente utilizza un parametro di interrogazione di tipo nativo.

```
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)
}
```

Il seguente esempio di codice utilizza più parametri di interrogazione.

```
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'esempio di codice seguente utilizza un elenco di parametri di interrogazione.

```
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**
>
> Quando si esegue una query senza una ricerca indicizzata, viene richiamata una scansione completa della tabella. In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le query possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Inserimento di documenti

Il seguente esempio di codice inserisce tipi di dati nativi.

```
_, 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
})
```

Questa transazione inserisce un documento nella`Person` tabella. Prima dell'inserimento, controlla innanzitutto se il documento esiste già nella tabella. Questo controllo rende la transazione di natura idempotente. Anche se esegui questa transazione più volte, non causerà effetti collaterali indesiderati.

> **Note**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Inserimento di più documenti in un'unica dichiarazione

Per inserire più documenti utilizzando una singolaINSERT istruzione, è possibile passare un elenco di parametri di tipo all'istruzione come segue.

```
// people is a list
txn.Execute("INSERT INTO People ?", people)
```

Non si racchiude la variabile placeholder (?) tra parentesi a doppio angolo (`<<...>>`) quando si passa un elenco. Nelle istruzioni PartiQL, le parentesi a doppio angolo indicano una raccolta non ordinata nota come borsa.

Aggiornamento dei documenti

Il seguente esempio di codice utilizza tipi di dati nativi.

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("UPDATE Person SET FirstName = ? WHERE GovId = ?", "John",
 "TOYENC486FH")
})
```

> **Note**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Eliminazione di documenti

Il seguente esempio di codice utilizza tipi di dati nativi.

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("DELETE FROM Person WHERE GovId = ?", "TOYENC486FH")
})
```

> **Note**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Esecuzione di più rendiconti in una transazione

```
// 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
}
```

Logica di ripetizione dei tentativi

La`Execute` funzione del driver dispone di un meccanismo di ripetizione integrato che riprova la transazione se si verifica un'eccezione ripetibile (ad esempio timeout o conflitti OCC). Il numero massimo di tentativi di ripetizione e la strategia di backoff sono configurabili.

Il limite di tentativi predefinito è`4` e la strategia di backoff predefinita è [ExponentialBackoffStrategy](#)basata su`10` millisecondi. È possibile impostare la politica di ripetizione per istanza del driver e anche per transazione utilizzando un'istanza di [RetryPolicy](#).

L'esempio di codice seguente specifica la logica dei tentativi con un limite di tentativi personalizzato e una strategia di backoff personalizzata per un'istanza del driver.

```
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'esempio di codice seguente specifica la logica dei tentativi con un limite di tentativi personalizzato e una strategia di backoff personalizzata per una particolare funzione anonima. La`SetRetryPolicy` funzione sostituisce la politica di ripetizione impostata per l'istanza del driver.

```
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")
  })
}
```

Implementazione dei vincoli di unicità

QLDB non supporta indici univoci, ma puoi implementare questo comportamento nella tua applicazione.

Si supponga di voler implementare un vincolo di unicità sul`GovId` campo della`Person` tabella. Per eseguire questa operazione, è possibile scrivere una transazione che esegue le seguenti operazioni di creazione, lettura di una transazione che esegue le seguenti operazioni di creazione

1. Asserisci che la tabella non ha documenti esistenti con un valore specificato`GovId`.
2. Inserisci il documento se l'asserzione è valida.

Se una transazione concorrente supera contemporaneamente l'asserzione, solo una delle transazioni verrà confermata con successo. L'altra transazione avrà esito negativo con un'eccezione di conflitto OCC.

L'esempio di codice seguente mostra come implementare questa logica di vincolo di unicità.

```
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**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Come lavorare con Amazon Ion

Le sezioni seguenti mostrano come utilizzare i dati Amazon Ion per elaborare i dati Ion.

Indice

Importazione del modulo Ion

```
import "github.com/amzn/ion-go/ion"
```

Creazione di tipi di Ion

La libreria Ion per Go attualmente non supporta il Document Object Model (DOM), quindi non è possibile creare tipi di dati Ion. Ma puoi distinguere e deselezionare tra i tipi nativi di Go e i binari Ion quando lavori con QLDB.

Come ottenere Ion Binary

```
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]
```

Come ottenere testo 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"}
```

Per ulteriori informazioni su Ion, consulta la documentazione di Amazon Ion su GitHub. Per altri esempi di codice sull'utilizzo di Ion in QLDB, vedereUtilizzo dei tipi di dati Amazon Ion in Amazon QLDB.

Driver Amazon QLDB per Node.js

Per utilizzare i dati nel registro, puoi connetterti ad Amazon QLDB dall'applicazione Node.js utilizzando un driver fornito. AWS Negli argomenti seguenti viene descritto come iniziare a utilizzare il driver QLDB per Node.js.

Argomenti

- Risorse per i driver
- Prerequisiti
- Installazione
- Consigli di configurazione
- Driver Amazon QLDB per Node.js — Tutorial di avvio rapido
- Driver Amazon QLDB per Node.js — Riferimento al ricettario

Risorse per i driver

Per ulteriori informazioni sulle funzionalità supportate dal driver Node.js, consultate le seguenti risorse:

- Riferimento API: 3.x, 2.x, 1.x
- Codice sorgente del driver () GitHub
- Codice sorgente dell'applicazione di esempio (GitHub)
- Esempi di codice Amazon Ion
- Crea una semplice operazione CRUD e un flusso di dati su QLDB usando (BlogAWS Lambda) AWS

Prerequisiti

Prima di iniziare a utilizzare il driver QLDB per Node.js, è necessario effettuare le seguenti operazioni:

1. Segui le istruzioni di AWS configurazione riportate in. Accesso ad Amazon QLDB Questo include gli output seguenti:

   1. Iscriviti aAWS.
   2. Crea un utente con le autorizzazioni QLDB appropriate.
   3. Concedi l'accesso programmatico per lo sviluppo.
2. Installa Node.js versione 14.x o successiva dal sito di download Node.js. (Le versioni precedenti del driver supportano la versione 10.x o successiva di Node.js).
3. Configura il tuo ambiente di sviluppo per l'AWSSDK per Node.js JavaScript :

   1. Configura le tue AWS credenziali. Ti consigliamo di creare un file di credenziali condiviso.

      Per istruzioni, consulta Caricamento delle credenziali in Node.js dal file delle credenziali condivise nella Guida per gli AWS SDK for JavaScript sviluppatori.
   2. Imposta i tuoi valori predefiniti. Regione AWS Per sapere come, vedi Impostazione di Regione AWS.

      Per un elenco completo delle regioni disponibili, consulta gli endpoint e le quote di Amazon QLDB nel. Riferimenti generali di AWS

Successivamente, puoi scaricare l'applicazione di esempio completa del tutorial oppure puoi installare solo il driver in un progetto Node.js ed eseguire brevi esempi di codice.

- Per installare il driver QLDB e AWS l'SDK JavaScript per Node.js in un progetto esistente, procedi a. Installazione
- Per configurare un progetto ed eseguire brevi esempi di codice che illustrano le transazioni di dati di base su un registro, consulta la. Guida di avvio rapido
- Per eseguire esempi più approfonditi delle operazioni delle API relative ai dati e alla gestione nell'applicazione di esempio completa del tutorial, consulta il. Tutorial su Node.js

Installazione

QLDB supporta le seguenti versioni dei driver e le relative dipendenze da Node.js.

Versione driver	Node.js version (Versione Node.js)	Stato	Data di rilascio più recente
1.x	10.x o versione successiva	Versione di produzione	5 giugno 2020
2.x	10.x o versione successiva	Versione di produzione	6 maggio 2021
3.x	14.x o versione successiva	Versione di produzione	10 novembre 2023

Per installare il driver QLDB usando npm (il gestore di pacchetti Node.js), inserisci il seguente comando dalla directory principale del tuo progetto.

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
```

Il driver ha dipendenze peer-to-peer dai seguenti pacchetti. È inoltre necessario installare questi pacchetti come dipendenze nel progetto.

3.x

Client QLDB aggregato modulare (API di gestione)

```
npm install @aws-sdk/client-qldb
```

Client di sessione QLDB aggregato modulare (API dati)

```
npm install @aws-sdk/client-qldb-session
```

Formato dati Amazon Ion

```
npm install ion-js
```

JavaScript Implementazione pura di `BigInt`

```
npm install jsbi
```

2.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Formato dati Amazon Ion

```
npm install ion-js@4.0.0
```

JavaScript Implementazione pura di `BigInt`

```
npm install jsbi@3.1.1
```

1.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Formato dati Amazon Ion

```
npm install ion-js@4.0.0
```

JavaScript Implementazione pura di `BigInt`

```
npm install jsbi@3.1.1
```

Utilizzo del driver per connettersi a un registro

Quindi puoi importare il driver e usarlo per connetterti a un libro mastro. Il seguente esempio di TypeScript codice mostra come creare un'istanza di driver per un nome di registro specificato e. Regione 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);
});
```

Per brevi esempi di codice su come eseguire transazioni di dati di base su un registro, vedere il. Riferimento del libro di cucina

Consigli di configurazione

Riutilizzo delle connessioni con keep-alive

Driver QLDB Node.js versione 3

L'agente HTTP/HTTPS Node.js predefinito crea una nuova connessione TCP per ogni nuova richiesta. Per evitare il costo di stabilire una nuova connessione, la versione AWS SDK for JavaScript v3 riutilizza le connessioni TCP per impostazione predefinita. Per ulteriori informazioni e per scoprire come disabilitare il riutilizzo delle connessioni, consulta Riutilizzo delle connessioni con keep-alive in Node.js nella Guida per gli sviluppatori. AWS SDK for JavaScript

Si consiglia di utilizzare l'impostazione predefinita per riutilizzare le connessioni nel driver QLDB per Node.js. Durante l'inizializzazione del driver, imposta l'opzione HTTP del client di basso livello sullo stesso valore `maxSockets` per cui hai impostato. `maxConcurrentTransactions`

Ad esempio, vedete quanto segue JavaScript o TypeScript codice.

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);
```

Driver QLDB Node.js versione 2

L'agente HTTP/HTTPS Node.js predefinito crea una nuova connessione TCP per ogni nuova richiesta. Per evitare il costo di stabilire una nuova connessione, si consiglia di riutilizzare una connessione esistente.

Per riutilizzare le connessioni nel driver QLDB per Node.js, utilizzare una delle seguenti opzioni:

- Durante l'inizializzazione del driver, imposta le seguenti opzioni HTTP del client di basso livello:
  - `keepAlive` – `true`
  - `maxSockets`— Lo stesso valore che hai impostato per `maxConcurrentTransactions`

  Ad esempio, vedere quanto segue JavaScript o TypeScript codice.

  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);
```

- In alternativa, è possibile impostare la variabile di `AWS_NODEJS_CONNECTION_REUSE_ENABLED` ambiente su1. Per ulteriori informazioni, consulta Riutilizzo delle connessioni con Keep-Alive in Node.js nella Guida per gli AWS SDK for JavaScript sviluppatori.

> **Note**
>
> Se imposti questa variabile di ambiente, influirà su tutti quelli Servizi AWS che utilizzano il. AWS SDK for JavaScript

Driver Amazon QLDB per Node.js — Tutorial di avvio rapido

In questo tutorial, imparerai come configurare un'applicazione semplice utilizzando il driver Amazon QLDB per Node.js. Questa guida include passaggi per l'installazione del driver JavaScript TypeScript ed esempi di creazione, lettura, aggiornamento ed eliminazione CRUD. Per esempi più approfonditi che dimostrano queste operazioni in un'applicazione di esempio completa, consulta ilTutorial su Node.js.

> **Note**
>
> Ove applicabile, alcuni passaggi hanno esempi di codice diversi per ciascuna versione principale supportata del driver QLDB per Node.js.

Argomenti

- Prerequisiti

Prerequisiti

Prima di iniziare, eseguire la seguente procedura:

1. Completa il driverPrerequisiti per Node.js, se non l'hai già fatto. Ciò include la registrazioneAWS, la concessione dell'accesso programmatico per lo sviluppo e l'installazione di Node.js.
2. Crea un libro contabile denominato`quick-start`.

   Per informazioni su come creare un libro mastro, consultaOperazioni di base per i libri mastri Amazon QLDB oFase 1: creazione di un nuovo libro mastro in Guida introduttiva alla console.

Se si utilizza TypeScript, è necessario eseguire anche i seguenti passaggi di configurazione.

Usando TypeScript

Da installare TypeScript

1. Installare il TypeScript pacchetto. Il driver QLDB viene eseguito su TypeScript 3.8.x.

   ```
   $ npm install --global typescript@3.8.0
   ```

2. Dopo aver installato il pacchetto, eseguire il seguente comando per assicurarsi che il TypeScript compilatore sia installato.

   ```
   $ tsc --version
   ```

Per eseguire il codice nei passaggi seguenti, tieni presente che devi prima traspilare il TypeScript file in JavaScript codice eseguibile, come segue.

```
$ tsc app.ts; node app.js
```

Fase 1: Configurazione del progetto

In primo luogo, configura il tuo progetto Node.js.

1. Crea una cartella per la tua applicazione.

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

2. Per inizializzare il progetto, inserisci il seguentenpm comando e rispondi alle domande che vengono poste durante la configurazione. Puoi usare i valori predefiniti per la maggior parte delle domande.

   ```
   $ npm init
   ```

3. Installa il driver Amazon QLDB per Node.js.
   - Utilizzo della versione 3.x

     ```
     $ npm install amazon-qldb-driver-nodejs --save
     ```

   - Utilizzo della versione 2.x

     ```
     $ npm install amazon-qldb-driver-nodejs@2.2.0 --save
     ```

   - Utilizzo della versione 1.x

     ```
     $ npm install amazon-qldb-driver-nodejs@1.0.0 --save
     ```

4. Installa le dipendenze peer del driver.
   - Utilizzo della versione 3.x

     ```
     $ npm install @aws-sdk/client-qldb-session --save
     $ npm install ion-js --save
     $ npm install jsbi --save
     ```

   - Utilizzo della versione 2.x o 1.x

     ```
     $ npm install aws-sdk --save
     ```

```
$ npm install ion-js@4.0.0 --save
$ npm install jsbi@3.1.1 --save
```

5. Crea un nuovo file denominato`app.js` per JavaScript, o`app.ts` per TypeScript.

   Quindi, aggiungi in modo incrementale gli esempi di codice nei passaggi seguenti per provare alcune operazioni CRUD di base. In alternativa, puoi saltare il step-by-step tutorial ed eseguire invece l'applicazione completa.

Fase 2: Inizializzazione del driver

Inizializza un'istanza del driver che si connette al registro denominato`quick-start`. Aggiungi il seguente codice al tuo`app.ts` file`app.js` or.

Utilizzo della versione 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**
>
> - In questo esempio di codice, sostituisci *us-east-1* con ilRegione AWS luogo in cui hai creato il tuo libro mastro.
> - Per semplicità, gli altri esempi di codice in questa guida utilizzano un driver con impostazioni predefinite, come specificato nell'esempio seguente per la versione 1.x. Puoi anche utilizzare la tua istanza del driver con una`RetryConfig` personalizzata.

Utilizzo della versione 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

- In questo esempio di codice, sostituisci *us-east-1* con ilRegione AWS luogo in cui hai creato il tuo libro mastro.
- La versione 2.x introduce il nuovo parametro opzionale`RetryConfig` per l'inizializzazione`QldbDriver`.
- Per semplicità, gli altri esempi di codice in questa guida utilizzano un driver con impostazioni predefinite, come specificato nell'esempio seguente per la versione 1.x. Puoi anche utilizzare la tua istanza del driver con una`RetryConfig` personalizzata.

- Questo esempio di codice inizializza un driver che riutilizza le connessioni esistenti impostando le opzioni keep-alive. Per ulteriori informazioni, consultaConsigli di configurazione il driver Node.js.

Utilizzo della versione 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**
>
> È possibile impostare la variabile di`AWS_REGION` ambiente per specificare la regione. Per ulteriori informazioni, consulta Impostare ilRegione AWS nella Guida per gliAWS SDK for JavaScript sviluppatori.

Fase 3: Creare una tabella e un indice

Gli esempi di codice seguenti mostrano come eseguire`CREATE INDEX` istruzioni`CREATE TABLE` e.

1. Aggiungete la seguente funzione che crea una tabella denominata`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. Aggiungi la seguente funzione che crea un indice per il`firstName` campo della`People` tabella. Gli indici sono necessari per ottimizzare le prestazioni delle query e contribuire a limitare le eccezioni di conflitto OCC (Optimistic Concurrency Control).

   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. Nella`main` funzione, prima`createTable` chiami e poi chiami`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. Esegui il codice per creare la tabella e l'indice.

JavaScript

```
$ node app.js
```

TypeScript

```
$ tsc app.ts; node app.js
```

Fase 4: Inserimento di un documento

L'esempio di codice seguente mostra come eseguire un'`INSERT`istruzione. QLDB supporta il linguaggio di interrogazione PartiQL (compatibile con SQL) e il formato dati Amazon Ion (superset di JSON).

1. Aggiungete la seguente funzione che inserisce un documento nella`People` tabella.

   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);
   }
   ```

   Questo esempio utilizza un punto interrogativo (?) come segnaposto variabile per passare le informazioni del documento alla dichiarazione. Il`execute` metodo supporta valori sia nei tipi Amazon Ion che nei tipi nativi Node.js.

> **Tip**
>
> Per inserire più documenti utilizzando una singolaINSERT istruzione, è possibile passare un elenco di parametri di tipo all'istruzione come segue.
>
> ```
> // people is a list
> txn.execute("INSERT INTO People ?", people);
> ```
>
> Non si racchiude la variabile placeholder (?) tra parentesi a doppio angolo (<<...>>) quando si passa un elenco. Nelle istruzioni PartiQL, le parentesi a doppio angolo indicano una raccolta non ordinata nota come borsa.

2. Nella`main` funzione, rimuovi`createIndex` le chiamate`createTable` e e aggiungi una chiamata a`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();
}
```

Fase 5: esegui query sul documento

L'esempio di codice seguente mostra come eseguire un'`SELECT`istruzione.

1. Aggiungete la seguente funzione che interroga un documento dalla`People` tabella.

   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. Nella`main` funzione, aggiungi la seguente chiamata a`fetchDocuments` dopo la chiamata a`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();
}
```

Fase 6: Aggiornamento del documento

L'esempio di codice seguente mostra come eseguire un'UPDATEistruzione.

1. Aggiungete la seguente funzione che aggiorna un documento nellaPeople tabella passandolastName a"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. Nellamain funzione, aggiungi la seguente chiamata aupdateDocuments dopo la chiamata afetchDocuments. Quindi, chiamafetchDocuments di nuovo per vedere i risultati aggiornati.

   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. Esegui il codice per inserire, interrogare e aggiornare un documento.

JavaScript

```
$ node app.js
```

TypeScript

```
$ tsc app.ts; node app.js
```

Esecuzione dell'applicazione completa

I seguenti esempi di codice sono le versioni complete di`app.js` e`app.ts`. Invece di eseguire i passaggi precedenti singolarmente, puoi anche eseguire questo codice dall'inizio alla fine. Questa applicazione dimostra alcune operazioni CRUD di base sul registro denominato`quick-start`.

> **Note**
>
> Prima di eseguire questo codice, assicuratevi di non avere già una tabella attiva denominata`People` nel`quick-start` libro mastro.

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();
}
```

Per eseguire l'applicazione completa, immetti il seguente comando.

JavaScript

```
$ node app.js
```

TypeScript

```
$ tsc app.ts; node app.js
```

Driver Amazon QLDB per Node.js — Riferimento al ricettario

Questa guida di riferimento mostra i casi d'uso più comuni del driver Amazon QLDB per Node.js. JavaScript Fornisce esempi di TypeScript codice che mostrano come utilizzare il driver per eseguire operazioni di creazione, lettura, aggiornamento ed eliminazione (CRUD). Include anche esempi di codice per l'elaborazione dei dati di Amazon Ion. Inoltre, questa guida evidenzia le migliori pratiche per rendere le transazioni idempotenti e implementare i vincoli di unicità.

Indice

Importazione del driver

Il seguente esempio di codice importa il driver.

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**
>
> Questo esempio importa anche il pacchetto Amazon Ion (`ion-js`). È necessario questo pacchetto per elaborare i dati Ion quando si eseguono alcune operazioni sui dati in questo riferimento. Per ulteriori informazioni, consulta Come lavorare con Amazon Ion.

Istanziazione del driver

L'esempio di codice seguente crea un'istanza del driver che si connette a un nome di registro specificato utilizzando le impostazioni predefinite.

JavaScript

```
const qldbDriver = new qldb.QldbDriver("vehicle-registration");
```

TypeScript

```
const qldbDriver: QldbDriver = new QldbDriver("vehicle-registration");
```

Operazioni CRUD

QLDB esegue operazioni di creazione, lettura, aggiornamento ed eliminazione (CRUD) come parte di una transazione.

> **Warning**
>
> Come best practice, cerca di rendere le tue transazioni di scrittura rigorosamente idempotenti.

Rendere le transazioni idempotenti

Ti consigliamo di effettuare transazioni di scrittura idempotenti per evitare effetti collaterali imprevisti in caso di nuovi tentativi. Una transazione è idempotente se può essere eseguita più volte e produrre risultati identici ogni volta.

Ad esempio, considera una transazione che inserisce un documento in una tabella denominata`Person`. La transazione deve innanzitutto verificare se il documento esiste già o meno nella tabella. Senza questo controllo, la tabella potrebbe contenere documenti duplicati.

Supponiamo che QLDB esegua correttamente la transazione sul lato server, ma che il client scada in attesa di una risposta. Se la transazione non è idempotente, lo stesso documento potrebbe essere inserito più di una volta in caso di nuovo tentativo.

Utilizzo degli indici per evitare la scansione completa delle tabelle

Si consiglia inoltre di eseguire istruzioni con una clausola`WHERE` predicativa utilizzando un operatore di uguaglianza su un campo indicizzato o un ID di documento, ad esempio`WHERE indexedField = 123` o`WHERE indexedField IN (456, 789)`. Senza questa ricerca indicizzata, QLDB deve eseguire una scansione della tabella, che può portare a timeout delle transazioni o conflitti OCC (Optimistic Concurrency Control).

Per ulteriori informazioni sull'OCC, consultaModello di concorrenza Amazon QLDB.

Transazioni create implicitamente

Il metodo QldbDriver.executeLambda accetta una funzione lambda che riceve un'istanza di TransactionExecutor, che è possibile utilizzare per eseguire istruzioni. L'istanza di`TransactionExecutor` avvolge una transazione creata implicitamente.

È possibile eseguire istruzioni all'interno della funzione lambda utilizzando il metodo execute dell'esecutore della transazione. Il driver esegue implicitamente la transazione quando la funzione lambda ritorna.

> **Note**
>
> Il`execute` metodo supporta sia i tipi Amazon Ion che i tipi nativi Node.js. Se si passa un tipo nativo di Node.js come argomento`execute`, il driver lo converte in un tipo Ion utilizzando il`ion-js` pacchetto (a condizione che sia supportata la conversione per il tipo di dati Node.js specificato). Per i tipi di dati e le regole di conversione supportati, consulta il README JavaScript DOM di Ion.

Le sezioni seguenti mostrano come eseguire operazioni CRUD di base, specificare una logica di ripetizione personalizzata e implementare vincoli di unicità.

Indice

Creazione di tabelle

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');
    });
}());
```

Creazione di indici

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)');
    });
}());
```

Leggere documenti

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']
        }
    });
}());
```

Utilizzo dei parametri della query

L'esempio di codice seguente utilizza un parametro di interrogazione di tipo nativo.

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'esempio di codice seguente utilizza un parametro di interrogazione di tipo 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']
        }
    });
}());
```

Il seguente esempio di codice utilizza più parametri di interrogazione.

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'esempio di codice seguente utilizza un elenco di parametri di interrogazione.

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**
>
> Quando si esegue una query senza una ricerca indicizzata, viene richiamata una scansione completa della tabella. In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le query possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Inserimento documento

Il seguente esempio di codice inserisce tipi di dati nativi.

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);
        }
    });
}());
```

Il seguente esempio di codice inserisce i tipi di dati 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);
        }
    });
}());
```

Questa transazione inserisce un documento nella`Person` tabella. Prima dell'inserimento, controlla innanzitutto se il documento esiste già nella tabella. Questo controllo rende la transazione di natura idempotente. Anche se esegui questa transazione più volte, non causerà effetti collaterali indesiderati.

> **Note**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Inserimento di più documenti in un'unica dichiarazione

Per inserire più documenti utilizzando una singolaINSERT istruzione, è possibile passare un elenco di parametri di tipo all'istruzione come segue.

```
// people is a list
txn.execute("INSERT INTO People ?", people);
```

Non si racchiude la variabile placeholder (?) tra parentesi a doppio angolo (`<<...>>`) quando si passa un elenco. Nelle istruzioni PartiQL, le parentesi a doppio angolo indicano una raccolta non ordinata nota come borsa.

Aggiornamento dei documenti

Il seguente esempio di codice utilizza tipi di dati nativi.

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');
    });
}());
```

Il seguente esempio di codice utilizza i tipi di dati 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**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Eliminazione di documenti

Il seguente esempio di codice utilizza tipi di dati nativi.

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');
    });
}());
```

Il seguente esempio di codice utilizza i tipi di dati 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**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Esecuzione di più rendiconti in una transazione

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;
    });
};
```

Logica di ripetizione dei tentativi

Il`executeLambda` metodo del driver dispone di un meccanismo di ripetizione integrato che riprova la transazione se si verifica un'eccezione ripetibile (ad esempio timeout o conflitti OCC). Il numero massimo di tentativi di ripetizione e la strategia di backoff sono configurabili.

Il limite di tentativi predefinito è`4` e la strategia di backoff predefinita è [defaultBackoffFunction](#)basata su`10` millisecondi. È possibile impostare la configurazione dei tentativi per istanza del driver e anche per transazione utilizzando un'istanza di [RetryConfig](#).

L'esempio di codice seguente specifica la logica dei tentativi con un limite di tentativi personalizzato e una strategia di backoff personalizzata per un'istanza del driver.

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'esempio di codice seguente specifica la logica dei tentativi con un limite di tentativi personalizzato e una strategia di backoff personalizzata per una particolare esecuzione lambda. Questa configurazione per`executeLambda` sovrascrive la logica dei tentativi impostata per l'istanza del driver.

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);
}());
```

Implementazione dei vincoli di unicità

QLDB non supporta indici univoci, ma puoi implementare questo comportamento nella tua applicazione.

Supponiamo di voler implementare un vincolo di unicità sul`GovId` campo della`Person` tabella. Questa operazione può essere eseguita nel seguente modo, è possibile scrivere una transazione che esegue le seguenti operazioni:

1. Asserisci che la tabella non ha documenti esistenti con un valore specificato`GovId`.
2. Inserisci il documento se l'asserzione è valida.

Se una transazione concorrente supera contemporaneamente l'asserzione, solo una delle transazioni verrà confermata con successo. L'altra transazione avrà esito negativo con un'eccezione di conflitto OCC.

L'esempio di codice seguente mostra come implementare questa logica di vincolo di unicità.

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**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Come lavorare con Amazon Ion

Le sezioni seguenti mostrano come utilizzare il modulo Amazon Ion per elaborare i dati Ion.

Indice

- Ottenere un dump di testo Ion

Importazione del modulo Ion

JavaScript

```
var ionjs = require('ion-js');
```

TypeScript

```
import { dom, dumpBinary, dumpText, load } from "ion-js";
```

Creazione di tipi di ioni

Il seguente esempio di codice crea un oggetto Ion dal testo 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'esempio di codice seguente crea un oggetto Ion da un dizionario 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']
```

Ottenere un dump binario 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
```

Ottenere un dump di testo 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"}
```

Per ulteriori informazioni su Ion, consulta la documentazione di Amazon Ion su GitHub. Per altri esempi di codice sull'utilizzo di Ion in QLDB, vedereUtilizzo dei tipi di dati Amazon Ion in Amazon QLDB.

Driver Amazon QLDB per Python

Per lavorare con i dati contenuti nel tuo registro, puoi connetterti ad Amazon QLDB dalla tua applicazione Python utilizzando un driverAWS fornito. Negli argomenti seguenti viene descritto come iniziare con il driver QLDB.

Argomenti

- Risorse per i conducenti
- Prerequisiti
- Installazione
- Driver Amazon QLDB per Python — Tutorial di avvio rapido
- Driver Amazon QLDB per Python — riferimento al ricettario

Risorse per i conducenti

Per ulteriori informazioni sulla funzionalità supportata dal driver Python, vedere le risorse seguenti:

- Riferimento API: 3.x, 2.x
- Codice sorgente del driver (GitHub)
- Codice sorgente dell'applicazione di esempio (GitHub)
- Esempi di codice Amazon Ion

Prerequisiti

Per poter utilizzare il driver QLDB per Python, bisogna eseguire le seguenti operazioni:

1. Segui le istruzioniAWS di configurazione inAccesso ad Amazon QLDB. Questo include gli output seguenti:

1. Per registrarti e ottenere un accountAWS.
2. Crea un utente con le autorizzazioni QLDB.
3. Concessione dell'accesso programmatico per lo sviluppo.

2. Installa una delle seguenti versioni di Python dal sito di download di Python:
   - 3.6 o successivo — driver QLDB per Python v3
   - 3.4 o successivo — driver QLDB per Python v2
3. ImpostaAWS le tue credenziali e quelle predefiniteRegione AWS. Per istruzioni, consulta Quickstart nellaAWS SDK for Python (Boto3) documentazione.

   Per un elenco completo delle regioni disponibili, consulta gli endpoint e le quote di Amazon QLDB nel Riferimenti generali di AWS.

Successivamente, puoi scaricare l'applicazione di esempio del tutorial completo oppure puoi installare solo il driver in un progetto Python ed eseguire brevi esempi di codice.

- Per installare il driver QLDB e il driverAWS SDK for Python (Boto3) in un progetto esistente, procedere aInstallazione.
- Per configurare un progetto ed eseguire brevi esempi di codice che dimostrino le transazioni di dati di base su un libro mastro, consulta ilGuida di avvio rapido.
- Per eseguire esempi più approfonditi di operazioni relative ai dati e alle API di gestione nell'applicazione di esempio del tutorial completo, consulta ilTutorial di Python.

Installazione

QLDB supporta le seguenti versioni di driver e le relative dipendenze da Python.

Versione driver	Versione di Python	Stato	Data di rilascio:
2.x	3.4 o versione successiva	Versione di produzione	7 maggio 2020
3. x	3.6 o versioni successive	Versione di produzione	28 ottobre 2021

Per installare il driver QLDB da PyPI utilizzando`pip` (un gestore di pacchetti per Python), inserisci quanto segue nella riga di comando.

3.x

```
pip install pyqldb
```

2.x

```
pip install pyqldb==2.0.2
```

L'installazione del driver installa anche le relative dipendenze, inclusi i pacchetti Amazon Ion AWS SDK for Python (Boto3)e Amazon.

Utilizzo del driver per connettersi a un registro

Quindi puoi importare il driver e utilizzarlo per connetterti a un libro mastro. Negli esempi di codice Python riportati come creare una sessione per un nome di registro specificato.

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)
```

Per brevi esempi di codice su come eseguire transazioni di dati di base su un libro mastro, vedere ilRiferimento del libro di cucina.

Driver Amazon QLDB per Python — Tutorial di avvio rapido

In questo tutorial, imparerai come configurare una semplice applicazione utilizzando l'ultima versione del driver Amazon QLDB per Python. Questa guida include i passaggi per l'installazione del driver ed esempi di creazione, lettura, aggiornamento ed eliminazione (CRUD). Per esempi più approfonditi che dimostrano queste operazioni in un'applicazione di esempio completa, consulta ilTutorial di Python.

Argomenti

Prerequisiti

Prima di iniziare, assicurati di completare quanto segue:

1. Completa il driverPrerequisiti for the Python, se non l'hai già fatto. Ciò include la registrazioneAWS, la concessione dell'accesso programmatico per lo sviluppo e l'installazione di Python versione 3.6 o successiva.
2. Crea un libro contabile denominato`quick-start`.

   Per informazioni su come creare un libro mastro, consultaOperazioni di base per i libri mastri Amazon QLDB oFase 1: creazione di un nuovo libro mastro in Guida introduttiva alla console.

Fase 1: Configurazione del progetto

In primo luogo, configura il tuo progetto Python.

> **Note**
>
> Se utilizzi un IDE con funzionalità per automatizzare questi passaggi di configurazione, puoi passare direttamente aFase 2: Inizializzazione del driver.

1. Crea una cartella per la tua applicazione.

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

2. Per installare il driver QLDB per Python da PyPI, immettere il seguentepip comando.

   ```
   $ pip install pyqldb
   ```

   L'installazione del driver installa anche le relative dipendenze, inclusi i pacchetti Amazon Ion AWS SDK for Python (Boto3)e Amazon.

3. Creare un nuovo file denominato `app.py`.

   Quindi, aggiungi in modo incrementale gli esempi di codice nei passaggi seguenti per provare alcune operazioni CRUD di base. In alternativa, puoi saltare il step-by-step tutorial ed eseguire invece l'applicazione completa.

Fase 2: Inizializzazione del driver

Inizializza un'istanza del driver che si connette al registro denominato`quick-start`. Aggiungi il seguente codice al`app.py` file.

```
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)
```

Fase 3: Crea una tabella e un indice

L'esempio di codice seguente mostra come eseguire`CREATE TABLE` le`CREATE INDEX` istruzioni.

Aggiungi il codice seguente per creare una tabella denominata`People` e un indice per il`lastName` campo di quella tabella. Gli indici sono necessari per ottimizzare le prestazioni delle query e contribuire a limitare le eccezioni di conflitto OCC (Optimistic Concurrency Control).

```
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))
```

Fase 4: inserisci un documento

L'esempio di codice seguente mostra come eseguire un'`INSERT`istruzione. QLDB supporta il linguaggio di interrogazione PartiQL (compatibile con SQL) e il formato dati Amazon Ion (superset di JSON).

Aggiungete il seguente codice che inserisce un documento nella`People` tabella.

```
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))
```

Questo esempio utilizza un punto interrogativo (?) come segnaposto variabile per passare le informazioni del documento alla dichiarazione. Il`execute_statement` metodo supporta valori sia nei tipi Amazon Ion che nei tipi nativi di Python.

> **Tip**
>
> Per inserire più documenti utilizzando una singolaINSERT istruzione, è possibile passare un elenco di parametri di tipo all'istruzione come segue.
>
> ```
> # people is a list
> transaction_executor.execute_statement("INSERT INTO Person ?", people)
> ```
>
> Non si racchiude la variabile placeholder (?) tra parentesi a doppio angolo (`<<...>>`) quando si passa un elenco. Nelle istruzioni PartiQL, le parentesi a doppio angolo indicano una raccolta non ordinata nota come borsa.

Fase 5: esegui query sul documento

L'esempio di codice seguente mostra come eseguire un'`SELECT`istruzione.

Aggiungete il codice seguente per interrogare un documento dalla`People` tabella.

```
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))
```

Fase 6: Aggiornamento del documento

L'esempio di codice seguente mostra come eseguire un'`UPDATE`istruzione.

1. Aggiungi il codice seguente che aggiorna un documento nella`People` tabella eseguendo l'aggiornamento`age` a`42`.

```
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. Interroga nuovamente la tabella per visualizzare il valore aggiornato.

```
# Query the updated document
qldb_driver.execute_lambda(lambda executor: read_documents(executor))
```

3. Per eseguire l'applicazione, inserisci il comando seguente dalla directory del progetto.

```
$ python app.py
```

Esecuzione dell'applicazione completa

Il seguente esempio di codice è la versione completa dell'`app.py`applicazione. Invece di eseguire i passaggi precedenti singolarmente, puoi anche copiare ed eseguire questo esempio di codice dall'inizio alla fine. Questa applicazione dimostra alcune operazioni CRUD di base sul registro denominato`quick-start`.

> **Note**
>
> Prima di eseguire questo codice, assicuratevi di non avere già una tabella attiva denominata`People` nel`quick-start` libro mastro.

```
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))
```

Per eseguire l'applicazione completa, inserisci il seguente comando dalla directory del progetto.

```
$ python app.py
```

Driver Amazon QLDB per Python — riferimento al ricettario

Questa guida di riferimento mostra i casi d'uso più comuni del driver Amazon QLDB per Python. Fornisce esempi di codice Python che mostrano come utilizzare il driver per eseguire operazioni di creazione, lettura, aggiornamento ed eliminazione (CRUD). Include anche esempi di codice per l'elaborazione dei dati di Amazon Ion. Inoltre, questa guida evidenzia le migliori pratiche per rendere le transazioni idempotenti e implementare i vincoli di unicità.

> **Note**
>
> Ove applicabile, alcuni casi d'uso hanno esempi di codice diversi per ciascuna versione principale supportata del driver QLDB per Python.

Indice

Importazione del driver

Il seguente esempio di codice importa il driver.

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**
>
> Questo esempio importa anche il pacchetto Amazon Ion (`amazon.ion.simpleion`). È necessario questo pacchetto per elaborare i dati Ion quando si eseguono alcune operazioni sui dati in questo riferimento. Per ulteriori informazioni, consulta Uso di Amazon Ion.

Istantizzazione del driver

L'esempio di codice seguente crea un'istanza del driver che si connette a un nome di registro specificato utilizzando le impostazioni predefinite.

3.x

```
qldb_driver = QldbDriver(ledger_name='vehicle-registration')
```

2.x

```
qldb_driver = PooledQldbDriver(ledger_name='vehicle-registration')
```

Operazioni CRUD

QLDB esegue operazioni di creazione, lettura, aggiornamento ed eliminazione (CRUD) come parte di una transazione.

> **Warning**
>
> Come best practice, cerca di rendere le tue transazioni di scrittura rigorosamente idempotenti.

Rendere le transazioni idempotenti

Ti consigliamo di effettuare transazioni di scrittura idempotenti per evitare effetti collaterali imprevisti in caso di nuovi tentativi. Una transazione è idempotente se può essere eseguita più volte e produrre risultati identici ogni volta.

Ad esempio, si consideri una transazione che inserisce un documento in una tabella denominata`Person`. La transazione deve innanzitutto verificare se il documento esiste già o meno nella tabella. Senza questo controllo, la tabella potrebbe contenere documenti duplicati.

Supponiamo che QLDB esegua correttamente la transazione sul lato server, ma che il client scada in attesa di una risposta. Se la transazione non è idempotente, lo stesso documento potrebbe essere inserito più di una volta in caso di nuovo tentativo.

Utilizzo degli indici per evitare la scansione completa delle tabelle

Si consiglia inoltre di eseguire istruzioni con una clausola`WHERE` predicativa utilizzando un operatore di uguaglianza su un campo indicizzato o un ID di documento, ad esempio`WHERE indexedField = 123` o`WHERE indexedField IN (456, 789)`. Senza questa ricerca indicizzata, QLDB deve eseguire una scansione della tabella, che può portare a timeout delle transazioni o conflitti OCC (Optimistic Concurrency Control).

Per ulteriori informazioni su OCC, consultaModello di concorrenza Amazon QLDB.

Transazioni create implicitamente

Il metodo pyqldb.driver.qldb_driver.execute_lambda accetta una funzione lambda che riceve un'istanza di pyQLDB.EXECUTION.EXECUTOR.EXECUTOR, che è possibile utilizzare per eseguire istruzioni. L'istanza di`Executor` avvolge una transazione creata implicitamente.

È possibile eseguire istruzioni all'interno della funzione lambda utilizzando il metodo execute_statement dell'esecutore della transazione. Il driver esegue implicitamente la transazione quando la funzione lambda ritorna.

> **Note**
>
> Il`execute_statement` metodo supporta sia i tipi Amazon Ion che i tipi nativi di Python. Se si passa un tipo nativo di Python come argomento`execute_statement`, il driver lo converte in un tipo Ion utilizzando il`amazon.ion.simpleion` modulo (a condizione che sia supportata la conversione per il tipo di dati Python specificato). Per i tipi di dati e le regole di conversione supportati, consulta il codice sorgente simpleion.

Le sezioni seguenti mostrano come eseguire operazioni CRUD di base, specificare una logica di ripetizione personalizzata e implementare vincoli di unicità.

Indice

Creazione di tabelle

```
def create_table(transaction_executor):
    transaction_executor.execute_statement("CREATE TABLE Person")

qldb_driver.execute_lambda(lambda executor: create_table(executor))
```

Creazione di indici

```
def create_index(transaction_executor):
    transaction_executor.execute_statement("CREATE INDEX ON Person(GovId)")

qldb_driver.execute_lambda(lambda executor: create_index(executor))
```

Lettura di documenti

```
# 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))
```

Utilizzo dei parametri della query

L'esempio di codice seguente utilizza un parametro di interrogazione di tipo nativo.

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId = ?",
 'TOYENC486FH')
```

L'esempio di codice seguente utilizza un parametro di interrogazione di tipo Ion.

```
name = ion.loads('Brent')
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE FirstName
 = ?", name)
```

Il seguente esempio di codice utilizza più parametri di interrogazione.

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId = ?
 AND FirstName = ?", 'TOYENC486FH', "Brent")
```

L'esempio di codice seguente utilizza un elenco di parametri di interrogazione.

```
gov_ids = ['TOYENC486FH','ROEE1','YH844']
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId IN
 (?,?,?)", *gov_ids)
```

> **Note**
>
> Quando si esegue una query senza una ricerca indicizzata, viene richiamata una scansione completa della tabella. In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le query possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Inserimento dei documenti

Il seguente esempio di codice inserisce tipi di dati nativi.

```
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))
```

Il seguente esempio di codice inserisce i tipi di dati 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))
```

Questa transazione inserisce un documento nella`Person` tabella. Prima dell'inserimento, controlla innanzitutto se il documento esiste già nella tabella. Questo controllo rende la transazione di natura idempotente. Anche se esegui questa transazione più volte, non causerà effetti collaterali indesiderati.

> **Note**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Inserimento di più documenti in un'unica dichiarazione

Per inserire più documenti utilizzando una singolaINSERT istruzione, è possibile passare un elenco di parametri di tipo all'istruzione come segue.

```
# people is a list
transaction_executor.execute_statement("INSERT INTO Person ?", people)
```

Non si racchiude la variabile placeholder (?) tra parentesi a doppio angolo (`<<...>>`) quando si passa un elenco. Nelle istruzioni PartiQL, le parentesi a doppio angolo indicano una raccolta non ordinata nota come borsa.

Aggiornamento dei documenti

Il seguente esempio di codice utilizza tipi di dati nativi.

```
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))
```

Il seguente esempio di codice utilizza i tipi di dati 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**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Eliminazione di documenti

Il seguente esempio di codice utilizza tipi di dati nativi.

```
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))
```

Il seguente esempio di codice utilizza i tipi di dati 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**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Esecuzione di più rendiconti in una transazione

```
# 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))
```

Logica di ripetizione dei tentativi

Il`execute_lambda` metodo del driver dispone di un meccanismo di ripetizione integrato che riprova la transazione se si verifica un'eccezione ripetibile (ad esempio timeout o conflitti OCC).

3.x

Il numero massimo di tentativi di ripetizione e la strategia di backoff sono configurabili.

Il limite di tentativi predefinito è`4` e la strategia di backoff predefinita è Backoff esponenziale e jitter con una base di`10` millisecondi. È possibile impostare la configurazione dei tentativi per istanza del driver e anche per transazione utilizzando un'istanza di pyqldb.config.retry_config. RetryConfig.

L'esempio di codice seguente specifica la logica dei tentativi con un limite di tentativi personalizzato e una strategia di backoff personalizzata per un'istanza del driver.

```
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'esempio di codice seguente specifica la logica dei tentativi con un limite di tentativi personalizzato e una strategia di backoff personalizzata per una particolare esecuzione lambda. Questa configurazione per`execute_lambda` sovrascrive la logica dei tentativi impostata per l'istanza del driver.

```
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

Il numero massimo di tentativi è configurabile. È possibile configurare il limite di tentativi impostando la`retry_limit` proprietà durante l'inizializzazione`PooledQldbDriver`.

Il limite di tentativi predefinito è`4`.

Implementazione dei vincoli di unicità

QLDB non supporta indici univoci, ma puoi implementare questo comportamento nella tua applicazione.

Si supponga di voler implementare un vincolo di unicità sul`GovId` campo della`Person` tabella. Per eseguire questa operazione, è possibile scrivere una transazione che esegue le seguenti operazioni:

1. Asserisci che la tabella non ha documenti esistenti con un valore specificato`GovId`.
2. Inserisci il documento se l'asserzione è valida.

Se una transazione concorrente supera contemporaneamente l'asserzione, solo una delle transazioni verrà confermata con successo. L'altra transazione avrà esito negativo con un'eccezione di conflitto OCC.

L'esempio di codice seguente mostra come implementare questa logica di vincolo di unicità.

```
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**
>
> In questo esempio, consigliamo di avere un indice sul`GovId` campo per ottimizzare le prestazioni. Senza un indice attivo`GovId`, le dichiarazioni possono avere una maggiore latenza e possono anche portare a eccezioni di conflitto OCC o a timeout delle transazioni.

Uso di Amazon Ion

Le sezioni seguenti mostrano come utilizzare il modulo Amazon Ion per elaborare i dati Ion.

Indice

Importazione del modulo Ion

```
import amazon.ion.simpleion as simpleion
```

Creazione di tipi di ioni

Il seguente esempio di codice crea un oggetto Ion dal testo 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
```

Il seguente esempio di codice crea un oggetto Ion da 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
```

Ottenere un dump binario 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'
```

Ottenere un dump di testo Ion

```
# ion_obj is an Ion struct
print(simpleion.dumps(ion_obj, binary=False)) # prints $ion_1_0
 {GovId:'TOYENC486FH',FirstName:"Brent"}
```

Per ulteriori informazioni sull'utilizzo di Ion, consulta la documentazione di Amazon Ion su GitHub. Per altri esempi di codice sull'utilizzo di Ion in QLDB, vedereUtilizzo dei tipi di dati Amazon Ion in Amazon QLDB.

Comprensione della gestione delle sessioni con il driver in Amazon QLDB

Se hai esperienza nell'uso di un sistema di gestione di database relazionali (RDBMS, Relational Database Management System), potresti avere familiarità con le connessioni simultanee. QLDB non ha lo stesso concetto di connessione RDBMS tradizionale perché le transazioni vengono eseguite con messaggi di richiesta e risposta HTTP.

In QLDB, il concetto analogo è una sessione attiva. Una sessione è concettualmente simile all'accesso di un utente: gestisce le informazioni sulle richieste di transazioni di dati inviate a un registro. Una sessione attiva è una sessione che esegue attivamente una transazione. Può anche trattarsi di una sessione che ha recentemente concluso una transazione in cui il servizio prevede che avvierà immediatamente un'altra transazione. QLDB supporta una transazione in esecuzione attiva per sessione.

Il limite di sessioni attive simultanee per libro mastro è definito inQuote e limiti in Amazon QLDB. Una volta raggiunto questo limite, qualsiasi sessione che tenti di avviare una transazione genererà un errore (`LimitExceededException`).

Per le best practice per la configurazione di un pool di sessioni nell'applicazione utilizzando il driver QLDB, consulta iConfigurazione della QldbDriver oggetto consigli sui driver di Amazon QLDB.

Indice

Ciclo di vita di una sessione

La seguente sequenza di operazioni dell'API QLDB Session rappresenta il ciclo di vita tipico di una sessione QLDB:

1. `StartSession`
2. `StartTransaction`
3. `ExecuteStatement`
4. `CommitTransaction`
5. Ripeti i passaggi da 2 a 4 per avviare più transazioni (una transazione alla volta).
6. `EndSession`

Scadenza della sessione

QLDB scade e scarta una sessione dopo una durata totale di 13-17 minuti, indipendentemente dal fatto che stia eseguendo attivamente una transazione. Le sessioni possono essere perse o compromesse per diversi motivi, ad esempio guasti hardware, errori di rete o riavvii delle applicazioni. QLDB impone una durata massima delle sessioni per garantire che l'applicazione client sia resiliente agli errori della sessione.

Gestione delle sessioni nel driver QLDB

Sebbene sia possibile utilizzare unAWS SDK per interagire direttamente con l'API di sessione QLDB, ciò aggiunge complessità e richiede il calcolo di un commit digest. QLDB utilizza questo commit digest per garantire l'integrità delle transazioni. Invece di interagire direttamente con questa API, consigliamo di utilizzare il driver QLDB.

Il driver fornisce un livello di astrazione di alto livello sopra l'API dei dati transazionali. Semplifica il processo di esecuzione delle istruzioni PartiQL sui dati contabili gestendo le chiamate SendCommandAPI. Queste chiamate API richiedono diversi parametri che il driver gestisce per te, tra cui la gestione delle sessioni, delle transazioni e la politica di riprova in caso di errori.

Argomenti

- Panoramica del pool di sessioni
- Raggruppamento delle sessioni e logica delle transazioni
- Restituzione delle sessioni in piscina

Panoramica del pool di sessioni

Nelle versioni precedenti del driver QLDB (ad esempio, Java v1.1.0), abbiamo fornito due implementazioni dell'oggetto driver: una standard, non in pool`QldbDriver` e una`PooledQldbDriver`. Come suggerisce il nome,`PooledQldbDriver` mantiene un pool di sessioni che vengono riutilizzate tra le transazioni.

In base al feedback degli utenti, gli sviluppatori preferiscono utilizzare la funzionalità di pooling e i relativi vantaggi come impostazione predefinita, anziché utilizzare il driver standard. Quindi, abbiamo rimosso`PooledQldbDriver` e spostato la funzionalità di pooling delle sessioni su`QldbDriver`. Questa modifica è inclusa nell'ultima versione di ogni driver (ad esempio, Java v2.0.0).

Il driver offre tre livelli di astrazioni:

- Driver (implementazione del driver in pool): l'astrazione di primo livello. Il conducente gestisce e gestisce un pool di sessioni. Quando si chiede al conducente di eseguire una transazione, il conducente sceglie una sessione dal pool e la utilizza per eseguire la transazione. Se la transazione fallisce a causa di un errore di sessione (`InvalidSessionException`), il driver utilizza un'altra sessione per riprovare la transazione. In sostanza, il conducente offre un'esperienza di sessione completamente gestita.
- Sessione: un livello al di sotto dell'astrazione del driver. Una sessione è contenuta in un pool e il driver gestisce il ciclo di vita della sessione. Se una transazione fallisce, il driver la riprova fino a un numero specificato di tentativi utilizzando la stessa sessione. Ma se la sessione incontra un errore (`InvalidSessionException`), QLDB lo scarta internamente. Il conducente è quindi responsabile di ottenere un'altra sessione dal pool per riprovare la transazione.
- Transazione: il livello di astrazione più basso. Una transazione è contenuta in una sessione e la sessione gestisce il ciclo di vita della transazione. La sessione è responsabile del nuovo tentativo

della transazione in caso di errore. La sessione garantisce inoltre che non venga divulgata una transazione aperta che non è stata confermata o annullata.

Nella versione più recente di ogni driver, è possibile eseguire operazioni solo a livello di driver di astrazione. Non hai il controllo diretto sulle singole sessioni e transazioni (ovvero, non ci sono operazioni API per avviare manualmente una nuova sessione o transazione).

Raggruppamento delle sessioni e logica delle transazioni

L'ultima versione di ogni driver non fornisce più un'implementazione non condivisa dell'oggetto driver. Per impostazione predefinita, l'`QldbDriver`oggetto gestisce il pool di sessioni. Quando si effettua una chiamata per eseguire una transazione, il conducente effettua le seguenti operazioni:

1. Il conducente verifica se ha raggiunto il limite del pool di sessioni. In tal caso, il conducente genera immediatamente un'`NoSessionAvailable`eccezione. Altrimenti, passa alla fase successiva.
2. Il driver verifica se il pool dispone di una sessione disponibile.
   - Se nel pool è disponibile una sessione, il driver la utilizza per eseguire una transazione.
   - Se nel pool non è disponibile alcuna sessione, il driver crea una nuova sessione e la utilizza per eseguire una transazione.
3. Dopo aver ottenuto una sessione nel passaggio 2, il driver richiama l'`execute`operazione sull'istanza della sessione.
4. Durante il`execute` funzionamento della sessione, il conducente tenta di avviare una transazione chiamando`startTransaction`.
   - Se la sessione non è valida, la`startTransaction` chiamata fallisce e il conducente torna al passaggio 1.
   - Se la`startTransaction` chiamata ha esito positivo, il conducente passa alla fase successiva.
5. Il conducente esegue la tua espressione lambda. Questa espressione lambda può contenere una o più chiamate per eseguire istruzioni PartiQL. Dopo che l'espressione lambda termina l'esecuzione senza errori, il driver procede al salvataggio della transazione.
6. L'esecuzione della transazione può avere uno dei due risultati:
   - Il commit ha esito positivo e il driver restituisce il controllo al codice dell'applicazione.
   - Il commit fallisce a causa di un conflitto ottimistico per il controllo della concorrenza (OCC). In questo caso, il conducente riprova i passaggi da 4 a 6 utilizzando la stessa sessione.

Il numero massimo di tentativi di ripetizione è configurabile nel codice dell'applicazione. Il limite predefinito è4.

> **Note**
>
> `SeInvalidSessionException` viene generato un durante i passaggi da 4 a 6, il driver contrassegna la sessione come chiusa e torna al passaggio 1 per riprovare la transazione. Se viene generata un'altra eccezione durante i passaggi da 4 a 6, il conducente verifica se l'eccezione può essere ritentata. In tal caso, riprova la transazione fino al numero specificato di tentativi. In caso contrario, propaga (si riempie di bolle e genera) l'eccezione al codice dell'applicazione.

Restituzione delle sessioni in piscina

`SeInvalidSessionException` si verifica un errore in qualsiasi momento nel corso di una transazione attiva, il conducente non riporta la sessione al pool. QLDB scarta invece la sessione e il driver ottiene un'altra sessione dal pool. In tutti gli altri casi, il driver riporta la sessione nel pool.

Suggerimenti per i driver Amazon QLDB

Questa sezione descrive le best practice per la configurazione e l'utilizzo del driver Amazon QLDB per qualsiasi lingua supportata. Gli esempi di codice forniti sono specificamente per Java.

Questi consigli si applicano alla maggior parte dei casi d'uso tipici, ma la taglia unica non è adatta a tutti. Utilizza i seguenti consigli come preferisci per la tua applicazione.

Argomenti

Configurazione della QldbDriver oggetto

La`QldbDriver`object gestisce le connessioni al libro mastro mantenendo un pool disessioniche vengono riutilizzati tra le transazioni. UNsessionerappresenta una singola connessione con il libro contabile. QLDB supporta una transazione attiva per sessione.

Important

Per le versioni precedenti del driver, la funzionalità di pooling delle sessioni è ancora presente`PooledQldbDriver`l'oggetto anzicché`QldbDriver`. Se si utilizza una delle seguenti versioni, sostituire qualsiasi menzione di`QldbDriver`con`PooledQldbDriver`per il resto di questo argomento.

Driver	Versione
Java	`1.1.0`o precedenti
.NET	`0.1.0-beta`
Node.js	`1.0.0-rc.1` o precedenti
Python	`2.0.2`o precedenti

La`PooledQldbDriver`l'oggetto è obsoleto nella versione più recente dei driver. Ti consigliamo di eseguire l'aggiornamento alla versione più recente e convertire qualsiasi istanza di`PooledQldbDriver`a`QldbDriver`.

Configura QldbDriver come oggetto globale

Per ottimizzare l'uso di driver e sessioni, assicurarsi che nell'istanza dell'applicazione esista solo un'istanza globale del driver. Ad esempio, in Java, è possibile utilizzareiniezione dipendenzaFramework comePrimavera,Guida Google, oppurePugnale. Il codice di esempio seguente mostra come configurare`QldbDriver`come 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();
}
```

Configurare i tentativi di nuovo tentativo

Il conducente ritenta automaticamente le transazioni quando eccezioni transitorie comuni (come ad esempio`SocketTimeoutException`o`NoHttpResponseException`) si verificano. Per impostare il numero massimo di tentativi, è possibile utilizzare il`maxRetries`parametro del`transactionRetryPolicy`oggetto di configurazione durante la creazione di un'istanza di`QldbDriver`. (Per le versioni precedenti del driver elencate nella sezione precedente, utilizzare il`retryLimit`parametro di`PooledQldbDriver`.)

Il valore predefinito di `maxRetries` è 4.

Errori lato client come`InvalidParameterException`non può essere riprovato. Quando si verificano, la transazione viene interrotta, la sessione viene restituita al pool e l'eccezione viene lanciata al client del conducente.

Configurare il numero massimo di sessioni e transazioni simultanee

Il numero massimo di sessioni contabili utilizzate da un'istanza di`QldbDriver`per eseguire transazioni è definito dal suo`maxConcurrentTransactions`Parametro . (Per le versioni precedenti del driver elencate nella sezione precedente, questo è definito dal`poolLimit`parametro di`PooledQldbDriver`.)

Questo limite deve essere maggiore di zero e minore o uguale al numero massimo di connessioni HTTP aperte consentite dal client di sessione, come definito dallo specificoAWSSDK. Ad esempio, in Java, il numero massimo di connessioni è impostato nel[ClientConfiguration](#)oggetto.

Il valore predefinito di`maxConcurrentTransactions`è l'impostazione massima di connessione del tuoAWSSDK.

Quando si configura il`QldbDriver`nella tua applicazione, prendi le seguenti considerazioni di scalabilità:

- Il pool dovrebbe avere sempre almeno il numero di sessioni pari al numero di transazioni in esecuzione contemporanea che si prevede di avere.
- In un modello multi-thread in cui un thread di supervisore delega ai thread di lavoro, il driver dovrebbe avere almeno il numero di sessioni del numero di thread di lavoro. Altrimenti, al picco di carico, i thread saranno in linea per una sessione disponibile.
- Il limite di servizio delle sessioni attive simultanee per libro contabile è definito inQuote e limiti in Amazon QLDB. Assicurati di non configurare più di questo limite di sessioni simultanee da utilizzare per un singolo libro contabile in tutti i client.

Nuovi tentativi di eccezione

Quando si riprova le eccezioni che si verificano in QLDB, prendere in considerazione le seguenti raccomandazioni.

Nuovi tentativi su OCCConflictException

Controllo ottimistico della concorrenzaLe eccezioni di conflitto (OCC) si verificano quando i dati a cui si accede alla transazione sono cambiati dall'inizio della transazione. QLDB genera questa eccezione mentre si tenta di eseguire il commit della transazione. Il conducente ritira la transazione fino a quante volte`maxRetries`è configurato.

Per ulteriori informazioni sull'OCC e sulle best practice per l'utilizzo di indici per limitare i conflitti OCC, vedereModello di concorrenza Amazon QLDB.

Riprovare altre eccezioni al di fuori di QLDBDriver

Per riprovare una transazione al di fuori del driver quando personalizzate vengono generate eccezioni definite dall'applicazione durante il runtime, è necessario avvolgere la transazione. Ad esempio, in Java, il codice seguente mostra come utilizzare il codiceReslienza 4Jlibreria per riprovare una transazione in 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**
>
> Il ritentativo di una transazione al di fuori del driver QLDB ha un effetto moltiplicatore. Ad esempio, se`QldbDriver`è configurato per riprovare tre volte e anche la logica di riprova personalizzata torna tre volte, la stessa transazione può essere ripetuta fino a nove volte.

Rendere idempotenti le transazioni

Come best practice, rendi le tue transazioni di scrittura idempotenti per evitare effetti collaterali imprevisti in caso di tentativi. Una transazione è.idempotentese può essere eseguito più volte e produrre risultati identici ogni volta.

Per ulteriori informazioni, consulta Modello di concorrenza Amazon QLDB.

Ottimizzazione delle prestazioni

Per ottimizzare le prestazioni quando si eseguono transazioni utilizzando il driver, attenersi alle seguenti considerazioni:

- La`execute`l'operazione fa sempre un minimo di tre`SendCommand`Chiamate API a QLDB, inclusi i seguenti comandi:
  1. `StartTransaction`
  2. `ExecuteStatement`

     Questo comando viene richiamato per ogni istruzione PartiQL eseguita nella`execute`blocco.
  3. `CommitTransaction`

  Considera il numero totale di chiamate API effettuate quando si calcola il carico di lavoro complessivo dell'applicazione.
- In generale, si consiglia di iniziare con uno scrittore a thread singolo e di ottimizzare le transazioni mediante il batch di più istruzioni all'interno di una singola transazione. Massimizzare le quote relative alle dimensioni della transazione, alla dimensione del documento e al numero di documenti per transazione, come definito nellaQuote e limiti in Amazon QLDB.
- Se il batch non è sufficiente per carichi di transazioni di grandi dimensioni, puoi provare il multi-threading aggiungendo altri scrittori. Tuttavia, è necessario considerare attentamente i requisiti applicativi per il sequenziamento di documenti e transazioni e la complessità aggiuntiva che questo introduce.

Esecuzione di più rendiconti per

Come descritto nelsezione precedente, è possibile eseguire più rendiconti per transazione per ottimizzare le prestazioni dell'applicazione. Nell'esempio di codice seguente, si esegue una query su una tabella e quindi si aggiorna un documento in tale tabella all'interno di una transazione. Lo fai passando un'espressione lambda al`execute`operazione.

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))
```

Il conducente`execute`operazione avvia implicitamente una sessione e una transazione in quella sessione. Ogni istruzione eseguita nell'espressione lambda è racchiusa nella transazione. Dopo

aver eseguito tutte le istruzioni, il conducente commette automaticamente la transazione. Se una dichiarazione non riesce dopo che il limite di tentativi automatici è stato esaurito, la transazione viene interrotta.

Propagare le eccezioni in una transazione

Quando si eseguono più rendiconti per transazione, generalmente non consigliamo di catch e ingoiare eccezioni all'interno della transazione.

Ad esempio, in Java, il seguente programma cattura qualsiasi istanza di`RuntimeException`, registra l'errore e continua. Questo esempio di codice è considerato una cattiva pratica perché la transazione ha esito positivo anche quando`UPDATE`l'istruzione fallisce. Quindi, il client potrebbe presumere che l'aggiornamento abbia avuto esito positivo quando non è stato eseguito.

> **Warning**
>
> Non utilizzare questo esempio di codice. Viene fornito per mostrare un esempio anti-pattern considerato una cattiva pratica.

```
// 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.");
}
```

Propaga (ribalta) invece l'eccezione. Se una qualsiasi parte della transazione fallisce, lascia che`execute`operazione interrompe la transazione in modo che il cliente possa gestire l'eccezione di conseguenza.

Informazioni sulla politica di riprova con il driver in Amazon QLDB

Il driver Amazon QLDB utilizza una politica di riprova per gestire le eccezioni transitorie riprovando in modo trasparente una transazione fallita. Queste eccezioni, come`CapacityExceededException`e`RateExceededException`, di solito correggersi dopo un periodo di tempo. Se la transazione che ha avuto esito negativo con l'eccezione viene ritentata dopo un ritardo adeguato, è probabile che abbia esito positivo. Ciò aiuta a migliorare la stabilità dell'applicazione che utilizza QLDB.

Argomenti

- Tipi di errori riprovabili
- Policy di nuovi tentativi predefinita

Tipi di errori riprovabili

Il conducente ritorna automaticamente una transazione se e solo se si verifica una delle seguenti eccezioni durante un'operazione all'interno di tale transazione:

- Eccezione della capacità superata— Restituito quando la richiesta supera la capacità di elaborazione del libro contabile.
- Eccezione di sessione non valida— Restituito quando una sessione non è più valida o se la sessione non esiste.
- LimitExceededException— Restituito se viene superato un limite di risorse come il numero di sessioni attive.
- Eccezione del conflitto OCC— Restituito quando una transazione non può essere scritta sul giornale di registrazione a causa di un errore nella fase di verifica dicontrollo ottimistico della concorrenza(OCC).
- Eccezione della tariffa superata— Restituito quando la velocità delle richieste supera il throughput consentito.

Policy di nuovi tentativi predefinita

La politica di riprova consiste in una condizione di riprova e una strategia di backoff. La condizione di riprova definisce quando una transazione deve essere ripetuta, mentre la strategia di backoff definisce quanto tempo attendere prima di riprovare la transazione.

Quando si crea un'istanza del driver, il criterio di riprova predefinito specifica di riprovare fino a quattro volte e di utilizzare una strategia di backoff esponenziale. La strategia di backoff esponenziale utilizza un ritardo minimo di 10 millisecondi e un ritardo massimo di 5000 millisecondi, con uguale jitter. Se la transazione non può essere eseguita correttamente nel criterio di riprova, ti consigliamo di provare la transazione in un altro momento.

Il concetto di backoff esponenziale è di utilizzare tempi di attesa progressivamente più lunghi tra i tentativi per le risposte di errore consecutive. Per ulteriori informazioni, consulta la .AWSpost di blogBackoff esponenziale e jitter.

Errori comuni del driver Amazon QLDB

Questa sezione descrive gli errori di runtime che possono essere generati dal driver Amazon QLDB quando interagisce con l'API di sessione QLDB.

Di seguito è riportato un elenco di eccezioni comuni restituite dal driver. Ogni eccezione include il messaggio di errore specifico, seguito da una breve descrizione e suggerimenti per possibili soluzioni.

CapacityExceededException

> Messaggio: Capacità superata
>
> Amazon QLDB ha rifiutato la richiesta perché superava la capacità di elaborazione del libro mastro. QLDB applica un limite di scalabilità interno per registro per mantenere l'integrità e le prestazioni del servizio. Questo limite varia in base alla dimensione del carico di lavoro di ogni singola richiesta. Ad esempio, una richiesta può avere un carico di lavoro maggiore se esegue transazioni di dati inefficienti, ad esempio scansioni di tabelle risultanti da una query qualificata non indicizzata.
>
> Ti consigliamo di attendere prima di riprovare la richiesta. Se la tua applicazione riscontra costantemente questa eccezione, ottimizza i tuoi rendiconti e riduci la frequenza e il volume delle

richieste che invii al libro mastro. Esempi di ottimizzazione dei rendiconti includono l'esecuzione di un minor numero di rendiconti per transazione e l'ottimizzazione degli indici delle tabelle. Per informazioni su come ottimizzare le dichiarazioni ed evitare la scansione delle tabelle, consultaOttimizzazione delle prestazioni delle query.

Ti consigliamo anche di utilizzare sempre la versione più recente del driver QLDB. Il driver ha una politica di ripetizione predefinita che utilizza Exponential Backoff e Jitter per riprovare automaticamente in caso di eccezioni come questa. Il concetto di backoff esponenziale consiste nell'utilizzare tempi di attesa progressivamente più lunghi tra i tentativi per le risposte di errore consecutive.

InvalidSessionException

Messaggio: *`TransactionID della`* transazione è scaduto

Una transazione ha superato la sua durata massima. Una transazione può essere eseguita fino a 30 secondi prima di essere confermata. Dopo questo limite di timeout, qualsiasi lavoro svolto sulla transazione viene rifiutato e QLDB elimina la sessione. Questo limite protegge il cliente dalla perdita di sessioni avviando transazioni e non eseguendole o annullandole.

Se questa è un'eccezione comune nella tua applicazione, è probabile che le transazioni stiano semplicemente impiegando troppo tempo per essere eseguite. Se il tempo di esecuzione delle transazioni richiede più di 30 secondi, ottimizza i rendiconti per velocizzare le transazioni. Esempi di ottimizzazione dei rendiconti includono l'esecuzione di un minor numero di rendiconti per transazione e l'ottimizzazione degli indici delle tabelle. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.

InvalidSessionException

Messaggio: *`SessionID`* di sessione scaduto

QLDB ha eliminato la sessione perché ha superato la sua durata totale massima. QLDB elimina le sessioni dopo 13-17 minuti, indipendentemente da una transazione attiva. Le sessioni possono essere perse o compromesse per una serie di motivi, ad esempio guasti hardware, errori di rete o riavvii delle applicazioni. Pertanto, QLDB impone una durata massima delle sessioni per garantire che il software client sia resistente agli errori di sessione.

Se riscontri questa eccezione, ti consigliamo di acquisire una nuova sessione e riprovare a eseguire la transazione. Si consiglia inoltre di utilizzare la versione più recente del driver QLDB, che gestisce il pool di sessioni e il suo stato per conto dell'applicazione.

InvalidSessionException

Messaggio: sessione non disponibile

Il client ha provato a effettuare transazioni con QLDB utilizzando una sessione che non esiste. Supponendo che il client stia utilizzando una sessione che esisteva in precedenza, la sessione potrebbe non esistere più a causa di uno dei seguenti motivi:

- Se una sessione è coinvolta in un errore interno del server (ovvero un errore con il codice di risposta HTTP 500), QLDB potrebbe scegliere di eliminare completamente la sessione, anziché consentire al cliente di effettuare transazioni con una sessione in stato incerto. Quindi, qualsiasi tentativo di nuovo tentativo in quella sessione fallisce e questo errore.
- Le sessioni scadute vengono infine dimenticate da QLDB. Quindi, qualsiasi tentativo di continuare a utilizzare la sessione genererà questo errore, anziché quello iniziale`InvalidSessionException`.

Se riscontri questa eccezione, ti consigliamo di acquisire una nuova sessione e riprovare a eseguire la transazione. Si consiglia inoltre di utilizzare la versione più recente del driver QLDB, che gestisce il pool di sessioni e il suo stato per conto dell'applicazione.

RateExceededException

Messaggio: La tariffa è stata superata

QLDB ha bloccato un client in base all'identità del chiamante. QLDB impone la limitazione per regione e per account utilizzando un algoritmo di limitazione dei bucket di token. QLDB fa questo per aiutare le prestazioni del servizio e garantire un utilizzo equo a tutti i clienti QLDB. Ad esempio, il tentativo di acquisire un numero elevato di sessioni simultanee utilizzando l'`StartSessionRequest`operazione potrebbe comportare una limitazione.

Per mantenere l'integrità dell'applicazione e mitigare l'ulteriore limitazione, puoi riprovare a utilizzare questa eccezione utilizzando Exponential Backoff e Jitter. Il concetto di backoff esponenziale consiste nell'utilizzare tempi di attesa progressivamente più lunghi tra i tentativi per le risposte di errore consecutive. Ti consigliamo di utilizzare sempre la versione più recente del driver QLDB. Il driver ha una politica di ripetizione predefinita che utilizza backoff e jitter esponenziali per riprovare automaticamente in caso di eccezioni come questa.

L'ultima versione del driver QLDB può essere utile anche se l'applicazione viene costantemente limitata da QLDB per`StartSessionRequest` le chiamate. Il driver mantiene un pool di sessioni che vengono riutilizzate tra le transazioni, il che può aiutare a ridurre il numero

di`StartSessionRequest` chiamate effettuate dall'applicazione. Per richiedere un aumento dei limiti di throttling delle API, contatta il AWS SupportCentro.

LimitExceededException

Messaggio: superato il limite di sessione

Un libro mastro ha superato la quota (nota anche come limite) relativa al numero di sessioni attive. Questa quota è definita inQuote e limiti in Amazon QLDB. Il conteggio delle sessioni attive di un libro contabile è alla fine coerente e i libri contabili che si avvicinano costantemente alla quota potrebbero vedere periodicamente questa eccezione.

Per mantenere l'integrità dell'applicazione, ti consigliamo di riprovare con questa eccezione. Per evitare questa eccezione, assicuratevi di non aver configurato più di 1.500 sessioni simultanee da utilizzare per un singolo libro mastro su tutti i client. Ad esempio, puoi utilizzare il maxConcurrentTransactionsmetodo del driver Amazon QLDB per Java per configurare il numero massimo di sessioni disponibili in un'istanza di driver.

QldbClientException

Messaggio: un risultato in streaming è valido solo quando la transazione principale è aperta

La transazione è chiusa e non può essere utilizzata per recuperare i risultati da QLDB. Una transazione si chiude quando viene confermata o annullata.

Questa eccezione si verifica quando il client lavora direttamente con l'`Transaction`oggetto e sta cercando di recuperare i risultati da QLDB dopo aver eseguito o annullato una transazione. Per ovviare a questo problema, il cliente deve leggere i dati prima di chiudere la transazione.

Guida introduttiva ad Amazon QLDB utilizzando un esempio di tutorial applicativo

In questo tutorial, utilizzi il driver Amazon QLDB con unAWS SDK per creare un registro QLDB e popolarlo con dati di esempio. Il driver consente all'applicazione di interagire con QLDB utilizzando l'API dei dati transazionali. L'AWSSDK supporta l'interazione con l'API di gestione delle risorse QLDB.

Il registro di esempio creato in questo scenario è un database del dipartimento dei veicoli a motore (DMV) che tiene traccia delle informazioni storiche complete sulle immatricolazioni dei veicoli.

I seguenti argomenti spiegano come aggiungere le immatricolazioni dei veicoli, modificarle e visualizzare la cronologia delle modifiche a tali immatricolazioni. Questa guida mostra anche come verificare crittograficamente un documento di registrazione e si conclude ripulendo le risorse ed eliminando il registro di esempio.

Questo tutorial di un'applicazione di esempio è disponibile per i seguenti linguaggi di programmazione.

Argomenti

- Guida su Java di Amazon QLDB
- Tutorial su Amazon QLDB Node.js
- Tutorial di Amazon QLDB di Python

Guida su Java di Amazon QLDB

In questa implementazione dell'applicazione di esempio del tutorial, si utilizza il driver Amazon QLDB con ilAWS SDK for Java per creare un registro QLDB e compilarlo con dati di esempio.

Man mano che si procede con questo tutorial, è possibile fare riferimento alla AWS SDK for JavaDocumentazione di riferimento delle API di gestione. Per le operazioni sui dati transazionali, è possibile fare riferimento al QLDB Driver for Java API Reference.

> **Note**
>
> Ove applicabile, alcuni passaggi del tutorial hanno comandi o esempi di codice diversi per ciascuna versione principale supportata del driver QLDB per Java.

Argomenti

- Installazione dell'applicazione di esempio Java Amazon QLDB
- Fase 1: creazione di un nuovo libro mastro
- Fase 2. Test della connettività al libro mastro
- Fase 3: creare tabelle, indici e dati di esempio
- Fase 4: Interrogare le tabelle in un libro mastro
- Fase 5: Modificare i documenti in un libro mastro

- Passaggio 6: visualizzare la cronologia delle revisioni di un documento
- Fase 7: Verificare un documento in un libro mastro
- Fase 8: Esportazione e convalida dei dati del giornale contabile in un libro mastro
- Fase 9 (opzionale): eliminazione delle risorse

Installazione dell'applicazione di esempio Java Amazon QLDB

Questa sezione descrive come installare ed eseguire l'applicazione di esempio Amazon QLDB fornita per il tutorial step-by-step Java. Il caso d'uso di questa applicazione di esempio è un database del dipartimento dei veicoli a motore (DMV) che tiene traccia delle informazioni storiche complete sulle immatricolazioni dei veicoli.

L'applicazione di esempio DMV per Java è open source nel GitHub repository aws-samples/amazon-qldb-dmv-sample -java.

Prerequisiti

Prima di iniziare, assicurarti di aver completato il driver QLDB per JavaPrerequisiti. Questo include gli output seguenti:

1. Registrazione aAWS.
2. Crea un utente con le autorizzazioni QLDB appropriate. Per completare tutti i passaggi di questo tutorial, è necessario l'accesso amministrativo completo alla risorsa contabile tramite l'API QLDB.
3. Se utilizzi un IDE diverso daAWS Cloud9, installa Java e concedi l'accesso programmatico per lo sviluppo.

Installazione

I passaggi seguenti descrivono come scaricare e configurare l'applicazione di esempio con un ambiente di sviluppo locale. In alternativa, è possibile automatizzare la configurazione dell'applicazione di esempio utilizzandolaAWS Cloud9 come IDE e unAWS CloudFormation modello per fornire le risorse di sviluppo.

Ambiente di sviluppo locale

Queste istruzioni descrivono come scaricare e installare l'applicazione di esempio Java QLDB utilizzando le proprie risorse e l'ambiente di sviluppo.

Per scaricare ed esegui l'applicazione di esempio di

1. Immettere il seguente seguente seguente seguente per clonare l'applicazione di esempio 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
   ```

   Questo pacchetto include la configurazione Gradle e il codice completo diTutorial su Java.

2. Carica ed esegui l'applicazione fornita.

   - Se stai usando Eclipse:

     a. Avvia Eclipse e nel menu Eclipse scegli File, Importa e quindi Progetto Gradle esistente.

     b. Nella directory principale del progetto, sfoglia e seleziona la directory dell'applicazione che contiene il`build.gradle` file. Quindi, scegli Fine per utilizzare le impostazioni Gradle predefinite per l'importazione.

     c. Puoi provare a eseguire il`ListLedgers` programma come esempio. Facendo clic con il pulsante destro del mouse, aprire il menu contestuale per il`ListLedgers.java` file e selezionare Esecuzione come applicazione Java.

   - Se usi IntelliJ:

     a. Avvia IntelliJ e, nel menu IntelliJ, scegli File, quindi Apri.

     b. Nella directory principale del progetto, sfoglia e seleziona la directory dell'applicazione che contiene il`build.gradle` file. Quindi, scegli OK. Mantieni le impostazioni predefinite e scegli nuovamente OK.

     c. Puoi provare a eseguire il`ListLedgers` programma come esempio. Apri il menu contestuale (fai clic con il pulsante destro del mouse) per il`ListLedgers.java` file e scegli Esecuzione 'ListLedgers'.

3. ProcediFase 1: creazione di un nuovo libro mastro con l'avvio del tutorial e crea un libro mastro.

AWS Cloud9

Queste istruzioni descrivono come automatizzare la configurazione dell'applicazione di esempio di registrazione dei veicoli Amazon QLDB per Java, utilizzando AWS Cloud9come IDE. In questa guida, utilizzi un AWS CloudFormationmodello per fornire le tue risorse di sviluppo.

Per ulteriori informazioni su AWS Cloud9, consulta la Guida per l'utente di AWS Cloud9. Per ulteriori informazioni su AWS CloudFormation, consulta la Guida per l'utente di AWS CloudFormation.

Argomenti

- Fase 1: Metti a disposizione le risorse a tua disposizione
- Fase 2: configurare l'IDE
- Parte 3: Esegui l'applicazione di esempio QLDB DMV

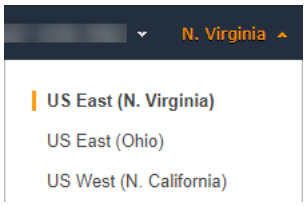
Fase 1: Metti a disposizione le risorse a tua disposizione

In questa prima fase, deviAWS CloudFormation fornire le risorse necessarie per configurare il tuo ambiente di sviluppo con l'applicazione di esempio Amazon QLDB.

Per aprire laAWS CloudFormation console e caricare il modello di applicazione di esempio QLDB

1. Accedere alla AWS Management Console e aprire la console di AWS CloudFormation all'indirizzo https://console.aws.amazon.com/cloudformation.

   Passa a una regione che supporti QLDB. Per un elenco completo, consulta gli endpoint e le quote di Amazon QLDB nel Riferimenti generali di AWS. La seguente schermataAWS Management Console mostra US East (N. Virginia) come selezionatiRegione AWS.

2. SullaAWS CloudFormation console, scegli Crea stack, quindi scegli Con nuove risorse (standard).
3. Nella pagina Crea stack, sotto Specifica modello, scegli l'URL di Amazon S3.
4. Inserisci il seguente URL e scegli Avanti.

   ```
   https://amazon-qldb-assets.s3.amazonaws.com/templates/QLDB-DMV-SampleApp.yml
   ```

5. Inserisci un nome per lo stack (ad esempio**`qldb-sample-app`**) e scegli Avanti.
6. Puoi aggiungere qualsiasi tag in base alle tue esigenze e mantenere le opzioni predefinite. Quindi scegli Next (Successivo).
7. Controlla le impostazioni dello stack e scegli Crea stack. Il completamento dellaAWS CloudFormation sceneggiatura potrebbe richiedere alcuni minuti.

   Questo script fornisce all'AWS Cloud9ambiente un'istanza Amazon Elastic Compute Cloud (Amazon EC2) associata che usi per eseguire l'applicazione di esempio QLDB in questo tutorial. Inoltre clona il repository aws-samples/amazon-qldb-dmv-sample -java dall' GitHub ambiente diAWS Cloud9 sviluppo dell'utente.

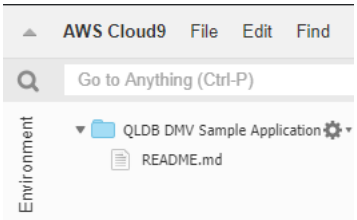
Fase 2: configurare l'IDE

In questa fase si completa la configurazione dell'ambiente di sviluppo cloud. Scaricate ed eseguite uno script di shell fornito per configurare il vostroAWS Cloud9 IDE con le dipendenze dell'applicazione di esempio.

Per configurare l'AWS Cloud9ambiente

1. Apri laAWS Cloud9 console all'indirizzo https://console.aws.amazon.com/cloud9/.
2. In I tuoi ambienti, individua la scheda per l'ambiente denominata QLDB DMV Sample Application e scegli Open IDE. Il caricamento dell'ambiente potrebbe richiedere un minuto all'avvio dell'istanza EC2 sottostante.

   L'AWS Cloud9ambiente è preconfigurato con le dipendenze di sistema necessarie per eseguire il tutorial. Nel riquadro di navigazione Ambiente della console, conferma di visualizzare una cartella denominata`QLDB DMV Sample Application`. La seguente schermata dellaAWS Cloud9 console mostra il riquadro della cartella dell'ambiente QLDB DMV Sample Application.



   Se non vedi un pannello di navigazione, attiva la scheda Ambiente sul lato sinistro della console. Se non vedi alcuna cartella nel riquadro,

abilita Show Environment Root utilizzando l'icona delle impostazioni
().

3. Nel riquadro inferiore della console, dovresti vedere una finestra dibash terminale aperta. Se non lo vedi, scegli Nuovo terminale dal menu Finestra nella parte superiore della console.
4. Quindi, scarica ed esegui uno script di installazione per installare OpenJDK 8 e, se applicabile, controlla il ramo appropriato dal repository Git. NelAWS Cloud9 terminale creato nel passaggio precedente, esegui i seguenti due comandi in ordine:

   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
   ```

   Al termine, dovrebbe essere visualizzato il seguente messaggio stampato nel terminale:

   ```
   ** DMV Sample App setup completed , enjoy!! **
   ```

5. Prenditi un momento per sfogliare il codice dell'applicazione di esempioAWS Cloud9, in particolare nel seguente percorso di directory:`src/main/java/software/amazon/qldb/tutorial`.

Parte 3: Esegui l'applicazione di esempio QLDB DMV

In questo passaggio, imparerai come eseguire le attività dell'applicazione di esempio Amazon QLDB DMV utilizzandoAWS Cloud9. Per eseguire il codice di esempio, torna al tuoAWS Cloud9 terminale o crea una nuova finestra di terminale come hai fatto nella Parte 2: Configura il tuo IDE.

Per eseguire un'applicazione di esempio

1. Esecuzione del seguente seguente seguente seguente seguente seguente nel terminale per passare alla directory principale del progetto:

   ```
   cd ~/environment/amazon-qldb-dmv-sample-java
   ```

   Assicurati di eseguire gli esempi nel seguente percorso di directory.

   ```
   /home/ec2-user/environment/amazon-qldb-dmv-sample-java/
   ```

2. Il comando seguente mostra la sintassi Gradle per eseguire ogni attività.

   ```
   ./gradlew run -Dtutorial=Task
   ```

   Ad esempio, esegui il seguente seguente seguente seguente per elencare tutti i libri mastri nella Regione correnteAccount AWS e nella tua.

   ```
   ./gradlew run -Dtutorial=ListLedgers
   ```

3. ProcediFase 1: creazione di un nuovo libro mastro con l'avvio del tutorial e crea un libro mastro.
4. (Facoltativo) Dopo aver completato il tutorial, ripulire leAWS CloudFormation risorse se non sono più necessarie.

   a. Apri laAWS CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation ed elimina lo stack creato nella Parte 1: Provisions Your Resources.

   b. Elimina anche loAWS Cloud9 stack che ilAWS CloudFormation modello ha creato per te.

Fase 1: creazione di un nuovo libro mastro

In questo passaggio, crei un nuovo registro Amazon QLDB denominato`vehicle-registration`.

Per creare un nuovo libro mastro

1. Esaminate il seguente file (`Constants.java`), che contiene valori costanti utilizzati da tutti gli altri programmi di questo tutorial.

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**
>
> Per il pacchetto Amazon Ion, devi usare il namespace`com.amazon.ion` della tua applicazione. AWS SDK for JavaDipende da un altro pacchetto Ion nel namespace`software.amazon.ion`, ma questo è un pacchetto legacy che non è compatibile con il driver 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**
>
> Questa`Constants` classe include un'istanza della`IonValueMapper` classe Jackson open source. Puoi utilizzare questo mappatore per elaborare i dati di Amazon Ion durante le transazioni di lettura e scrittura.

Il`CreateLedger.java` file dipende anche dal seguente programma (`DescribeLedger.java`), che descrive lo stato corrente del libro contabile.

```
/*
 * 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. Compila ed esegui il`CreateLedger.java` programma per creare un registro denominato`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**
>
> - Nella`createLedger` chiamata, è necessario specificare un nome contabile e una modalità di autorizzazione. Ti consigliamo di utilizzare la modalità di`STANDARD` autorizzazione per incrementare la sicurezza dei dati nel libro mastro.
> - Quando si crea un libro mastro, la protezione dall'eliminazione è abilitata per impostazione predefinita. Questa è una funzionalità di QLDB che impedisce a un utente qualsiasi di eliminare i libri mastro. È possibile disabilitare la protezione dall'eliminazione durante la creazione del libro contabile utilizzando l'API QLDB o ilAWS Command Line Interface (AWS CLI).
> - Facoltativamente, è anche possibile specificare i tag da allegare al libro mastro.

Per verificare la tua connessione al nuovo registro, procedi aFase 2. Test della connettività al libro mastro.

Fase 2. Test della connettività al libro mastro

In questo passaggio, verifichi di poterti connettere al`vehicle-registration` libro mastro in Amazon QLDB utilizzando l'endpoint dell'API dei dati transazionali.

Per testare la connettività al libro mastro

1. Esamina il seguente programma (`ConnectToLedger.java`), che crea una connessione di sessione dati al`vehicle-registration` libro mastro.

   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**
>
> - Per eseguire operazioni sui dati sul libro mastro, è necessario creare un'istanza della`QldbDriver` classe per connettersi a un libro mastro specifico. Si tratta di un oggetto`AmazonQLDB` client diverso da quello utilizzato nel passaggio precedente per creare il registro. Il client precedente viene utilizzato solo per eseguire le operazioni dell'API di gestione elencate in[Documentazione di riferimento dell'API Amazon QLDB](#).
> - Innanzitutto, crea un`QldbDriver` oggetto. È necessario specificare un nome contabile quando si crea questo driver.
>
>   Quindi, è possibile utilizzare il`execute` metodo di questo driver per eseguire istruzioni PartiQL.
> - Facoltativamente, puoi specificare un numero massimo di tentativi per le eccezioni delle transazioni. Il`execute` metodo riprova automaticamente i conflitti OCC (Optimistic Concurrency Control) e altre eccezioni transitorie comuni fino a questo limite configurabile. Il valore di default è 4.
>
>   Se la transazione continua a fallire dopo il raggiungimento del limite, il driver genera l'eccezione. Per ulteriori informazioni, consulta [Informazioni sulla politica di riprova con il driver in 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**
>
> - Per eseguire operazioni sui dati sul libro mastro, è necessario creare un'istanza della`QldbDriver` classe`PooledQldbDriver` o da connettere a un libro mastro specifico. Si tratta di un oggetto`AmazonQLDB` client diverso da quello utilizzato nel passaggio precedente per creare il registro. Il client precedente viene utilizzato solo per eseguire le operazioni dell'API di gestione elencate in[Documentazione di riferimento dell'API Amazon QLDB](#).
>
> Si consiglia di utilizzare`PooledQldbDriver` a meno che non sia necessario implementare un pool di sessioni personalizzato con`QldbDriver`. La dimensione predefinita del pool`PooledQldbDriver` è il [numero massimo di connessioni HTTP aperte](#) consentite dal client di sessione.
> - Innanzitutto, crea un`PooledQldbDriver` oggetto. È necessario specificare un nome contabile quando si crea questo driver.
>
> Quindi, è possibile utilizzare il`execute` metodo di questo driver per eseguire istruzioni PartiQL. In alternativa, è possibile creare manualmente una sessione da questo oggetto driver in pool e utilizzare il`execute` metodo della sessione. Una sessione rappresenta una singola connessione con il libro mastro.
> - Facoltativamente, puoi specificare un numero massimo di tentativi per le eccezioni delle transazioni. Il`execute` metodo riprova automaticamente i conflitti OCC (Optimistic Concurrency Control) e altre eccezioni transitorie comuni fino a questo limite configurabile. Il valore di default è 4.

Se la transazione continua a fallire dopo il raggiungimento del limite, il driver genera l'eccezione. Per ulteriori informazioni, consulta Informazioni sulla politica di riprova con il driver in Amazon QLDB.

2. Compila ed esegui il`ConnectToLedger.java` programma per testare la connettività della sessione dati al`vehicle-registration` registro.

Sovrascrivere ilRegione AWS

L'applicazione di esempio si connette a QLDB come impostazione predefinitaRegione AWS, che è possibile impostare come descritto nel passaggio dei prerequisitiImpostazione delleAWS credenziali e della regione predefinite. È inoltre possibile modificare la regione modificando le proprietà del generatore del client di sessione QLDB.

2.x

Il seguente esempio di codice crea un'istanza di un nuovo`QldbSessionClientBuilder` oggetto.

```
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);
```

È possibile utilizzare il`region` metodo per eseguire il codice in QLDB in ogni regione in cui sia disponibile. Per un elenco completo, consulta gli endpoint e le quote di Amazon QLDB nel Riferimenti generali di AWS.

1.x

Il seguente esempio di codice crea un'istanza di un nuovo`AmazonQLDBSessionClientBuilder` oggetto.

```
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);
```

È possibile utilizzare il`withRegion` metodo per eseguire il codice in QLDB in ogni regione in cui sia disponibile. Per un elenco completo, consulta gli endpoint e le quote di Amazon QLDB nel Riferimenti generali di AWS.

Per creare tabelle nel`vehicle-registration` registro, procedere aFase 3: creare tabelle, indici e dati di esempio.

Fase 3: creare tabelle, indici e dati di esempio

Quando il tuo registro Amazon QLDB è attivo e accetta connessioni, puoi iniziare a creare tabelle con i dati sui veicoli, i loro proprietari e le loro informazioni di registrazione. Dopo aver creato le tabelle e gli indici, puoi caricarli con i dati.

In questa fase, verranno create quattro tabelle nel`vehicle-registration` libro mastro:

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

È inoltre possibile creare i seguenti indici.

Nome tabella	Campo
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Quando si inseriscono dati di esempio, si inseriscono innanzitutto i documenti nella`Person` tabella. Quindi, si utilizza il sistema assegnato`id` da ciascun`Person` documento per compilare i campi corrispondenti nei`DriversLicense` documenti`VehicleRegistration` e nei documenti appropriati.

> **Tip**
>
> Come procedura consigliata, utilizzare`id` come chiave esterna assegnata dal sistema di un documento. Sebbene sia possibile definire campi destinati a essere identificatori univoci (ad esempio, il VIN di un veicolo), il vero identificatore univoco di un documento è il suo`id`. Questo campo è incluso nei metadati del documento, che è possibile interrogare nella vista commit (la vista definita dal sistema di una tabella).
> Per ulteriori informazioni sulle viste in QLDB, consultareConcetti principali. Per ulteriori informazioni sui metadati, consultaInterrogazione dei metadati dei documenti.

Per impostare i dati di esempio

1. Esamina i seguenti`.java` file. Queste classi di modelli rappresentano i documenti archiviati nelle`vehicle-registration` tabelle. Sono serializzabili da e verso il formato Amazon Ion.

   > **Note**
   >
   > Documentazione Amazon QLDBsono archiviati in formato Ion, che è un superset di JSON. Quindi, puoi usare la libreria Jackson FasterXML per modellare i dati in 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. Esaminate il seguente file (`SampleData.java`), che rappresenta i dati di esempio che inserite nelle`vehicle-registration` tabelle.

   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**
>
> Per il pacchetto Amazon Ion, devi usare il namespace`com.amazon.ion` della tua applicazione. AWS SDK for JavaDipende da un altro pacchetto Ion nel namespace`software.amazon.ion`, ma questo è un pacchetto legacy che non è compatibile con il driver 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**
>
> - Questa classe utilizza le librerie Ion per fornire metodi di supporto che convertono i dati da e verso il formato Ion.
> - Il`getDocumentId` metodo esegue una query su una tabella con il prefisso`_ql_committed_`. Si tratta di un prefisso riservato che indica che si desidera interrogare la visualizzazione confermata di una tabella. In questa visualizzazione, i dati sono annidati nel`data` campo e i metadati sono annidati nel`metadata` campo.

3. Compila ed esegui il seguente programma (`CreateTable.java`) per creare le tabelle menzionate in precedenza.

   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**
>
> Questo programma dimostra come passare un`TransactionExecutor` lambda a l`execute` metodo. In questo esempio, si eseguono più istruzioni`CREATE TABLE` PartiQL in una singola transazione utilizzando un'espressione lambda.
> Il`execute` metodo avvia implicitamente una transazione, esegue tutte le istruzioni nella lambda e quindi esegue automaticamente il commit della transazione.

4. Compilate ed eseguite il seguente programma (`CreateIndex.java`) per creare indici sulle tabelle, come descritto in precedenza.

   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. Compila ed esegui il seguente programma (`InsertDocument.java`) per inserire i dati di esempio nelle tabelle.

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**
>
> - Questo programma dimostra come richiamare il`execute` metodo con valori parametrizzati. È possibile passare parametri di dati di tipo`IonValue` in aggiunta all'istruzione PartiQL che si desidera eseguire. Usa un punto interrogativo (?) come segnaposto variabile nella stringa del rendiconto.
> - Se un'`INSERT`istruzione ha successo, restituisce il valore`id` di ogni documento inserito.

Successivamente, è possibile utilizzare`SELECT` le dichiarazioni per leggere i dati dalle tabelle del`vehicle-registration` libro mastro. Continua con la Fase 4: Interrogare le tabelle in un libro mastro.

Fase 4: Interrogare le tabelle in un libro mastro

Dopo aver creato le tabelle in un registro Amazon QLDB e averle caricate con i dati, puoi eseguire interrogazioni per esaminare i dati di immatricolazione del veicolo che hai appena inserito. QLDB utilizza PartiQL come linguaggio di interrogazione e Amazon Ion come modello di dati orientato ai documenti.

PartiQL è un linguaggio di interrogazione open source e compatibile con SQL che è stato esteso per funzionare con Ion. Con PartiQL, puoi inserire, interrogare e gestire i tuoi dati con operatori SQL familiari. Amazon Ion è un superset di JSON. Ion è un formato di dati open source basato su documenti che offre la flessibilità di archiviare ed elaborare dati strutturati, semistrutturati e annidati.

In questo passaggio, si utilizzano iSELECT rendiconti per leggere i dati dalle tabelle delvehicle-registration libro mastro.

> **Warning**
>
> Quando si esegue una query in QLDB senza una ricerca indicizzata, viene richiamata una scansione completa della tabella. PartiQL supporta tali interrogazioni perché è compatibile con SQL. Tuttavia, non eseguite scansioni di tabelle per casi d'uso di produzione in QLDB. Le scansioni delle tabelle possono causare problemi di prestazioni su tabelle di grandi dimensioni, inclusi conflitti di concorrenza e timeout delle transazioni.
> Per evitare la scansione delle tabelle, è necessario eseguire istruzioni con una clausolaWHERE predicativa utilizzando un operatore di uguaglianza su un campo indicizzato o un ID di documento; ad esempio,`WHERE indexedField = 123` o`WHERE indexedField IN (456, 789)`. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.

Per interrogare le tabelle

- Compila ed esegui il seguente programma (`FindVehicles.java`) per interrogare tutti i veicoli registrati da una persona nel tuo registro.

  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**
>
> Innanzitutto, questo programma interroga la`Person` tabella del documento per`GovId` `LEWISR261LL` ottenere il relativo campo di`id` metadati.
> Quindi, utilizza questo documento`id` come chiave esterna per interrogare la`VehicleRegistration` tabella`PrimaryOwner.PersonId`. Inoltre si`VehicleRegistration` unisce al`Vehicle` tavolo sul`VIN` campo.

Per informazioni sulla modifica dei documenti nelle tabelle del`vehicle-registration` libro mastro, vedereFase 5: Modificare i documenti in un libro mastro.

Fase 5: Modificare i documenti in un libro mastro

Ora che hai dati su cui lavorare, puoi iniziare a modificare i documenti nel`vehicle-registration` registro in Amazon QLDB. In questa fase, i seguenti esempi di codice dimostrano come eseguire istruzioni DML (Data Manipulation) Queste dichiarazioni aggiornano il proprietario principale di un veicolo e aggiungono un proprietario secondario a un altro veicolo.

Per modificare i documenti

1. Compila ed esegui il seguente programma (`TransferVehicleOwnership.java`) per aggiornare il proprietario principale del veicolo con il`1N4AL11D75C109151` VIN nel tuo registro.

   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. Compila ed esegui il seguente programma (`AddSecondaryOwner.java`) per aggiungere un proprietario secondario al veicolo con il`KM8SRDHF6EU074761` VIN nel registro.

   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.");
    }
}
```

Per esaminare queste modifiche nel`vehicle-registration` registro, vedere[Passaggio 6: visualizzare la cronologia delle revisioni di un documento](#).

Passaggio 6: visualizzare la cronologia delle revisioni di un documento

Dopo aver modificato i dati di registrazione di un veicolo nel passaggio precedente, è possibile interrogare la cronologia di tutti i proprietari registrati e qualsiasi altro campo aggiornato. In questo passaggio, è necessario interrogare la cronologia delle revisioni di un documento nella`VehicleRegistration` tabella del`vehicle-registration` libro contabile.

Per visualizzare la cronologia delle revisioni

1. Esamina il seguente programma (`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**
>
> - È possibile visualizzare la cronologia delle revisioni di un documento interrogando la sintassi integrataFunzione di cronologia nella seguente sintassi.

```
SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
[ WHERE h.metadata.id = 'id' ]
```

- L'ora di inizio e l'ora di fine sono entrambe opzionali. Sono valori letterali di Amazon Ion che possono essere indicati con delle barrette inverse (`` `...` ``). Per ulteriori informazioni, consulta Interrogazione di Ion con PartiQL in Amazon QLDB.
- Come procedura consigliata, qualifica un'interrogazione cronologica con un intervallo di date (ora di inizio e ora di fine) e un ID del documento (`metadata.id`). QLDB elabora le`SELECT` interrogazioni nelle transazioni, che sono soggette a un limite di timeout delle transazioni.

  La cronologia QLDB è indicizzata in base all'ID del documento e al momento non è possibile creare indici di cronologia aggiuntivi. Le interrogazioni cronologiche che includono un'ora di inizio e un'ora di fine ottengono il vantaggio della qualificazione dell'intervallo di date.

2. Compila ed esegui il`QueryHistory.java` programma per interrogare la cronologia delle revisioni del`VehicleRegistration` documento con `VIN1N4AL11D75C109151`.

Per verificare crittograficamente una revisione del documento nel`vehicle-registration` libro mastro, procedere aFase 7: Verificare un documento in un libro mastro.

Fase 7: Verificare un documento in un libro mastro

Con Amazon QLDB, puoi verificare in modo efficiente l'integrità di un documento nel giornale contabile utilizzando l'hashing crittografico con SHA-256. Per ulteriori informazioni su come funzionano la verifica e l'hashing crittografico in QLDB, consultaVerifica dei dati in Amazon QLDB.

In questo passaggio, si verifica una revisione del documento nella`VehicleRegistration` tabella del`vehicle-registration` libro contabile. Innanzitutto, richiedi un riassunto, che viene restituito come file di output e funge da firma dell'intera cronologia delle modifiche del tuo libro contabile. Quindi, richiedi una prova della revisione relativa a quel riassunto. Utilizzando questa prova, l'integrità della revisione viene verificata se tutti i controlli di convalida vengono superati.

Per verificare la revisione di un documento

1. Esamina i seguenti`.java` file, che rappresentano gli oggetti QLDB necessari per la verifica e le classi di utilità con metodi di supporto per i valori Ion e stringa.

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. Utilizza due `.java` file (`GetDigest.javaeGetRevision.java`) per eseguire i seguenti passaggi:

   - Richiedi un nuovo riassunto dal `vehicle-registration` registro.
   - Richiedi una prova per ogni revisione di un documento dalla `VehicleRegistration` tabella.
   - Verifica le revisioni utilizzando il riassunto e la prova restituiti ricalcolando il riassunto.

   Il `GetDigest.java` programma contiene il seguente codice.

```
/*
 * 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**
>
> Utilizza `getDigest` questo metodo per richiedere un riassunto che copra la parte corrente del diario nel tuo libro contabile. Il suggerimento del diario si riferisce all'ultimo blocco salvato al momento in cui QLDB riceve la tua richiesta.

Il `GetRevision.java` programma contiene il seguente codice.

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**
>
> Dopo che il`getRevision` metodo ha restituito una prova per la revisione del documento specificata, questo programma utilizza un'API lato client per verificare tale revisione. Per una panoramica dell'algoritmo utilizzato da questa API, consultaUtilizzo di una bozza per ricalcolare il riassunto.

3. Compila ed esegui il`GetRevision.java` programma per verificare crittograficamente il`VehicleRegistration` documento con VIN`1N4AL11D75C109151`.

Per esportare e convalidare i dati del giornale`vehicle-registration` contabile nel libro mastro, procedere aFase 8: Esportazione e convalida dei dati del giornale contabile in un libro mastro.

Fase 8: Esportazione e convalida dei dati del giornale contabile in un libro mastro

In Amazon QLDB, puoi accedere al contenuto del giornale nel tuo registro per vari scopi, come la conservazione dei dati, l'analisi e il controllo. Per ulteriori informazioni, consulta Esportazione dei dati del diario da Amazon QLDB.

In questo passaggio, si esportano i blocchi di scritture`vehicle-registration` contabili dal libro contabile in un bucket Amazon S3. Quindi, si utilizzano i dati esportati per convalidare la catena di hash tra i blocchi di journal e i singoli componenti hash all'interno di ciascun blocco.

L'entità principaleAWS Identity and Access Management (IAM) che utilizzi deve disporre di autorizzazioni IAM sufficienti per creare un bucket Amazon S3 nel tuoAccount AWS. Per informazioni, consulta Policy e autorizzazioni in Amazon S3 nella Guida per l'utente di Amazon S3. Devi inoltre disporre delle autorizzazioni per creare un ruolo IAM con una politica di autorizzazioni allegata che consenta a QLDB di scrivere oggetti nel tuo bucket Amazon S3. Per ulteriori informazioni, consulta Autorizzazioni necessarie per accedere alle risorse IAM nella Guida per l'utente di IAM.

Per esportare e convalidare i dati del giornale

1. Esaminate il seguente file (`JournalBlock.java`), che rappresenta un blocco di giornale e il relativo contenuto di dati. Include un metodo denominato`verifyBlockHash()` che dimostra come calcolare ogni singolo componente di un hash di blocco.

```
/*
 * 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. Compila ed esegui il seguente programma (`ValidateQldbHashChain.java`) per eseguire i seguenti passaggi:

   1. Esporta i blocchi di scritture`vehicle-registration` contabili dal libro contabile in un bucket Amazon S3 denominato**`qldb-tutorial-journal-export-`*`111122223333`*** (sostituiscilo con il tuoAccount AWS numero).
   2. Convalida i singoli componenti hash all'interno di ogni blocco chiamando`verifyBlockHash()`.
   3. Convalida la catena di hash tra i blocchi del 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;
        }
    }

}
```

Se non è più necessario utilizzare il`vehicle-registration` libro mastro, procedere aFase 9 (opzionale): eliminazione delle risorse.

Fase 9 (opzionale): eliminazione delle risorse

Puoi continuare a usare il`vehicle-registration` libro mastro. Tuttavia, se non è più necessario, è necessario eliminarlo.

Per eliminare il libro mastro

1. Compila ed esegui il seguente programma (`DeleteLedger.java`) per eliminare il`vehicle-registration` libro mastro e tutto il suo contenuto.

```
/*
 * 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**
>
> Se la protezione dall'eliminazione è abilitata per il libro mastro, è necessario innanzitutto disabilitarla prima di poter eliminare il libro mastro utilizzando l'API QLDB.

2. Se hai esportato i dati del journal nel passaggio precedente e non ne hai più bisogno, utilizza la console Amazon S3 per eliminare il bucket S3.

   Apri la console di Amazon S3 su https://console.aws.amazon.com/s3/.

Tutorial su Amazon QLDB Node.js

In questa implementazione dell'applicazione di esempio del tutorial, utilizzi il driver Amazon QLDB con AWS l'SDK JavaScript per Node.js per creare un registro QLDB e compilarlo con dati di esempio.

Mentre segui questo tutorial, puoi fare riferimento all'API Reference per le AWS SDK for JavaScript operazioni API di gestione. Per le operazioni sui dati transazionali, puoi fare riferimento al QLDB Driver for Node.js API Reference.

> **Note**
>
> Ove applicabile, alcuni passaggi del tutorial contengono comandi o esempi di codice diversi per ogni versione principale supportata del driver QLDB per Node.js.

Argomenti

- Installazione dell'applicazione di esempio Amazon QLDB Node.js
- Fase 1: Creare un nuovo libro mastro
- Passaggio 2: verifica la connettività al registro

- Passaggio 3: Creare tabelle, indici e dati di esempio
- Passaggio 4: interrogare le tabelle in un libro mastro
- Fase 5: Modificare i documenti in un libro mastro
- Fase 6: Visualizzare la cronologia delle revisioni di un documento
- Passaggio 7: Verificare un documento in un libro mastro
- Passaggio 8 (opzionale): Pulisci le risorse

Installazione dell'applicazione di esempio Amazon QLDB Node.js

Questa sezione descrive come installare ed eseguire l'applicazione di esempio Amazon QLDB fornita per step-by-step il tutorial Node.js. Il caso d'uso di questa applicazione di esempio è un database del Dipartimento dei veicoli a motore (DMV) che tiene traccia delle informazioni storiche complete sulle immatricolazioni dei veicoli.

L'applicazione di esempio DMV per Node.js è open source nel GitHub repository aws-samples/ - nodejs. amazon-qldb-dmv-sample

Prerequisiti

Prima di iniziare, assicurati di aver completato il driver QLDB per Node.js. Prerequisiti Ciò include l'installazione di Node.js e le seguenti operazioni:

1. Iscriviti perAWS.
2. Crea un utente con le autorizzazioni QLDB appropriate.
3. Concedi l'accesso programmatico per lo sviluppo.

Per completare tutti i passaggi di questo tutorial, è necessario l'accesso amministrativo completo alla risorsa di registro tramite l'API QLDB.

Installazione

Per installare l'applicazione di esempio

1. Immettete il seguente comando da cui clonare l'applicazione di GitHub esempio.

   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'applicazione di esempio racchiude il codice sorgente completo di questo tutorial e le relative dipendenze, inclusi il driver Node.js e l'AWSSDK per JavaScript Node.js. Questa applicazione è scritta in. TypeScript

2. Passa alla directory in cui viene clonato il `amazon-qldb-dmv-sample-nodejs` pacchetto.

```
cd amazon-qldb-dmv-sample-nodejs
```

3. Esegui un'installazione pulita delle dipendenze.

```
npm ci
```

4. Transpila il pacchetto.

```
npm run build
```

I JavaScript file transpilati vengono scritti nella directory. `./dist`

5. Procedi con l'Fase 1: Creare un nuovo libro mastroavvio del tutorial e la creazione di un libro mastro.

Fase 1: Creare un nuovo libro mastro

In questo passaggio, crei un nuovo registro Amazon QLDB denominato. `vehicle-registration`

Per creare un nuovo registro

1. Esamina il seguente file (`Constants.ts`), che contiene valori costanti utilizzati da tutti gli altri programmi di questo tutorial.

```
/*
 * 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. Utilizzate il seguente programma (`CreateLedger.ts`) per creare un registro denominato`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**
>
> - Nella `createLedger` chiamata, è necessario specificare un nome di registro e una modalità di autorizzazione. Ti consigliamo di utilizzare la modalità `STANDARD` autorizzazioni per massimizzare la sicurezza dei dati del registro.
> - Quando si crea un registro, la protezione da eliminazione è abilitata per impostazione predefinita. Questa è una funzionalità di QLDB che impedisce l'eliminazione dei registri da parte di qualsiasi utente. Hai la possibilità di disabilitare la protezione da eliminazione durante la creazione del registro utilizzando l'API QLDB o il (). AWS Command Line Interface AWS CLI
> - Facoltativamente, puoi anche specificare i tag da allegare al tuo libro mastro.

3. Per eseguire il programma transpiled, immettete il seguente comando.

```
node dist/CreateLedger.js
```

Per verificare la connessione al nuovo registro, procedi a. Passaggio 2: verifica la connettività al registro

Passaggio 2: verifica la connettività al registro

In questo passaggio, verifichi di poterti connettere al `vehicle-registration` registro in Amazon QLDB utilizzando l'endpoint API dei dati transazionali.

Per testare la connettività al registro

1. Utilizzate il seguente programma (`ConnectToLedger.ts`) per creare una connessione di sessione dati al `vehicle-registration` registro.

   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**
>
> Per eseguire transazioni di dati sul registro, è necessario creare un oggetto driver QLDB per connettersi a un registro specifico. Si tratta di un oggetto client diverso dall'`qldbClient`oggetto utilizzato nel passaggio precedente per creare il registro. Quel

client precedente viene utilizzato solo per eseguire le operazioni dell'API di gestione elencate in. Documentazione di riferimento dell'API Amazon QLDB

2. Per eseguire il programma transpiled, immettete il seguente comando.

```
node dist/ConnectToLedger.js
```

Per creare tabelle nel `vehicle-registration` registro, procedi a. Passaggio 3: Creare tabelle, indici e dati di esempio

Passaggio 3: Creare tabelle, indici e dati di esempio

Quando il tuo registro Amazon QLDB è attivo e accetta connessioni, puoi iniziare a creare tabelle per i dati sui veicoli, i loro proprietari e le loro informazioni di immatricolazione. Dopo aver creato le tabelle e gli indici, puoi caricarli con i dati.

In questo passaggio, crei quattro tabelle nel `vehicle-registration` registro:

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

Vengono inoltre creati i seguenti indici.

Nome tabella	Campo
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Quando si inseriscono dati di esempio, si inseriscono innanzitutto i documenti nella tabella. `Person` Quindi, si utilizzano i dati assegnati `id` dal sistema a ciascun `Person` documento per compilare i campi corrispondenti nei documenti e nei documenti appropriati. `VehicleRegistration` `DriversLicense`

> **Tip**
>
> Come procedura ottimale, utilizzate il sistema assegnato a un documento come chiave esterna`id`. Sebbene sia possibile definire campi destinati a essere identificatori univoci (ad esempio, il VIN di un veicolo), il vero identificatore univoco di un documento è il suo. `id` Questo campo è incluso nei metadati del documento, a cui è possibile eseguire una query nella visualizzazione confermata (la vista di una tabella definita dal sistema).
> Per ulteriori informazioni sulle viste in QLDB, vedere. Concetti principali Per ulteriori informazioni sui metadati, consulta. Interrogazione dei metadati dei documenti

Per creare tabelle e indici

1. Utilizzate il seguente programma (`CreateTable.ts`) per creare le tabelle menzionate in precedenza.

```
/*
 * 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**
>
> Questo programma dimostra come utilizzare la `executeLambda` funzione in un'istanza del driver QLDB. In questo esempio, esegui più istruzioni `CREATE TABLE` PartiQL con una singola espressione lambda.
> Questa funzione di esecuzione avvia implicitamente una transazione, esegue tutte le istruzioni in lambda e quindi esegue automaticamente il commit della transazione.

2. Per eseguire il programma transpiled, immettete il seguente comando.

```
node dist/CreateTable.js
```

3. Utilizzate il seguente programma (`CreateIndex.ts`) per creare indici sulle tabelle, come descritto in precedenza.

```
/*
 * 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. Per eseguire il programma transpiled, immettete il seguente comando.

```
node dist/CreateIndex.js
```

Per caricare i dati nelle tabelle

1. Esamina i seguenti `.ts` file.

    1. `SampleData.ts`— Contiene i dati di esempio da inserire nelle `vehicle-registration` tabelle.

```
/*
 * 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 modulo di utilità che importa dal `ion-js` pacchetto per fornire funzioni di supporto che convertono, analizzano e stampano i dati di 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` funzione esegue una query che restituisce gli ID dei documenti assegnati dal sistema da una tabella. Per ulteriori informazioni, vedi Utilizzo della clausola BY per interrogare l'ID del documento.

2. Utilizzate il seguente programma (`InsertDocument.ts`) per inserire i dati di esempio nelle tabelle.

```
/*
 * 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

- Questo programma dimostra come chiamare la `execute` funzione con valori parametrizzati. È possibile passare parametri di dati oltre all'istruzione PartiQL che

si desidera eseguire. Utilizzate un punto interrogativo (?) come segnaposto variabile nella stringa dell'istruzione.

- Se un'`INSERT`istruzione ha esito positivo, restituisce il valore `id` di ogni documento inserito.

3. Per eseguire il programma transpiled, immettete il seguente comando.

```
node dist/InsertDocument.js
```

Successivamente, è possibile utilizzare `SELECT` le istruzioni per leggere i dati dalle tabelle del registro. `vehicle-registration` Passa a Passaggio 4: interrogare le tabelle in un libro mastro.

Passaggio 4: interrogare le tabelle in un libro mastro

Dopo aver creato tabelle in un registro Amazon QLDB e averle caricate con i dati, puoi eseguire query per rivedere i dati di immatricolazione dei veicoli che hai appena inserito. QLDB utilizza PartiQL come linguaggio di interrogazione e Amazon Ion come modello di dati orientato ai documenti.

PartiQL è un linguaggio di query open source compatibile con SQL che è stato esteso per funzionare con Ion. Con PartiQL, puoi inserire, interrogare e gestire i tuoi dati con operatori SQL familiari. Amazon Ion è un superset di JSON. Ion è un formato di dati open source basato su documenti che offre la flessibilità di archiviazione ed elaborazione di dati strutturati, semistrutturati e annidati.

In questo passaggio, si utilizzano `SELECT` le istruzioni per leggere i dati dalle tabelle del registro. `vehicle-registration`

Warning

Quando si esegue una query in QLDB senza una ricerca indicizzata, viene richiamata una scansione completa della tabella. PartiQL supporta tali query perché è compatibile con SQL. Tuttavia, non eseguire scansioni di tabelle per casi d'uso di produzione in QLDB. Le scansioni delle tabelle possono causare problemi di prestazioni su tabelle di grandi dimensioni, inclusi conflitti di concorrenza e timeout delle transazioni.

Per evitare le scansioni delle tabelle, è necessario eseguire istruzioni con una clausola di `WHERE` predicato utilizzando un operatore di uguaglianza su un campo indicizzato o un ID di documento; ad esempio, o. `WHERE indexedField = 123` `WHERE indexedField IN (456, 789)` Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.

Per interrogare le tabelle

1. Usa il seguente programma (`FindVehicles.ts`) per interrogare tutti i veicoli registrati con una persona nel tuo registro.

```
/*
 * 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**
>
> Innanzitutto, questo programma interroga la `Person` tabella del documento per ottenere il relativo `GovId LEWISR261LL` campo di `id` metadati.
> Quindi, utilizza questo documento `id` come chiave esterna per interrogare la `VehicleRegistration` tabella. `PrimaryOwner.PersonId` Inoltre si unisce `VehicleRegistration` alla `Vehicle` tabella sul `VIN` campo.

2. Per eseguire il programma transpiled, inserisci il seguente comando.

```
node dist/FindVehicles.js
```

Per ulteriori informazioni sulla modifica dei documenti nelle tabelle del `vehicle-registration` registro, vedere. Fase 5: Modificare i documenti in un libro mastro

Fase 5: Modificare i documenti in un libro mastro

Ora che hai dei dati su cui lavorare, puoi iniziare a modificare i documenti nel `vehicle-registration` registro di Amazon QLDB. In questa fase, i seguenti esempi di codice mostrano come eseguire istruzioni DML (Data Manipulation Language). Queste istruzioni aggiornano il proprietario principale di un veicolo e aggiungono un proprietario secondario a un altro veicolo.

Per modificare i documenti

1. Utilizzate il seguente programma (`TransferVehicleOwnership.ts`) per aggiornare il proprietario principale del veicolo inserendo il `1N4AL11D75C109151` VIN nel registro.

```
/*
 * 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. Per eseguire il programma transpiled, immettete il seguente comando.

```
node dist/TransferVehicleOwnership.js
```

3. Utilizzate il seguente programma (`AddSecondaryOwner.ts`) per aggiungere un proprietario secondario al veicolo con VIN `KM8SRDHF6EU074761` nel registro.

```
/*
 * 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. Per eseguire il programma transpiled, immettete il seguente comando.

```
node dist/AddSecondaryOwner.js
```

Per esaminare queste modifiche nel `vehicle-registration` registro, vedere. Fase 6: Visualizzare la cronologia delle revisioni di un documento

Fase 6: Visualizzare la cronologia delle revisioni di un documento

Dopo aver modificato i dati di immatricolazione di un veicolo nel passaggio precedente, puoi interrogare la cronologia di tutti i proprietari registrati e qualsiasi altro campo aggiornato. In questo passaggio, esegui una query sulla cronologia delle revisioni di un documento nella `VehicleRegistration` tabella del registro. `vehicle-registration`

Per visualizzare la cronologia delle revisioni

1. Utilizzate il seguente programma (`QueryHistory.ts`) per interrogare la cronologia delle revisioni del `VehicleRegistration` documento con 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**
>
> - È possibile visualizzare la cronologia delle revisioni di un documento interrogando la sintassi incorporata Funzione di cronologia nella seguente sintassi.
>
>   ```
>   SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
>   [ WHERE h.metadata.id = 'id' ]
>   ```
>
> - L'ora di inizio e l'ora di fine sono entrambe opzionali. Sono valori letterali di Amazon Ion che possono essere indicati con backticks (). `` `...` `` Per ulteriori informazioni, consulta Interrogazione di Ion con PartiQL in Amazon QLDB.
> - Come best practice, qualifica una query cronologica con un intervallo di date (ora di inizio e ora di fine) e un ID del documento (). `metadata.id` QLDB SELECT elabora le query nelle transazioni, che sono soggette a un limite di timeout delle transazioni.
>
>   La cronologia QLDB è indicizzata in base all'ID del documento e al momento non è possibile creare indici di cronologia aggiuntivi. Le interrogazioni cronologiche che includono l'ora di inizio e l'ora di fine ottengono il vantaggio della qualificazione per intervalli di date.

2. Per eseguire il programma transpiled, immettete il seguente comando.

```
node dist/QueryHistory.js
```

Per verificare crittograficamente una revisione del documento nel registro, procedi a. `vehicle-registration` Passaggio 7: Verificare un documento in un libro mastro

Passaggio 7: Verificare un documento in un libro mastro

Con Amazon QLDB, puoi verificare in modo efficiente l'integrità di un documento nel diario del tuo libro mastro utilizzando l'hashing crittografico con SHA-256. Per ulteriori informazioni su come funzionano la verifica e l'hashing crittografico in QLDB, consulta. Verifica dei dati in Amazon QLDB

In questo passaggio, verifichi la revisione di un documento nella tabella del `VehicleRegistration` registro. `vehicle-registration` Innanzitutto, richiedi un digest, che viene restituito come file di output e funge da firma dell'intera cronologia delle modifiche del registro. Quindi, richiedi una bozza della revisione relativa a quel digest. Utilizzando questa prova, l'integrità della revisione viene verificata se tutti i controlli di convalida vengono superati.

Per verificare la revisione di un documento

1. Esamina i seguenti `.ts` file, che contengono gli oggetti QLDB necessari per la verifica.

    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. Utilizzate due `.ts` programmi (`GetDigest.tseGetRevision.ts`) per eseguire le seguenti operazioni:

   - Richiedete un nuovo digest dal `vehicle-registration` registro.
   - Richiedi una bozza per ogni revisione del documento con VIN 1N4AL11D75C109151 dalla tabella. `VehicleRegistration`
   - Verifica le revisioni utilizzando il digest restituito e verifica ricalcolando il digest.

   Il `GetDigest.ts` programma contiene il codice seguente.

```
/*
 * 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**
>
> Usa la `getDigest` funzione per richiedere un riassunto che copra la punta corrente del diario nel tuo libro mastro. Il tip of the journal si riferisce all'ultimo blocco eseguito nel momento in cui QLDB riceve la tua richiesta.

Il `GetRevision.ts` programma contiene il seguente codice.

```
/*
 * 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**
>
> Dopo che la `getRevision` funzione ha restituito una bozza per la revisione del documento specificata, questo programma utilizza un'API lato client per verificare tale revisione.

3. Per eseguire il programma transpiled, immettete il seguente comando.

```
node dist/GetRevision.js
```

Se non hai più bisogno di usare il `vehicle-registration` libro mastro, procedi a. Passaggio 8 (opzionale): Pulisci le risorse

Passaggio 8 (opzionale): Pulisci le risorse

Puoi continuare a utilizzare il libro mastro. `vehicle-registration` Tuttavia, se non è più necessario, è necessario eliminarlo.

Per eliminare il libro mastro

1. Utilizzate il seguente programma (`DeleteLedger.ts`) per eliminare il `vehicle-registration` registro e tutto il suo contenuto.

```
/*
 * 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**
>
> Se la protezione da eliminazione è abilitata per il registro, devi prima disabilitarla prima di poter eliminare il registro utilizzando l'API QLDB.

2. Per eseguire il programma transpiled, inserisci il seguente comando.

```
node dist/DeleteLedger.js
```

Tutorial di Amazon QLDB di Python

In questa implementazione dell'applicazione di esempio del tutorial, si utilizza il driver Amazon QLDB con ilAWS SDK for Python (Boto3) per creare un registro QLDB e compilarlo con dati di esempio.

Man mano che procedi in questo tutorial, puoi fare riferimento al client di basso livello QLDB nell'API ReferenceAWS SDK for Python (Boto3) per le operazioni API di gestione. Per le operazioni sui dati transazionali, puoi fare riferimento al QLDB Driver for Python API Reference.

> **Note**
>
> Ove applicabile, alcuni passaggi del tutorial hanno comandi o esempi di codice diversi per ciascuna versione principale supportata del driver QLDB per Python.

Argomenti

Installazione dell'applicazione di esempio Amazon QLDB Python

Questa sezione descrive come installare ed eseguire l'applicazione di esempio Amazon QLDB fornita per il tutorial step-by-step Python. Il caso d'uso di questa applicazione di esempio è un database del dipartimento dei veicoli a motore (DMV) che tiene traccia delle informazioni storiche complete sulle immatricolazioni dei veicoli.

L'applicazione di esempio DMV per Python è open source nel GitHub repository aws-samples/amazon-qldb-dmv-sample -python.

Prerequisiti

Prima di iniziare, assicurati di aver completato il driver QLDB per PythonPrerequisiti. Ciò include l'installazione di Python e le seguenti operazioni:

1. Registrazione aAWS.
2. Crea un utente con le autorizzazioni QLDB appropriate.
3. Concedi l'accesso programmatico per lo sviluppo.

Per completare tutti i passaggi di questo tutorial, è necessario l'accesso amministrativo completo alla risorsa contabile tramite l'API QLDB.

Installazione

Per installare l'applicazione di esempio

1. Immettete il seguentepip comando per clonare e installare l'applicazione di esempio da 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'applicazione di esempio include il codice sorgente completo di questo tutorial e le sue dipendenze, inclusi il driver Python e il AWS SDK for Python (Boto3).
2. Prima di iniziare a eseguire il codice dalla riga di comando, spostate la directory di lavoro corrente nella posizione in cui è installato ilpyqldbsamples pacchetto. Inserire il seguente comando.

   ```
   cd $(python -c "import pyqldbsamples; print(pyqldbsamples.__path__[0])")
   ```
3. ProcediFase 1: Creazione di un nuovo libro mastro con l'avvio del tutorial e crea un libro mastro.

Fase 1: Creazione di un nuovo libro mastro

In questo passaggio, crei un nuovo registro Amazon QLDB denominatovehicle-registration.

Per creare un nuovo libro mastro

1. Esaminate il seguente file (constants.py), che contiene valori costanti utilizzati da tutti gli altri programmi di questo tutorial.

   ```
   # 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. Utilizzate il seguente programma (`create_ledger.py`) per creare un libro contabile denominato`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**
>
> - Nella`create_ledger` chiamata, è necessario specificare un nome contabile e una modalità di autorizzazione. Ti consigliamo di utilizzare la modalità di`STANDARD` autorizzazione per massimizzare la sicurezza dei dati nel libro mastro.
> - Quando si crea un libro mastro, la protezione da eliminazione è abilitata per impostazione predefinita. Questa è una funzionalità di QLDB che impedisce ai registri di eliminare i registri da parte di un utente qualsiasi di eliminare i registri. È possibile disabilitare la protezione dall'eliminazione durante la creazione del libro contabile utilizzando l'API QLDB o ilAWS Command Line Interface (AWS CLI).
> - Eventualmente, è anche possibile specificare i tag da allegare al libro mastro.

3. Per eseguire il programma, immetti il comando seguente:

```
python create_ledger.py
```

Per verificare la tua connessione al nuovo libro mastro, procedi aFase 2: Test della connettività al libro mastro.

Fase 2: Test della connettività al libro mastro

In questo passaggio, verifichi di poterti connettere al`vehicle-registration` libro mastro in Amazon QLDB utilizzando l'endpoint dell'API dei dati transazionali.

Per testare la connettività al libro mastro

1. Utilizzate il seguente programma (`connect_to_ledger.py`) per creare una connessione di sessione dati al`vehicle-registration` libro mastro.

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**
>
> - Per eseguire transazioni di dati sul libro mastro, è necessario creare un oggetto driver QLDB da connettere a un libro mastro specifico. Si tratta di un oggetto client diverso da quello utilizzato nel passaggio precedente per creare il libro contabile.`qldb_client` Il client precedente viene utilizzato solo per eseguire le operazioni dell'API di gestione elencate in[Documentazione di riferimento dell'API Amazon QLDB](#).
> - È necessario specificare un nome contabile quando si crea questo oggetto driver.

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**
>
> - Per eseguire transazioni di dati sul libro mastro, è necessario creare un oggetto driver QLDB da connettere a un libro mastro specifico. Si tratta di un oggetto client diverso da quello utilizzato nel passaggio precedente per creare il libro contabile.`qldb_client` Il client precedente viene utilizzato solo per eseguire le operazioni dell'API di gestione elencate in[Documentazione di riferimento dell'API Amazon QLDB](#).
> - Innanzitutto, create un oggetto driver QLDB in pool. È necessario specificare un nome contabile quando si crea questo driver.
> - Quindi, è possibile creare sessioni da questo oggetto driver in pool.

2. Per eseguire il programma, immetti il comando seguente:

```
python connect_to_ledger.py
```

Per creare tabelle nel`vehicle-registration` registro, procedere a[Fase 3: creazione di tabelle, indici e dati di esempio](#).

Fase 3: creazione di tabelle, indici e dati di esempio

Quando il tuo registro Amazon QLDB è attivo e accetta connessioni, puoi iniziare a creare tabelle con i dati sui veicoli, i loro proprietari e le loro informazioni di registrazione. Dopo aver creato le tabelle e gli indici, è possibile caricarli con i dati.

In questa fase, creerai quattro tabelle nel`vehicle-registration` libro mastro:

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

È inoltre possibile creare i seguenti indici.

Nome tabella	Campo
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Quando si inseriscono dati di esempio, si inseriscono innanzitutto i documenti nella`Person` tabella. Quindi, si utilizza il sistema assegnato`id` da ciascun`Person` documento per compilare i campi corrispondenti nei`DriversLicense` documenti`VehicleRegistration` e nei documenti appropriati.

> **Tip**
>
> Come procedura consigliata, utilizzare`id` come chiave esterna assegnata dal sistema di un documento. Sebbene sia possibile definire campi destinati a essere identificatori univoci (ad esempio, il VIN di un veicolo), il vero identificatore univoco di un documento è il suo`id`. Questo campo è incluso nei metadati del documento, che è possibile interrogare nella vista commit (la vista definita dal sistema di una tabella).
> Per ulteriori informazioni sulle visualizzazioni in QLDB, consultaConcetti principali. Per ulteriori informazioni sui metadati, consultaInterrogazione dei metadati dei documenti.

Per creare tabelle e indici

1. Utilizzate il seguente programma (`create_table.py`) per creare le tabelle menzionate in precedenza.

   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**
>
> Questo programma illustra come utilizzare la`execute_lambda` funzione. In questo esempio, si eseguono più istruzioni`CREATE TABLE` PartiQL con una singola espressione lambda.
> Questa funzione di esecuzione avvia implicitamente una transazione, esegue tutte le istruzioni nella lambda e quindi esegue automaticamente la transazione.

2. Per eseguire il programma, immetti il comando seguente:

```
python create_table.py
```

3. Usate il seguente programma (`create_index.py`) per creare indici sulle tabelle, come descritto in precedenza.

   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. Per eseguire il programma, immetti il comando seguente:

```
python create_index.py
```

Per caricare i dati nelle tabelle

1. Esaminate il seguente file (`sample_data.py`), che rappresenta i dati di esempio che inserite nelle`vehicle-registration` tabelle. Questo file viene inoltre importato dala`amazon.ion` pacchetto per fornire funzioni di supporto che convertono, analizzano e stampano i dati di 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` funzione esegue una query che restituisce gli ID dei documenti assegnati dal sistema da una tabella. Per ulteriori informazioni, consulta Utilizzo della clausola BY per interrogare l'ID del documento.

2. Usa il seguente programma (`insert_document.py`) per inserire i dati di esempio nelle tue tabelle.

   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**
>
> - Questo programma dimostra come richiamare la`execute_statement` funzione con valori parametrizzati. È possibile passare parametri di dati oltre all'istruzione PartiQL che si desidera eseguire. Usa un punto interrogativo (?) come segnaposto variabile nella stringa del rendiconto.
> - Se un'`INSERT`istruzione ha successo, restituisce il valore`id` di ogni documento inserito.

3. Per eseguire il programma, immetti il comando seguente:

```
python insert_document.py
```

Successivamente, è possibile utilizzare`SELECT` le dichiarazioni per leggere i dati dalle tabelle del`vehicle-registration` libro mastro. Continua con la Fase 4: Interrogare le tabelle in un libro mastro.

Fase 4: Interrogare le tabelle in un libro mastro

Dopo aver creato le tabelle in un registro Amazon QLDB e averle caricate con i dati, puoi eseguire interrogazioni per esaminare i dati di immatricolazione del veicolo che hai appena inserito. QLDB utilizza PartiQL come linguaggio di interrogazione e Amazon Ion come modello di dati orientato ai documenti.

PartiQL è un linguaggio di interrogazione open source e compatibile con SQL che è stato esteso per funzionare con Ion. Con PartiQL, puoi inserire, interrogare e gestire i tuoi dati con operatori SQL familiari. Amazon Ion è un superset di JSON. Ion è un formato di dati open source basato su documenti che offre la flessibilità di archiviare ed elaborare dati strutturati, semistrutturati e annidati.

In questo passaggio, si utilizzano i`SELECT` rendiconti per leggere i dati dalle tabelle del`vehicle-registration` libro mastro.

> **Warning**
>
> Quando si esegue una query in QLDB senza una ricerca indicizzata, viene richiamata una scansione completa della tabella. PartiQL supporta tali interrogazioni perché è compatibile con SQL. Tuttavia, non eseguite scansioni di tabelle per casi d'uso di produzione in QLDB. Le scansioni delle tabelle possono causare problemi di prestazioni su tabelle di grandi dimensioni, inclusi conflitti di concorrenza e timeout delle transazioni.
> Per evitare la scansione delle tabelle, è necessario eseguire istruzioni con una clausola`WHERE` predicativa utilizzando un operatore di uguaglianza su un campo indicizzato o un ID di documento; ad esempio,`WHERE indexedField = 123` o`WHERE indexedField IN (456, 789)`. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.

Per interrogare le tabelle

1. Utilizza il seguente programma (`find_vehicles.py`) per interrogare tutti i veicoli registrati a nome di una persona nel tuo registro.

   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**
>
> Innanzitutto, questo programma interroga la`Person` tabella del documento per`GovId` `LEWISR261LL` ottenere il relativo campo di`id` metadati.

Quindi, utilizza questo documento`oid` come chiave esterna per interrogare la`VehicleRegistration` tabella`PrimaryOwner.PersonId`. Inoltre si`VehicleRegistration` unisce al`Vehicle` tavolo sul`VIN` campo.

2. Per eseguire il programma, immetti il comando seguente:

```
python find_vehicles.py
```

Per informazioni sulla modifica dei documenti nelle tabelle del`vehicle-registration` libro mastro, vedereFase 5: Modificare i documenti in un libro mastro.

Fase 5: Modificare i documenti in un libro mastro

Ora che hai dati su cui lavorare, puoi iniziare a modificare i documenti nel`vehicle-registration` registro in Amazon QLDB. In questa fase, i seguenti esempi di codice illustrano come eseguire istruzioni DML (Data Manipulation Language). Queste dichiarazioni aggiornano il proprietario principale di un veicolo e aggiungono un proprietario secondario a un altro veicolo.

Per modificare i documenti

1. Utilizza il seguente programma (`transfer_vehicle_ownership.py`) per aggiornare il proprietario principale del veicolo con il`1N4AL11D75C109151` VIN nel tuo registro.

   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. Per eseguire il programma, immetti il comando seguente:

```
python transfer_vehicle_ownership.py
```

3. Utilizza il seguente programma (`add_secondary_owner.py`) per aggiungere un proprietario secondario al veicolo con il`KM8SRDHF6EU074761` VIN nel registro.

   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. Per eseguire il programma, immetti il comando seguente:

```
python add_secondary_owner.py
```

Per esaminare queste modifiche nelvehicle-registration registro, vederePassaggio 6: visualizzare la cronologia delle revisioni di un documento.

Passaggio 6: visualizzare la cronologia delle revisioni di un documento

Dopo aver modificato i dati di registrazione di un veicolo nel passaggio precedente, è possibile interrogare la cronologia di tutti i proprietari registrati e qualsiasi altro campo aggiornato. In questo passaggio, è necessario interrogare la cronologia delle revisioni di un documento nellaVehicleRegistration tabella delvehicle-registration libro contabile.

Per visualizzare la cronologia delle revisioni

1. Usa il seguente programma (query_history.py) per interrogare la cronologia delle revisioni delVehicleRegistration documento con 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**
>
> - È possibile visualizzare la cronologia delle revisioni di un documento interrogando la sintassi integrataFunzione di cronologia nella seguente sintassi.
>
>   ```
>   SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
>   [ WHERE h.metadata.id = 'id' ]
>   ```
>
> - L'ora di inizio e l'ora di fine sono entrambe opzionali. Sono valori letterali di Amazon Ion che possono essere indicati con delle barrette inverse (`` `...` ``). Per ulteriori informazioni, consulta Interrogazione di Ion con PartiQL in Amazon QLDB.
> - Come procedura consigliata, qualifica un'interrogazione cronologica con un intervallo di date (ora di inizio e ora di fine) e un ID del documento (`metadata.id`). QLDB elabora le`SELECT` interrogazioni nelle transazioni, che sono soggette a un limite di timeout delle transazioni.
>
>   La cronologia QLDB è indicizzata in base all'ID del documento e al momento non è possibile creare indici di cronologia aggiuntivi. Le interrogazioni cronologiche che includono un'ora di inizio e un'ora di fine ottengono il vantaggio della qualificazione dell'intervallo di date.

2. Per eseguire il programma, immetti il comando seguente:

```
python query_history.py
```

Per verificare crittograficamente una revisione del documento nel`vehicle-registration` libro mastro, procedere aFase 7: Verificare un documento in un libro mastro.

Fase 7: Verificare un documento in un libro mastro

Con Amazon QLDB, puoi verificare in modo efficiente l'integrità di un documento nel giornale contabile utilizzando l'hashing crittografico con SHA-256. Per ulteriori informazioni su come funzionano la verifica e l'hashing crittografico in QLDB, consultaVerifica dei dati in Amazon QLDB.

In questo passaggio, si verifica una revisione del documento nella`VehicleRegistration` tabella del`vehicle-registration` libro contabile. Innanzitutto, richiedi un riassunto, che viene restituito come file di output e funge da firma dell'intera cronologia delle modifiche del tuo libro contabile. Quindi, richiedi una prova della revisione relativa a quel riassunto. Utilizzando questa prova, l'integrità della revisione viene verificata se tutti i controlli di convalida vengono superati.

Per verificare la revisione di un documento

1. Esamina i seguenti`.py` file, che rappresentano gli oggetti QLDB necessari per la verifica e un modulo di utilità con funzioni di supporto per convertire i tipi di risposta QLDB in stringhe.

   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. Usa due `.py` programmi (`get_digest.py`e`get_revision.py`) per eseguire i seguenti passaggi:

   - Richiedi un nuovo riassunto dal`vehicle-registration` registro.
   - Richiedi una prova per ogni revisione del documento con VIN`1N4AL11D75C109151` dalla`VehicleRegistration` tabella.
   - Verifica le revisioni utilizzando il riassunto e la prova restituiti ricalcolando il riassunto.

   Il`get_digest.py` programma contiene il seguente codice.

```
# 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**
>
> Usa la`get_digest_result` funzione per richiedere un riassunto che copra la parte corrente del diario nel tuo libro contabile. Il suggerimento del diario si riferisce all'ultimo blocco salvato al momento in cui QLDB riceve la tua richiesta.

Il`get_revision.py` programma contiene il seguente codice.

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**
>
> Dopo che la`get_revision` funzione restituisce una prova della revisione del documento specificata, questo programma utilizza un'API lato client per verificare tale revisione.

3. Per eseguire il programma, immetti il comando seguente:

```
python get_revision.py
```

Se non è più necessario utilizzare il`vehicle-registration` libro mastro, procedere aFase 8 (opzionale): eliminazione delle risorse.

Fase 8 (opzionale): eliminazione delle risorse

Puoi continuare a usare il`vehicle-registration` libro mastro. Tuttavia, se non ne hai più bisogno, dovresti eliminarlo.

Per eliminare il libro mastro

1. Utilizzate il seguente programma (`delete_ledger.py`) per eliminare il`vehicle-registration` libro contabile e tutto il suo contenuto.

```
# 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**
>
> Se la protezione da eliminazione è abilitata per il libro mastro, è necessario innanzitutto disabilitarla prima di poter eliminare la contabilità utilizzando l'API QLDB.

Il`delete_ledger.py` file dipende anche dal seguente programma (`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. Per eseguire il programma, immetti il comando seguente:

```
python delete_ledger.py
```

Utilizzo dei tipi di dati Amazon Ion in Amazon QLDB

Amazon QLDB archivia i propri dati nel formato Amazon Ion. Per lavorare con i dati in QLDB, è necessario utilizzare una libreria Ion come dipendenza per un linguaggio di programmazione supportato.

In questa sezione, scopri come convertire i dati dai tipi nativi ai loro equivalenti ionici e viceversa. Questa guida di riferimento mostra esempi di codice che utilizzano il driver QLDB per elaborare i dati Ion in un registro QLDB. Include esempi di codice per Java, .NET (C#) Go, Node.js (TypeScript) e Python.

Argomenti

- Prerequisiti
- Bool
- Int

- Float
- Decimale
- Time stamp
- Stringa
- Blob
- Elenco
- Struct
- Valori nulli e tipi dinamici
- Conversione verso il basso in JSON

Prerequisiti

I seguenti esempi di codice presuppongono che si disponga di un'istanza del driver QLDB connessa a un libro mastro attivo con una tabella denominata`ExampleTable`. La tabella contiene un singolo documento esistente che contiene i seguenti otto campi:

- ExampleBool
- ExampleInt
- ExampleFloat
- ExampleDecimal
- ExampleTimestamp
- ExampleString
- ExampleBlob
- ExampleList

> **Note**
>
> Ai fini di questo riferimento, si supponga che il tipo memorizzato in ogni campo corrisponda al suo nome. In pratica, QLDB non applica le definizioni di schemi o tipi di dati per i campi del documento.

Bool

Gli esempi di codice seguenti mostrano come elaborare il tipo booleano 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**
>
> Per convertire in codice sincrono, rimuovi`async` le parole chiave`await` e cambia il`IAsyncResult` tipo in`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

Gli esempi di codice seguenti mostrano come elaborare il tipo di numero intero 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**
>
> Per convertire in codice sincrono, rimuovi`async` le parole chiave`await` e cambia il`IAsyncResult` tipo in`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

Gli esempi di codice seguenti mostrano come elaborare il tipo 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

Usare 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**
>
> Per convertire in codice sincrono, rimuovi`async` le parole chiave`await` e cambia il`IAsyncResult` tipo in`IResult`.

Usare 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**
>
> Per convertire in codice sincrono, rimuovi`async` le parole chiave`await` e cambia il`IAsyncResult` tipo in`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))
```

Decimale

Gli esempi di codice seguenti mostrano come elaborare il tipo decimale 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**
>
> Per convertire in codice sincrono, rimuovi`async` le parole chiave`await` e cambia il`IAsyncResult` tipo in`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**
>
> Puoi anche usarloionValue.numberValue() per trasformare un decimale Ion in un JavaScript numero. L'utilizzoionValue.numberValue() offre prestazioni migliori, ma è meno preciso.

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))
```

Time stamp

Gli esempi di codice seguenti mostrano come elaborare il tipo di timestamp 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**
>
> Per convertire in codice sincrono, rimuovi`async` le parole chiave`await` e cambia il`IAsyncResult` tipo in`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))
```

Stringa

Gli esempi di codice seguenti mostrano come elaborare il tipo di stringa 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**
>
> Per convertire in codice sincrono, rimuovi`async` le parole chiave`await` e cambia il`IAsyncResult` tipo in`IResult`.

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

Gli esempi di codice seguenti mostrano come elaborare il tipo Ion blob.

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**
>
> Per convertire in codice sincrono, rimuovi`async` le parole chiave`await` e cambia il`IAsyncResult` tipo in`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))
```

Elenco

Gli esempi di codice seguenti mostrano come elaborare il tipo di elenco Ion.

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**
>
> Per convertire in codice sincrono, rimuovi`async` le parole chiave`await` e cambia il`IAsyncResult` tipo in`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

In QLDB,`struct` i tipi di dati sono particolarmente unici rispetto agli altri tipi di ioni. I documenti di primo livello che inserisci in una tabella devono essere del`struct` tipo. Un campo documento può anche memorizzare un file annidato`struct`.

Per semplicità, gli esempi seguenti definiscono un documento che contiene solo`ExampleInt` i campi`ExampleString` e.

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;
});
```

In alternativa, puoi usare la [libreria Jackson per mappare] i tipi di dati da e verso Ion. La libreria supporta gli altri tipi di dati Ion, ma questo esempio si concentra sul`struct` tipo.

```
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

Usa la libreria Amazon.QLDB.Driver.Serialization per mappare i tipi di dati C# nativi da e verso Ion. La libreria supporta gli altri tipi di dati Ion, ma questo esempio si concentra sul`struct` tipo.

```
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**
>
> Per convertire in codice sincrono, rimuovi`async` le parole chiave`await` e cambia il`IAsyncResult` tipo in`IResult`.

In alternativa, puoi utilizzare Amazon. IonDotnet.Libreria Builders per elaborare i tipi di dati 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))
```

Valori nulli e tipi dinamici

QLDB supporta contenuti aperti e non applica definizioni di schemi o tipi di dati per i campi del documento. È inoltre possibile memorizzare valori nulli di Ion in un documento QLDB. Tutti gli esempi precedenti presuppongono che ogni tipo di dati restituito sia noto e non sia nullo. Gli esempi seguenti mostrano come lavorare con Ion quando il tipo di dati non è noto o è probabilmente nullo.

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

Limitazioni di marshalling

In Go,`nil` i valori non mantengono il loro tipo quando li gestisci e poi li annulli. Un`nil` valore porta a Ion null, mentre Ion null si decompone a un valore zero anziché zero`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!
```

Conversione verso il basso in JSON

Se la tua applicazione richiede la compatibilità con JSON, puoi convertire i dati Amazon Ion in JSON. Tuttavia, la conversione di Ion in JSON comporta perdite in alcuni casi in cui i dati utilizzano i ricchi tipi di ioni che non esistono in JSON.

Per informazioni dettagliate sulle regole di conversione da Ion a JSON, consulta Down-Converting to JSON in Amazon Ion Cookbook.

Utilizzo di dati e cronologia in Amazon QLDB

Gli argomenti seguenti forniscono esempi di creazione, lettura, creazione, lettura, creazione, lettura, creazione, lettura, creazione, lettura, di creazione, di creazione, lettura, di creazione, lettura, creazione, lettura, È possibile eseguire manualmente queste istruzioni utilizzando l'editor PartiQL sulla console QLDB o la shell QLDB. Questa guida illustra anche il processo di gestione dei dati da parte di QLDB quando si apportano modifiche al registro.

QLDB supporta il linguaggio di interrogazione PartiQL.

Per esempi di codice che mostrano come eseguire istruzioni simili a livello di programmazione utilizzando il driver QLDB, consulta i tutorial inNozioni base sul driver.

> **Tip**
>
> Di seguito è riportato un breve riepilogo dei suggerimenti e delle migliori pratiche per lavorare con PartiQL in QLDB:
>
> - Comprendi i limiti di concorrenza e transazione: tutti i rendiconti, comprese le`SELECT` interrogazioni, sono soggetti a conflitti e limiti di transazione (OCC) ottimistici, incluso un timeout delle transazioni di 30 secondi.
> - Usa gli indici: utilizza indici ad alta cardinalità ed esegui interrogazioni mirate per ottimizzare i rendiconti ed evitare scansioni complete delle tabelle. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.
> - Usa i predicati di uguaglianza: le ricerche indicizzate richiedono un operatore di uguaglianza (`=`o`IN`). Gli operatori di disuguaglianza (`<>`,`LIKE`,,`BETWEEN`) non sono idonei per le ricerche indicizzate e generano scansioni complete delle tabelle.
> - Usa solo join interni: QLDB supporta solo i join interni. Come procedura consigliata, partecipa ai campi indicizzati per ogni tabella a cui ti stai unendo. Scegli indici ad alta cardinalità sia per i criteri di unione che per i predicati di uguaglianza.

Argomenti

- Utilizzo della clausola BY per interrogare l'ID del documento
- Aggiornamento ed eliminazione di documenti
- Esecuzione di query sulla cronologia delle revisioni
- Redazione delle revisioni dei documenti
- Ottimizzazione delle prestazioni delle query
- Ottenere le statistiche delle dichiarazioni PartiQL
- Interrogazione del catalogo di sistema
- Gestione di tabelle
- Gestione degli indici
- ID univoci in Amazon QLDB

Creazione di tabelle con indici e inserimento di documenti

Dopo aver creato un registro Amazon QLDB, il primo passo è creare una tabella con unaCREATE TABLE dichiarazione di base. Le tabelle sono costituite daDocumentazione QLDB, che sono set di dati in`struct` formato Amazon Ion.

Argomenti

- Creazione di tabelle e indici
- Inserimento di documenti

Creazione di tabelle e indici

Le tabelle hanno nomi semplici con distinzione tra maiuscole e minuscole senza namespace. QLDB supporta contenuti aperti e non applica schemi, quindi non si definiscono attributi o tipi di dati durante la creazione di tabelle.

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle
```

Un'`CREATE TABLE`istruzione restituisce l'ID assegnato dal sistema della nuova tabella. Tutti gli ID assegnati dal sistema in QLDB sono identificatori univoci universali (UUID), ciascuno rappresentato in una stringa codificata in Base62.

> **Note**
>
> Facoltativamente, puoi definire i tag per una risorsa della tabella durante la creazione della tabella. Per scoprire come fare, consulta Tag di tabelle durante la creazione.

Puoi inoltre creare indici su tabelle per ottimizzare le prestazioni delle query.

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

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

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

> **Important**
>
> QLDB richiede un indice per cercare in modo efficiente un documento. Senza un indice, QLDB deve eseguire una scansione completa della tabella durante la lettura dei documenti. Ciò può causare problemi di prestazioni su tabelle di grandi dimensioni, inclusi conflitti di concorrenza e timeout delle transazioni.
>
> Per evitare la scansione delle tabelle, è necessario eseguire istruzioni con una clausola`WHERE` predicativa utilizzando un operatore di uguaglianza (`=`o`IN`) su un campo indicizzato o un ID di documento. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.

Nota i seguenti vincoli durante la creazione degli indici:

- Un indice può essere creato solo su un singolo campo di primo livello. Gli indici compositi, annidati, unici e basati su funzioni non sono supportati.
- È possibile creare un indice su qualsiasi tipo di dati Ion, inclusi`list` `struct`. Tuttavia, è possibile eseguire la ricerca indicizzata solo in base all'uguaglianza dell'intero valore Ion indipendentemente dal tipo di Ion. Ad esempio, quando si utilizza un`list` tipo come indice, non è possibile eseguire una ricerca indicizzata per un elemento all'interno dell'elenco.
- Le prestazioni delle query migliorano solo quando si utilizza un predicato di uguaglianza, ad esempio`WHERE indexedField = 123` o`WHERE indexedField IN (456, 789)`.

QLDB non rispetta le disuguaglianze nei predicati delle query. Di conseguenza, le scansioni filtrate per intervallo non vengono implementate.

- I nomi dei campi indicizzati fanno distinzione tra maiuscole e può contenere fino a 128 caratteri.
- La creazione di indici in QLDB è asincrona. La quantità di tempo necessaria per completare la creazione di un indice su una tabella non vuota varia a seconda delle dimensioni della tabella. Per ulteriori informazioni, consulta Gestione degli indici.

Inserimento di documenti

Quindi puoi inserire documenti nelle tue tabelle. I documenti QLDB sono archiviati in formato Amazon Ion. Le seguentiINSERT dichiarazioni PartiQL includono un sottoinsieme dei dati di esempio di immatricolazione del veicolo utilizzati inNozioni di base sulla 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'
} >>
```

Sintassi e semantica di PartiQL

- I nomi dei campi sono racchiusi tra virgolette singole (`'...'`).
- I valori delle stringhe sono inoltre racchiusi tra virgolette singole (`'...'`).
- I timestamp sono racchiusi tra i contrassegni (`` `...` ``). I contrassegni possono essere usati per indicare qualsiasi valore letterale ionico.
- I numeri interi e i decimali sono valori letterali che non devono essere indicati.

Per maggiori dettagli sulla sintassi e sulla semantica di PartiQL, vedereInterrogazione di Ion con PartiQL in Amazon QLDB.

Un'`INSERT`istruzione crea la revisione iniziale di un documento con un numero di versione pari a zero. Per identificare in modo univoco ogni documento, QLDB assegna un ID del documento come parte dei metadati. Le istruzioni Insert restituiscono l'ID di ogni documento inserito.

> **Important**
>
> Poiché QLDB non impone lo schema, è possibile inserire lo stesso documento in una tabella più volte. Ogni dichiarazione di inserimento inserisce un documento separato nel giornale e QLDB assegna a ciascun documento un ID univoco.

Per sapere come interrogare i documenti che hai inserito nella tabella, procedi aEsecuzione di query sui dati.

Esecuzione di query sui dati

La vista utente restituisce solo l'ultima revisione non eliminata dei dati utente. Questa è la visualizzazione predefinita in Amazon QLDB. Ciò significa che non sono necessari qualificatori speciali quando si desidera interrogare solo i propri dati.

Per dettagli sulla sintassi e sui parametri dei seguenti esempi di query, consultaSELECT il riferimento Amazon QLDB PartiQL.

Argomenti

- Query di base
- Proiezioni e filtri
- Join
- Dati nidificati

Query di base

`SELECT`Le query di base restituiscono i documenti inseriti nella tabella.

> **Warning**
>
> Quando si esegue una query in QLDB senza una ricerca indicizzata, viene richiamata una scansione completa della tabella. PartiQL supporta tali interrogazioni perché è compatibile con SQL. Tuttavia, non eseguite scansioni di tabelle per casi d'uso di produzione in QLDB. Le scansioni delle tabelle possono causare problemi di prestazioni su tabelle di grandi dimensioni, inclusi conflitti di concorrenza e timeout delle transazioni.

Per evitare la scansione delle tabelle, è necessario eseguire istruzioni con una clausolaWHERE predicativa utilizzando un operatore di uguaglianza su un campo indicizzato o un ID di documento; ad esempio,`WHERE indexedField = 123` o`WHERE indexedField IN (456, 789)`. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.

Le seguenti interrogazioni mostrano i risultati dei documenti di immatricolazione del veicolo precedentemente inseritiCreazione di tabelle con indici e inserimento di documenti. L'ordine dei risultati non è specifico e può variare per ogni`SELECT` interrogazione. Non dovresti fare affidamento sull'ordine dei risultati per nessuna query in 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**
>
> In PartiQL, si utilizzano le virgolette singole per indicare le stringhe nel linguaggio DML (Data Manipulation Language) o nelle istruzioni di interrogazione. Tuttavia, la console QLDB e la shell QLDB restituiscono i risultati delle query in formato di testo Amazon Ion, quindi vengono visualizzate stringhe racchiuse tra virgolette doppie.
> Questa sintassi consente al linguaggio di query PartiQL di mantenere la compatibilità SQL e al formato di testo Amazon Ion di mantenere la compatibilità JSON.

Proiezioni e filtri

Puoi fare proiezioni (mirate`SELECT`) e altri filtri standard (`WHERE`clausole). La seguente query restituisce un sottoinsieme di campi del documento dalla`VehicleRegistration` tabella. Filtra i veicoli con i seguenti criteri:

- Filtro a stringa: è registrato a Seattle.

- Filtro decimale: ha un importo della penalità pendente inferiore a`100.0`.
- Filtro data: ha una data di registrazione valida a partire dal 4 settembre 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" }]
    }
}
```

Join

Puoi anche scrivere interrogazioni interne. L'esempio seguente mostra una query di unione interna implicita che restituisce tutti i documenti di immatricolazione insieme agli attributi dei veicoli registrati.

```
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"
}
```

In alternativa, puoi scrivere la stessa query di join interna nella sintassi esplicita seguente.

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

Dati nidificati

È possibile utilizzare PartiQL in QLDB per interrogare i dati annidati nei documenti. L'esempio seguente mostra una sottoquery correlata che appiattisce i dati annidati. Il@ personaggio è tecnicamente opzionale qui. Ma indica esplicitamente che si desidera la`Owners` struttura all'interno`VehicleRegistration`, non una raccolta con un nome diverso`Owners` (se ne esiste una).

```
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: []
}
```

Quanto segue mostra una subquery nell'`SELECT`elenco che proietta dati annidati, oltre a un inner join.

```
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 seguente interrogazione restituisce il numero ordinale`PersonId` e l'indice di ogni persona nell'`Owners.SecondaryOwners`elenco di un`VehicleRegistration` documento.

```
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
}
```

Per scoprire come interrogare i metadati del documento, procedi aInterrogazione dei metadati dei documenti.

Interrogazione dei metadati dei documenti

Un'`INSERT`istruzione crea la revisione iniziale di un documento con un numero di versione pari a zero. Per identificare in modo univoco ogni documento, Amazon QLDB assegna un ID del documento come parte dei metadati.

Oltre all'ID del documento e al numero di versione, QLDB memorizza altri metadati generati dal sistema per ogni documento in una tabella. Questi metadati includono le informazioni sulle transazioni, gli attributi del giornale e il valore hash del documento.

Tutti gli ID assegnati dal sistema sono identificatori univoci universali (UUID), ciascuno rappresentato in una stringa codificata in Base62. Per ulteriori informazioni, consulta ID univoci in Amazon QLDB.

Argomenti

Visione impegnata

È possibile accedere ai metadati del documento interrogando la vista confermata. Questa visualizzazione restituisce i documenti dalla tabella definita dal sistema che corrisponde direttamente alla tabella utente. Include l'ultima revisione salvata e non eliminata sia dei dati che dei metadati generati dal sistema. Per interrogare questa vista, aggiungi il prefisso`_ql_committed_` al nome della tabella nella tua query. (Il prefisso`_ql_` è riservato in QLDB agli oggetti di sistema.)

```
SELECT * FROM _ql_committed_VehicleRegistration AS r
```

```
WHERE r.data.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Utilizzando i dati precedentemente inseritiCreazione di tabelle con indici e inserimento di documenti, l'output di questa query mostra il contenuto di sistema dell'ultima revisione di ogni documento non eliminato. Il documento di sistema contiene metadati annidati nel`metadata` campo e i dati utente annidati nel`data` campo.

```
{
    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"
    }
}
```

Campi di visualizzazione impegnati

- `blockAddress`— La posizione del blocco nel diario contabile in cui è stata effettuata la revisione del documento. Un indirizzo, che può essere utilizzato per la verifica crittografica, ha i due campi seguenti.
  - `strandId`— L'ID univoco del filone del journal che contiene il blocco.
  - `sequenceNo`— Un numero di indice che specifica la posizione del blocco all'interno del filamento.

  > **Note**
  >
  > Entrambi i documenti in questo esempio hanno lo stesso identico`blockAddress`s`equenceNo`. Poiché questi documenti sono stati inseriti all'interno di un'unica transazione (e in questo caso, in un'unica dichiarazione), sono stati salvati nello stesso blocco.

- `hash`— Il valore hash SHA-256 Ion che rappresenta in modo univoco la revisione del documento. L'hash copre la revisione`data` e`metadata` i campi e può essere utilizzato per la verifica crittografica.
- `data`— Gli attributi dei dati utente del documento.

  Se si redige una revisione, questa`data` struttura viene sostituita da un`dataHash` campo, il cui valore è l'hash Ion della`data` struttura rimossa.

- `metadata`— Gli attributi dei metadati del documento.
  - `id`— L'ID univoco del documento assegnato dal sistema.
  - `version`— Il numero di versione del documento. Si tratta di un numero intero a base zero che aumenta con ogni revisione del documento.
  - `txTime`— Il timestamp in cui la revisione del documento è stata conferita al giornale.
  - `txId`— L'ID univoco della transazione che ha eseguito la revisione del documento.

Unire le opinioni degli utenti e quelle degli utenti

È possibile scrivere interrogazioni che uniscono una tabella nella visualizzazione commit con una tabella nella vista utente. Ad esempio, potresti voler unire il documento`oid` di una tabella con un campo definito dall'utente di un'altra tabella.

La seguente interrogazione unisce due tabelle denominate`DriversLicense` e rispettivamente`id` nei rispettivi campi`PersonId` e nel`Person` relativo campo del documento, utilizzando la visualizzazione commit per quest'ultima.

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS p
ON d.PersonId = p.metadata.id
WHERE p.metadata.id = '1CWScY2qHYI9G88C2SjvtH'
```

Per scoprire come interrogare il campo ID del documento nella visualizzazione utente predefinita, procedi aUtilizzo della clausola BY per interrogare l'ID del documento.

Utilizzo della clausola BY per interrogare l'ID del documento

Sebbene sia possibile definire campi che devono essere identificatori univoci (ad esempio, il VIN di un veicolo), il vero identificatore univoco di un documento è il campo dei`id` metadati, come descritto inInserimento di documenti. Per questo motivo, è possibile utilizzare il`id` campo per creare relazioni tra tabelle.

Il`id` campo del documento è direttamente accessibile solo nella vista confermata, ma è anche possibile proiettarlo nella visualizzazione utente predefinita utilizzando la`BY` clausola. Per un esempio, vedere la seguente interrogazione e i relativi risultati.

```
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" }]
    }
}
```

In questa query,`r_id` c'è un alias definito dall'utente che viene dichiarato nella`FROM` clausola utilizzando la`BY` parola chiave. Questo`r_id` alias si lega al campo dei`id` metadati per ogni documento nel set di risultati della query. È possibile utilizzare questo alias nella`SELECT` clausola e anche nella`WHERE` clausola di una query nella vista utente.

Per accedere ad altri attributi dei metadati, tuttavia, è necessario interrogare la vista commit.

Iscrizione in base all'ID del documento

Supponiamo di utilizzare il documento`id` di una tabella come chiave esterna in un campo definito dall'utente di un'altra tabella. È possibile utilizzare la`BY` clausola per scrivere una query di unione interna per le due tabelle su questi campi (simile[Unire le opinioni degli utenti e quelle degli utenti](#) a quella dell'argomento precedente).

L'esempio seguente unisce due tabelle denominate`DriversLicense` e`Person` rispettivamente`id` nei rispettivi campi`PersonId` e nel relativo campo del documento, utilizzando la`BY` clausola per quest'ultimo.

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
ON d.PersonId = pid
WHERE pid = '1CWScY2qHYI9G88C2SjvtH'
```

Per scoprire come apportare modifiche a un documento nella tabella, procedi a[Aggiornamento ed eliminazione di documenti](#).

Aggiornamento ed eliminazione di documenti

In Amazon QLDB, una revisione del documento è una struttura Amazon Ion che rappresenta una singola versione di una sequenza di documenti identificati da un ID di documento univoco. Ogni revisione contiene il set di dati completo del documento, inclusi sia i dati utente che i metadati generati dal sistema. Ogni revisione è identificata in modo univoco da una combinazione dell'ID del documento e di un numero di versione a base zero.

Quando si aggiorna un documento, QLDB crea una nuova revisione con lo stesso ID del documento e un numero di versione incrementato. Il ciclo di vita di un documento termina quando lo si elimina da una tabella. Ciò significa che non è possibile creare nuovamente alcuna revisione del documento con lo stesso ID del documento.

Effettuare revisioni dei documenti

Ad esempio, le seguenti dichiarazioni inseriscono una nuova immatricolazione del veicolo, aggiornano la città di immatricolazione e quindi eliminano la registrazione. Ciò comporta tre revisioni di un documento.

```
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**
>
> Le istruzioni Insert e altre istruzioni DML restituiscono l'ID di ogni documento interessato. Prima di continuare, salva questo ID perché ti serve per la funzione di cronologia nell'argomento successivo. In alternativa, puoi individuare l'ID del documento con la seguente query.

```
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'
```

Per ulteriori esempi e informazioni sulla sintassi di queste istruzioni DML, consultaUPDATE eDELETE nel riferimento Amazon QLDB PartiQL.

Per inserire e rimuovere elementi specifici all'interno di un documento, è possibile utilizzareUPDATE istruzioni o altre istruzioni DML che iniziano con laFROM parola chiave. Per maggiori informazioni ed esempi, consulta ilDA (INSERISCI, RIMUOVI o IMPOSTA) riferimento.

Dopo aver eliminato un documento, non è più possibile interrogarlo nelle visualizzazioni associate o utente. Per sapere come interrogare la cronologia delle revisioni di questo documento utilizzando la funzione di cronologia integrata, procedi aEsecuzione di query sulla cronologia delle revisioni.

Esecuzione di query sulla cronologia delle revisioni

Amazon QLDB archivia la cronologia completa di ogni documento in una tabella. Puoi vedere tutte e tre le revisioni del documento di immatricolazione del veicolo che hai precedentemente inserito, aggiornato ed eliminatoAggiornamento ed eliminazione di documenti interrogando la funzione di cronologia integrata.

Argomenti

- Funzione di cronologia
- Esempio di query sulla cronologia

Funzione di cronologia

La funzione di cronologia in QLDB è un'estensione PartiQL che restituisce le revisioni dalla vista definita dal sistema della tabella. Quindi, include sia i tuoi dati che i metadati associati nello stesso schema della vista confermata.

Sintassi

```
SELECT * FROM history( table_name | 'table_id' [, `start-time` [, `end-time` ] ] ) AS h
[ WHERE h.metadata.id = 'id' ]
```

Argomenti

nome_tabella | '***tabella_id***'

Il nome o l'ID della tabella. Un nome di tabella è un identificatore PartiQL che è possibile indicare con virgolette doppie o senza virgolette. L'ID di tabella è una stringa letterale che deve essere racchiusa tra virgolette singole. Per ulteriori informazioni sull'utilizzo degli ID di tabella, consultaInterrogazione della cronologia delle tabelle inattive.

`ora di ***inizio***`, `ora di ***fine***`

(Facoltativo) Specifica l'intervallo di tempo durante il quale le revisioni erano attive. Questi parametri non specificano l'intervallo di tempo durante il quale le revisioni sono state salvate nel giornale in una transazione.

L'ora di inizio e di fine sono valori letterali del timestamp Ion che possono essere indicati con contrassegni (`` `...` ``). Per ulteriori informazioni, consulta Interrogazione di Ion con PartiQL in Amazon QLDB.

Questi parametri temporali hanno il seguente comportamento:

- L'ora di inizio e l'ora di fine sono entrambe incluse. deve essere in formato data e ora e in formato UTC.
- L'ora di inizio deve essere inferiore o uguale all'ora di fine e può essere qualsiasi data passata arbitraria.
- L'ora di fine deve essere minore o uguale alla data e l'ora UTC.
- Se si specifica un'ora di inizio ma non un'ora di fine, l'interrogazione imposta come ora di fine la data e l'ora correnti. Se non si specifica nessuno dei due, l'interrogazione restituisce l'intera cronologia.

'*id*'

(Facoltativo) L'ID del documento per il quale si desidera interrogare la cronologia delle revisioni, indicato da virgolette singole.

> **Tip**
>
> Come procedura consigliata, qualifica un'interrogazione cronologica con un intervallo di date (ora di inizio e ora di fine) e un ID del documento (`metadata.id`). In QLDB, ogni`SELECT` query viene elaborata in una transazione ed è soggetta a un limite di timeout della transazione.
> Le interrogazioni relative alla cronologia non utilizzano gli indici creati su una tabella. La cronologia QLDB è indicizzata solo in base all'ID del documento e al momento non è possibile creare indici di cronologia aggiuntivi. Le interrogazioni cronologiche che includono un'ora di inizio e un'ora di fine ottengono il vantaggio della qualificazione dell'intervallo di date.

Esempio di query sulla cronologia

Per consultare la cronologia del documento di immatricolazione del veicolo, utilizza`id` quello precedentemente salvatoAggiornamento ed eliminazione di documenti. Ad esempio, la seguente query di cronologia restituisce tutte le revisioni dell'ID del documento`ADR2Ll1fGsU4Jr4EqTdnQF` che sono state attive tra`2019-06-05T00:00:00Z` e`2019-06-05T23:59:59Z`.

> **Note**
>
> Ricorda che i parametri dell'ora di inizio e di fine non specificano l'intervallo di tempo in cui le revisioni sono state salvate nel giornale in una transazione. Ad esempio, se una revisione è stata salvata prima`2019-06-05T00:00:00Z` ed è rimasta attiva dopo l'ora di inizio, questa query di esempio restituirà quella revisione nei risultati.

Assicurati di sostituire l'`id`ora di inizio e l'ora di fine con i tuoi valori, a seconda dei casi.

```
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
```

I risultati della query dovrebbero apparire simili ai seguenti.

```
{
    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"
    }
}
```

L'output include attributi di metadati che forniscono dettagli su quando ogni elemento è stato modificato e con quale transazione. Da questi dati, puoi visualizzare quanto segue:

- Il documento è identificato in modo univoco dal sistema assegnato`id`:`ADR2Ll1fGsU4Jr4EqTdnQF`. Si tratta di un UUID rappresentato in una stringa codificata in Base62.
- Un'`INSERT`istruzione crea la revisione iniziale di un documento (versione`0`).
- Ogni aggiornamento successivo crea una nuova revisione con lo stesso documento`id` e un numero di versione incrementato.
- Il`txId` campo indica la transazione in cui è stata salvata ogni revisione `etxTime` indica quando ciascuna di esse è stata confermata.
- Un'`DELETE`istruzione crea una nuova ma definitiva revisione di un documento. Questa revisione finale contiene solo metadati.

Per sapere come eliminare definitivamente una revisione, procedi aRedazione delle revisioni dei documenti.

Redazione delle revisioni dei documenti

In Amazon QLDB, un'`DELETE`istruzione elimina un documento solo in modo logico creando una nuova revisione che lo contrassegna come eliminato. QLDB supporta anche un'operazione di redazione dei dati che consente di eliminare definitivamente le revisioni dei documenti inattivi nella cronologia di una tabella.

> **Note**
>
> I registri creati prima del 22 luglio 2021 non sono attualmente idonei alla redazione. Puoi visualizzare l'ora di creazione del tuo libro contabile sulla console Amazon QLDB.

L'operazione di redazione elimina solo i dati utente nella revisione specificata e lascia invariati la sequenza del diario e i metadati del documento. Ciò mantiene l'integrità complessiva dei dati del registro.

Prima di iniziare con la redazione dei dati in QLDB, assicurati di esaminarliConsiderazioni e limitazioni nel riferimento Amazon QLDB PartiQL.

Argomenti

- Stored procedure
- Verificare se una redazione è completa
- Esempio di redazione
- Eliminazione e redazione di una revisione attiva
- Redazione di un campo particolare all'interno di una revisione

Stored procedure

È possibile utilizzare la proceduraREVISIONE_REDAZIONE memorizzata per eliminare definitivamente una singola revisione inattiva in un libro mastro. Questa procedura memorizzata elimina tutti i dati utente nella revisione specificata sia nell'archiviazione indicizzata che nell'archiviazione del journal. Tuttavia, lascia invariati la sequenza del diario e i metadati del documento, inclusi l'ID e l'hash del documento. Questa operazione è irreversibile.

La revisione del documento specificata deve essere una revisione inattiva nella cronologia. L'ultima revisione attiva di un documento non è idonea alla redazione.

Per oscurare più revisioni, è necessario eseguire la stored procedure una volta per ogni revisione. Puoi oscurare una revisione per transazione.

Sintassi

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

Argomenti

`indirizzo di blocco`

La posizione del blocco del diario in cui si trova la revisione del documento da oscurare. Un indirizzo è una struttura Amazon Ion con due campi:`strandId` e`sequenceNo`.

Questo è un valore letterale di Ion che viene indicato con contrassegni. Ad esempio:

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

'table-id'

L'ID univoco della tabella di cui si desidera oscurare la revisione del documento, indicato da virgolette singole.

'id del documento'

L'ID univoco del documento della revisione da oscurare, indicato da virgolette singole.

Verificare se una redazione è completa

Quando si invia una richiesta di redazione eseguendo la procedura memorizzata, QLDB elabora la redazione dei dati in modo asincrono. Al termine, i dati utente nella revisione (rappresentati dalla`data` struttura) vengono rimossi definitivamente. Per verificare se una richiesta di redazione è stata completata, puoi utilizzare uno dei seguenti:

- Esportazione di riviste
- Stream del diario
- GetBlock API operation (Operazione API)

- GetRevision API operation (Operazione API)
- Funzione di cronologia— Nota: dopo aver completato una redazione nel diario, può essere necessario del tempo prima che le interrogazioni sulla cronologia mostrino il risultato della redazione. Potresti vedere alcune revisioni cancellate prima di altre quando la redazione asincrona viene completata, ma le interrogazioni della cronologia mostreranno i risultati completati alla fine.

Una volta completata la redazione della revisione, la`data` struttura della revisione viene sostituita da un nuovo`dataHash` campo. Il valore di questo campo è l'hash Ion della`data` struttura rimossa, come mostrato nell'esempio seguente. Di conseguenza, il libro mastro mantiene l'integrità complessiva dei dati e rimane verificabile crittograficamente tramite le operazioni API di verifica esistenti. Per ulteriori informazioni sulla verifica, consultaVerifica dei dati in Amazon QLDB.

Esempio di redazione

Considera il documento di immatricolazione del veicolo che hai esaminato in precedenzaEsecuzione di query sulla cronologia delle revisioni. Supponiamo che si desideri redigere la seconda revisione (`version:1`). L'esempio di interrogazione seguente mostra questa revisione prima della redazione. Nei risultati della query, la`data` struttura che verrà oscurata è evidenziata in *`corsivo rosso`*.

```
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"
    }
}
```

`blockAddress`Annotalo nei risultati della query perché devi passare questo valore alla`REDACT_REVISION` stored procedure. Quindi, trova l'ID univoco della`VehicleRegistration` tabella interrogando il catalogo di sistema, come segue.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Utilizza questo ID di tabella insieme all'ID del documento e all'indirizzo del blocco per eseguire`REDACT_REVISION`. L'ID della tabella e l'ID del documento sono stringhe letterali che devono essere racchiuse tra virgolette singole e l'indirizzo del blocco è un valore letterale Ion racchiuso tra virgolette. Assicurati di sostituire questi argomenti con i tuoi valori, a seconda dei casi.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

> **Tip**
>
> Quando si utilizza la console QLDB o la shell QLDB per richiedere un ID di tabella o un ID di documento (o qualsiasi valore letterale di una stringa), il valore restituito è racchiuso tra virgolette doppie. Tuttavia, quando si specificano gli argomenti ID della tabella e dell'ID del documento della`REDACT_REVISION` stored procedure, è necessario racchiudere i valori tra virgolette singole.
>
> Questo perché si scrivono istruzioni in formato PartiQL, ma QLDB restituisce i risultati in formato Amazon Ion. Per dettagli sulla sintassi e sulla semantica di PartiQL in QLDB, vedereInterrogare Ion con PartiQL.

Una richiesta di redazione valida restituisce una struttura Ion che rappresenta la revisione del documento che si sta oscurando, come segue.

```
{
  blockAddress: {
    strandId: "JdxjkR9bSYB5jMHWcI464T",
    sequenceNo: 17
  },
  tableId: "5PLf9SXwndd63lPaSIa0O6",
  documentId: "ADR2Ll1fGsU4Jr4EqTdnQF",
  version: 1
}
```

Quando si esegue questa procedura memorizzata, QLDB elabora la richiesta di redazione in modo asincrono. Al termine della redazione, la`data` struttura viene rimossa definitivamente e sostituita da un nuovo *`dataHash`*campo. Il valore di questo campo è l'hash Ion della`data` struttura rimossa, come segue.

> **Note**
>
> Questo`dataHash` esempio viene fornito solo a scopo informativo e non è un valore hash calcolato reale.

```
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:17
    },
    hash:{{LGSFZ4iEYWZeMwmAqcxxNyT4wbCtuMOmFCj8pEd6Mp0=}},
    dataHash: {{s83jd7sfhsdfhksj7hskjdfjfpIPP/DP2hvionas2d4=}},
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:1,
        txTime:2019-06-05T21:01:442d-3Z,
        txId:"9cArhIQV5xf5Tf5vtsPwPq"
    }
}
```

Eliminazione e redazione di una revisione attiva

Le revisioni attive dei documenti (ovvero le ultime revisioni non eliminate di ogni documento) non sono idonee alla redazione dei dati. Prima di poter oscurare una revisione attiva, è necessario

aggiornarla o eliminarla. In questo modo la revisione precedentemente attiva viene spostata nella cronologia e la rende idonea alla redazione.

Se il tuo caso d'uso richiede che l'intero documento sia contrassegnato come eliminato, devi prima utilizzare un'istruzione DELETE. Ad esempio, la seguente dichiarazione elimina logicamente il`VehicleRegistration` documento con un VIN di`1HVBBAANXWH544237`.

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

Quindi, redigi la revisione precedente prima di questa eliminazione, come descritto in precedenza. Se necessario, puoi anche redigere individualmente eventuali revisioni precedenti.

Se il tuo caso d'uso richiede che il documento rimanga attivo, devi prima utilizzare un'istruzione UPDATE o FROM per oscurare o rimuovere i campi che desideri oscurare. Questo processo è descritto nella sezione seguente.

Redazione di un campo particolare all'interno di una revisione

QLDB non supporta la redazione di un campo particolare all'interno di una revisione del documento. A tale scopo, è possibile innanzitutto utilizzare un'istruzione UPDATE-REMOVE o FROM-REMOVE per rimuovere un campo esistente da una revisione. Ad esempio, la seguente dichiarazione rimuove il`LicensePlateNumber` campo dal`VehicleRegistration` documento con un VIN di`1HVBBAANXWH544237`.

```
UPDATE VehicleRegistration AS r
REMOVE r.LicensePlateNumber
WHERE r.VIN = '1HVBBAANXWH544237'
```

Quindi, redigi la revisione precedente prima di questa rimozione, come descritto in precedenza. Se necessario, puoi anche redigere individualmente eventuali revisioni precedenti che includono questo campo ora rimosso.

Per scoprire come ottimizzare le tue interrogazioni, procedi aOttimizzazione delle prestazioni delle query.

Ottimizzazione delle prestazioni delle query

Amazon QLDB ha lo scopo di soddisfare le esigenze di carichi di lavoro OLTP (OLTP) di elaborazione di transazioni online (OLTP). Ciò significa che QLDB è ottimizzato per un set specifico

di modelli di query, anche se supporta funzionalità di query di tipo SQL. È fondamentale progettare le applicazioni e i relativi modelli di dati in modo che funzionino con questi modelli di interrogazione. Altrimenti, man mano che le tabelle crescono, si verificheranno problemi di prestazioni significativi, tra cui latenza delle query, timeout delle transazioni e conflitti di concorrenza.

Questa sezione descrive i vincoli delle query in QLDB e fornisce una guida per scrivere interrogazioni ottimali in base a questi vincoli.

Argomenti

Limite di timeout della transazione

In QLDB, ogni istruzione PartiQL (inclusa ogni`SELECT` query) viene elaborata in una transazione ed è soggetta a un limite di timeout della transazione. Una transazione può essere eseguita fino a 30 secondi prima di essere confermata. Dopo questo limite, QLDB rifiuta qualsiasi lavoro svolto sulla transazione e scarta la sessione che ha eseguito la transazione. Questo limite protegge il cliente del servizio dalla perdita di sessioni avviando transazioni e non eseguendole o annullandole.

Conflitti di concorrenza

QLDB implementa il controllo della concorrenza utilizzando il controllo ottimistico della concorrenza (OCC). Le interrogazioni non ottimali possono anche portare a ulteriori conflitti OCC. Per informazioni su OCC, vedereModello di concorrenza Amazon QLDB.

Schemi di query ottimali

Come procedura consigliata, è consigliabile eseguire istruzioni con una clausola`WHERE` predicativa che filtra in base a un campo indicizzato o all'ID di un documento. QLDB richiede un operatore di uguaglianza (`=`o`IN`) su un campo indicizzato per cercare in modo efficiente un documento.

Di seguito sono riportati alcuni esempi di modelli di interrogazione ottimali nella vista utente.

```
--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'
```

Qualsiasi query che non segue questi schemi richiama una scansione completa della tabella. Le scansioni delle tabelle possono causare il timeout delle transazioni per le query su tabelle di grandi dimensioni o per le query che restituiscono set di risultati di grandi dimensioni. Possono anche portare a conflitti OCC con transazioni concorrenti.

Indici ad alta cardinalità

Consigliamo di indicizzare i campi che contengono valori di cardinalità elevati. Ad esempio, i`LicensePlateNumber` campi`VIN` e nella`VehicleRegistration` tabella sono campi indicizzati che devono essere univoci.

Evita di indicizzare campi a bassa cardinalità come codici di stato, stati o province degli indirizzi e codici postali. Se indicizzi un campo di questo tipo, le tue query possono produrre set di risultati di grandi dimensioni che hanno maggiori probabilità di causare timeout delle transazioni o conflitti OCC involontari.

Query di visualizzazione confermate

Le query eseguite nella visualizzazione commit seguono le stesse linee guida di ottimizzazione delle query di visualizzazione utente. Gli indici creati su una tabella vengono utilizzati anche per le interrogazioni nella vista confermata.

Interrogazioni sulle funzioni di cronologia

Le interrogazioni sulle funzioni di cronologia non utilizzano gli indici creati su una tabella. La cronologia QLDB è indicizzata solo in base all'ID del documento e al momento non è possibile creare indici di cronologia aggiuntivi.

Come procedura consigliata, qualifica un'interrogazione cronologica con un intervallo di date (ora di inizio e ora di fine) e un ID del documento (`metadata.id`). Le interrogazioni cronologiche che includono un'ora di inizio e un'ora di fine ottengono il vantaggio della qualificazione dell'intervallo di date.

Interrogazioni interne

Per le query di join interne, utilizza criteri di join che includano almeno un campo indicizzato per la tabella sul lato destro del join. Senza un indice di join, una query di join richiama più scansioni di tabelle: per ogni documento nella tabella sinistra del join, la query esegue la scansione completa della tabella destra. La procedura migliore consiste nell'unire i campi indicizzati per ogni tabella a cui si sta unendo, oltre a specificare un predicato di`WHERE` uguaglianza per almeno una tabella.

Ad esempio, la seguente query unisce le`Vehicle` tabelle`VehicleRegistration` e`VIN` nei rispettivi campi, che sono entrambi indicizzati. Questa interrogazione ha anche un predicato di uguaglianza su`VehicleRegistration.VIN`.

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

Scegli indici ad alta cardinalità sia per i criteri di unione che per i predicati di uguaglianza nelle tue query di partecipazione.

Schemi di query da evitare

Di seguito sono riportati alcuni esempi di istruzioni non ottimali che non si adattano bene alle tabelle più grandi in QLDB. Ti consigliamo vivamente di non fare affidamento su questi tipi di query per le tabelle che crescono nel tempo, perché alla fine le query comporteranno il timeout delle transazioni. Poiché le tabelle contengono documenti di dimensioni variabili, è difficile definire limiti precisi per le query non indicizzate.

```
--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)
```

In generale, non consigliamo di eseguire i seguenti tipi di pattern di query per i casi d'uso di produzione in QLDB:

- Interrogazioni di elaborazione analitica online (OLAP)
- Interrogazioni esplorative senza una clausola predicativa
- Richieste di segnalazione
- Ricerca testuale

Consigliamo invece lo streaming dei dati su un servizio di database appositamente creato e ottimizzato per casi d'uso analitici. Ad esempio, puoi trasmettere dati QLDB ad Amazon OpenSearch Service per fornire funzionalità di ricerca di testo completo sui documenti. Per un'applicazione di esempio che dimostri questo caso d'uso, consulta il GitHub repository aws-samples/amazon-qldb-streaming-amazon -opensearch-service-sample-python. Per informazioni sugli stream QLDB, consultaStreaming dei dati del diario da Amazon QLDB.

Monitoraggio delle prestazioni

Il driver QLDB fornisce informazioni sull'utilizzo degli I/O e sulla tempistica nell'oggetto risultato di un'istruzione. Puoi utilizzare questi parametri per identificare le istruzioni PartiQL inefficienti. Per saperne di più, procedi aOttenere le statistiche delle dichiarazioni PartiQL.

Puoi anche utilizzare Amazon CloudWatch per monitorare le prestazioni del tuo libro contabile per le operazioni sui dati. Monitora la`CommandLatency` metrica per un determinato`LedgerName`

eCommandType. Per ulteriori informazioni, consulta Monitoraggio con Amazon CloudWatch. Per scoprire come QLDB utilizza i comandi per gestire le operazioni sui dati, vedereGestione delle sessioni con il conducente.

Ottenere le statistiche delle dichiarazioni PartiQL

Amazon QLDB fornisce statistiche sull'esecuzione delle dichiarazioni che possono aiutarti a ottimizzare l'utilizzo di QLDB eseguendo istruzioni PartiQL più efficienti. QLDB restituisce queste statistiche insieme ai risultati della dichiarazione. Includono metriche che quantificano l'utilizzo di I/O consumato e il tempo di elaborazione lato server, che è possibile utilizzare per identificare le dichiarazioni inefficienti.

Questa funzionalità è attualmente disponibile nell'editor PartiQL sulla console QLDB, nella shell QLDB e nell'ultima versione del driver QLDB per tutte le lingue supportate. È inoltre possibile visualizzare le statistiche delle dichiarazioni relative alla cronologia delle interrogazioni sulla console.

Argomenti

Utilizzo I/O

La metrica di utilizzo dell'I/O descrive il numero di richieste di I/O di lettura. Se il numero di richieste di I/O di lettura è superiore al previsto, indica che l'istruzione non è ottimizzata, ad esempio la mancanza di un indice. Ti consigliamo di esaminareSchemi di query ottimali l'argomento precedente, Ottimizzazione delle prestazioni delle query.

> **Note**
>
> Quando si esegue un'`CREATE INDEX`istruzione su una tabella non vuota, la metrica di utilizzo dell'I/O include le richieste di lettura solo per la chiamata di creazione sincrona dell'indice.
> QLDB crea l'indice per qualsiasi documento esistente nella tabella in modo asincrono. Queste richieste di lettura asincrona non sono incluse nella metrica di utilizzo dell'I/O contenuta nei risultati del rendiconto. Le richieste di lettura asincrona vengono addebitate separatamente e vengono aggiunte al totale degli I/O di lettura dopo il completamento della creazione dell'indice.

Utilizzo della console QLDB

Per ottenere l'utilizzo dell'I/O di lettura di una dichiarazione utilizzando la console QLDB, procedi come segue:

1. Apri la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel riquadro di navigazione, seleziona l'editor PartiQL.
3. Scegli un libro contabile dall'elenco a discesa dei libri contabili.
4. Nella finestra dell'editor di query, inserisci qualsiasi istruzione di tua scelta, quindi scegli Esegui. Di seguito è riportato un esempio di interrogazione.

   ```
   SELECT * FROM testTable WHERE firstName = 'Jim'
   ```

   Per eseguire un'istruzione, puoi anche usare la scorciatoia da tastieraCtrl +Enter per Windows oCmd +Return per macOS. Per altre scelte rapide da tastiera, consultaScelte rapide da tastiera dell'editor PartiQL.
5. Sotto la finestra dell'editor delle query, i risultati della query includono gli I/O di lettura, ovvero il numero di richieste di lettura effettuate dall'istruzione.

Puoi anche visualizzare gli I/O letti della cronologia delle query eseguendo i seguenti passaggi:

1. Nel pannello di navigazione, scegli Query recenti nell'editor PartiQL.
2. La colonna Read I/O mostra il numero di richieste di lettura effettuate da ciascuna istruzione.

Utilizzo del driver QLDB

Per ottenere l'utilizzo dell'I/O di un'istruzione utilizzando il driver QLDB, chiamate l'`getConsumedIOs`operazione del cursore di flusso o del cursore bufferizzato del risultato.

Gli esempi di codice seguenti mostrano come ottenere gli I/O di lettura dal cursore di lettura dal cursore dello stream del risultato di un'istruzione.

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**
>
> Per convertire in codice sincrono, rimuovi le `async` parole chiave `await` e e modificane il `IAsyncResult` tipo in `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))
```

Gli esempi di codice seguenti mostrano come ottenere gli I/O di lettura dal cursore in formato UTC. Questo restituisce il totale degli I/O di letturaExecuteStatement eFetchPage delle richieste.

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**
>
> Per convertire in codice sincrono, rimuovi le`async` parole chiave`await` e e modificane il`IAsyncResult` tipo in`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**
>
> Il cursore dello stream è statico perché impagina il set di risultati. Pertanto, le operazioni `getTimingInformation` and `getConsumedIOs` restituiscono le metriche accumulate dal momento in cui le chiami.
> Il cursore bufferizzato memorizza il set di risultati in memoria e restituisce le metriche totali accumulate.

Informazioni sulla tempistica

La metrica delle informazioni sui tempi descrive il tempo di elaborazione lato server in millisecondi. Il tempo di elaborazione sul lato server è definito come la quantità di tempo che QLDB impiega per elaborare una dichiarazione. Questo non include il tempo dedicato alle chiamate di rete o alle pause. Questa metrica distingue il tempo di elaborazione sul lato del servizio QLDB dal tempo di elaborazione sul lato client.

Utilizzo della console QLDB

Per ottenere le informazioni sulla tempistica di un rendiconto utilizzando la console QLDB, procedi come segue:

1. Apri la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel riquadro di navigazione, seleziona l'editor PartiQL.
3. Scegli un libro contabile dall'elenco a discesa dei libri contabili.
4. Nella finestra dell'editor di query, inserisci qualsiasi istruzione di tua scelta, quindi scegli Esegui. Di seguito è riportato un esempio di interrogazione.

   ```
   SELECT * FROM testTable WHERE firstName = 'Jim'
   ```

   Per eseguire un'istruzione, puoi anche usare la scorciatoia da tastieraCtrl +Enter per Windows oCmd +Return per macOS. Per altre scelte rapide da tastiera, consultaScelte rapide da tastiera dell'editor PartiQL.
5. Sotto la finestra dell'editor delle query, i risultati della query includono la latenza sul lato server, che è la quantità di tempo tra il momento in cui QLDB ha ricevuto la richiesta di dichiarazione e il momento in cui ha inviato la risposta. Si tratta di un sottoinsieme della durata totale della query.

Puoi anche visualizzare le informazioni sulla tempistica della cronologia delle interrogazioni eseguendo i seguenti passaggi:

1. Nel pannello di navigazione, scegli Query recenti nell'editor PartiQL.
2. La colonna Tempo di esecuzione (ms) visualizza queste informazioni sulla tempistica per ogni istruzione.

Utilizzo del driver QLDB

Per ottenere le informazioni sulla temporizzazione di un'istruzione utilizzando il driver QLDB, chiamate l'`getTimingInformation`operazione del cursore di flusso o del cursore bufferizzato del risultato.

Gli esempi di codice seguenti mostrano come ottenere il tempo di elaborazione dal cursore di elaborazione dal cursore di flusso del risultato di un'istruzione.

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**
>
> Per convertire in codice sincrono, rimuovi le`async` parole chiave`await` e e modificane il`IAsyncResult` tipo in`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))
```

Gli esempi di codice seguenti mostrano come ottenere il tempo di elaborazione dal cursore in formato UTC. Questo restituisce il tempo totale di elaborazione delle`FetchPage` richieste`ExecuteStatement` e delle richieste.

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**
>
> Per convertire in codice sincrono, rimuovi le`async` parole chiave`await` e e modificane il`IAsyncResult` tipo in`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**
>
> Il cursore dello stream è statico perché impagina il set di risultati. Pertanto, le`getTimingInformation` operazioni`getConsumedIOs` and restituiscono le metriche accumulate dal momento in cui le chiami.
> Il cursore bufferizzato memorizza il set di risultati in memoria e restituisce le metriche totali accumulate.

Per informazioni su come eseguire query nel catalogo di sistema, procedere a[Interrogazione del catalogo di sistema](#).

Interrogazione del catalogo di sistema

Ogni tabella creata in un registro Amazon QLDB ha un ID univoco assegnato dal sistema. È possibile trovare l'ID di una tabella, il relativo elenco di indici e altri metadati interrogando la tabella del catalogo di sistema`information_schema.user_tables`.

Tutti gli ID assegnati dal sistema sono identificatori univoci universali (UUID), ciascuno rappresentato in una stringa codificata in Base62. Per ulteriori informazioni, consulta ID univoci in Amazon QLDB.

L'esempio seguente mostra i risultati di una query che restituisce gli attributi dei metadati della`VehicleRegistration` tabella.

```
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"
}
```

Campi di metadati della tabella

- `tableId`— L'ID univoco della tabella.
- `name`— Il nome della tabella.
- `indexes`— L'elenco degli indici nella tabella.
  - `indexId`— L'ID univoco dell'indice.
  - `expr`— Il percorso indicizzato del documento. Questo campo è una stringa nel formato:`[fieldName]`.
  - `status`— Lo stato attuale dell'indice (`BUILDING`FINALIZING,`ONLINE`,`FAILED`, o`DELETING`). QLDB non utilizza l'indice nelle query finché lo stato non è`ONLINE`.
  - `message`— Il messaggio di errore che descrive il motivo per cui l'indice ha uno`FAILED` stato. Questo campo è incluso se non è valido.

- `status`— Lo stato corrente della tabella (`ACTIVE`o`INACTIVE`). Un tavolo diventa`INACTIVE` quando`DROP` lo siedi tu.

Per imparare a gestire le tabelle utilizzando le`UNDROP TABLE` istruzioni`DROP TABLE` and, procedere a[Gestione di tabelle](#).

Gestione di tabelle

Questa sezione descrive come gestire le tabelle utilizzando le`UNDROP TABLE` istruzioni`DROP TABLE` and in Amazon QLDB. Descrive anche come etichettare le tabelle durante la loro creazione. Le quote per il numero di tabelle attive e le tabelle totali che è possibile creare sono definite in[Quote e limiti in Amazon QLDB](#).

Argomenti

- [Tag di tabelle durante la creazione](#)
- [Abbassare tabelle](#)
- [Interrogazione della cronologia delle tabelle inattive](#)
- [Riattivazione delle tabelle](#)

Tag di tabelle durante la creazione

> **Note**
>
> L'etichettatura delle tabelle al momento della creazione è attualmente supportata per i libri contabili solo in modalità`STANDARD` autorizzazioni.

Puoi contrassegnare le risorse della tabella. Per gestire i tag per le tabelle esistenti, utilizza le operazioniAWS Management Console o le`TagResource` API e`ListTagsForResource`.`UntagResource` Per ulteriori informazioni, consulta [Assegnazione di tag alle risorse Amazon QLDB](#).

È inoltre possibile definire i tag della tabella durante la creazione della tabella utilizzando la console QLDB o specificandoli in un'istruzione`CREATE TABLE` PartiQL. L'esempio seguente crea una tabella denominata`Vehicle` con il tag`environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

L'aggiunta di tag alle risorse in fase di creazione consente di evitare di eseguire script di tagging personalizzati dopo la creazione delle risorse. Dopo aver eseguito un tag, puoi controllare l'accesso alla tabella in base a tali tag. Ad esempio, puoi concedere l'accesso completo solo alle tabelle con un tag specifico. Per un esempio di policy JSON, consultaAccesso completo a tutte le azioni basate sui tag della tabella.

Abbassare tabelle

Per eliminare una tabella, usa un'DROP TABLEistruzione di base. Quando trascini una tabella in QLDB, la stai semplicemente disattivando.

Ad esempio, la seguente istruzione disattiva la`VehicleRegistration` tabella.

```
DROP TABLE VehicleRegistration
```

Un'`DROP TABLE`istruzione restituisce l'ID della tabella assegnato dal sistema. Lo stato di`VehicleRegistration` dovrebbe ora essere`INACTIVE` nella tabella del catalogo di sistema information_schema.user_tables.

```
SELECT status FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Interrogazione della cronologia delle tabelle inattive

Oltre al nome di una tabella, puoi anche interrogare il QLDBFunzione di cronologia con un ID di tabella come primo argomento di input. È necessario utilizzare l'ID della tabella per interrogare la cronologia di una tabella inattiva. Dopo la disattivazione di una tabella, non è più possibile interrogarne la cronologia con il nome della tabella.

Innanzitutto, trova l'ID della tabella interrogando la tabella del catalogo di sistema. Ad esempio, la seguente query restituisce`tableId` la`VehicleRegistration` tabella.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Quindi, puoi utilizzare questo ID per eseguire la stessa query di cronologia daEsecuzione di query sulla cronologia delle revisioni. Di seguito è riportato un esempio che interroga la cronologia dell'ID

del documentoADR2Ll1fGsU4Jr4EqTdnQF dall'ID della tabella5PLf9SXwndd63lPaSIa0O6. L'ID della tabella è una stringa letterale che deve essere racchiusa tra virgolette singole.

```
--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'
```

Riattivazione delle tabelle

Dopo aver disattivato una tabella in QLDB, è possibile utilizzare l'TAVOLO UNDROPistruzione per riattivarla.

Innanzitutto, trova l'ID della tabella da`information_schema.user_tables`. Ad esempio, la seguente query restituisce`tableId` la`VehicleRegistration` tabella. Lo stato dovrebbe essere`INACTIVE`.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Quindi, usa questo ID per riattivare la tabella. Di seguito è riportato un esempio che elimina l'ID della tabella5PLf9SXwndd63lPaSIa0O6. In questo caso, l'ID della tabella è un identificatore univoco racchiuso tra virgolette doppie.

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

Lo stato di`VehicleRegistration` dovrebbe ora essere`ACTIVE`.

Per scoprire come creare, descrivere ed eliminare gli indici, procedi aGestione degli indici.

Gestione degli indici

In questa sezione viene descritto come creare, descrivere ed eliminare indici in Amazon QLDB. La quota per il numero di indici per tabella che puoi creare è definita inQuote e limiti in Amazon QLDB.

Argomenti

- Creazione di indici
- Descrizione degli indici
- Abbassamento degli indici
- Errori comuni

Creazione di indici

Come descritto anche inCreazione di tabelle e indici, è possibile utilizzare l'istruzione CREATE INDEX per creare un indice su una tabella per un campo di primo livello specificato, come segue. Il nome della tabella e il nome del campo indicizzato fanno distinzione tra maiuscole e minuscole.

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

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

Ogni indice creato su una tabella ha un ID univoco assegnato dal sistema. Per individuare l'ID dell'indice, consulta la sezione seguenteDescrizione degli indici.

> **Important**
>
> QLDB richiede un indice per cercare in modo efficiente un documento. Senza un indice, QLDB deve eseguire una scansione completa della tabella durante la lettura dei documenti. Ciò può causare problemi di prestazioni su tabelle di grandi dimensioni, inclusi conflitti di concorrenza e timeout delle transazioni.
>
> Per evitare la scansione delle tabelle, è necessario eseguire istruzioni con una clausola`WHERE` predicativa utilizzando un operatore di uguaglianza (`=`o`IN`) su un campo indicizzato o un ID di documento. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.

Nota i seguenti vincoli durante la creazione degli indici:

- Un indice può essere creato solo su un singolo campo di primo livello. Gli indici compositi, annidati, unici e basati su funzioni non sono supportati.
- È possibile creare un indice su qualsiasi tipo di dati Ion, inclusi`list` `struct`. Tuttavia, è possibile eseguire la ricerca indicizzata solo in base all'uguaglianza dell'intero valore Ion indipendentemente dal tipo di Ion. Ad esempio, quando si utilizza un`list` tipo come indice, non è possibile eseguire una ricerca indicizzata per un elemento all'interno dell'elenco.
- Le prestazioni delle query migliorano solo quando si utilizza un predicato di uguaglianza, ad esempio`WHERE indexedField = 123` o`WHERE indexedField IN (456, 789)`.

  QLDB non rispetta le disuguaglianze nei predicati delle query. Di conseguenza, le scansioni filtrate per intervallo non vengono implementate.

- I nomi dei campi indicizzati fanno distinzione tra maiuscole e può contenere fino a 128 caratteri.
- La creazione di indici in QLDB è asincrona. La quantità di tempo necessaria per completare la creazione di un indice su una tabella non vuota varia a seconda delle dimensioni della tabella. Per ulteriori informazioni, consulta Gestione degli indici.

Descrizione degli indici

La creazione di indici in QLDB è asincrona. La quantità di tempo necessaria per completare la creazione di un indice su una tabella non vuota varia a seconda delle dimensioni della tabella. Per verificare lo stato di una generazione di indice, è possibile interrogare la tabella del catalogo di sistema information_schema.user_tables.

Ad esempio, l'istruzione seguente interroga il catalogo di sistema per tutti gli indici della`VehicleRegistration` tabella.

```
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"
}
```

Campi dell'indice

- `indexId`— L'ID univoco dell'indice.
- `expr`— Il percorso indicizzato del documento. Questo campo è una stringa nel formato:`[fieldName]`.
- `status`— Lo stato attuale dell'indice. Lo stato di un indice può avere uno dei seguenti valori:

- `BUILDING`— Sta creando attivamente l'indice per la tabella.
- `FINALIZING`— Ha terminato la creazione dell'indice e sta iniziando ad attivarlo per l'uso.
- `ONLINE`— È attivo e pronto per l'uso nelle interrogazioni. QLDB non utilizza l'indice nelle query finché lo stato non è online.
- `FAILED`— Non è in grado di creare l'indice a causa di un errore irreversibile. Gli indici in questo stato vengono ancora conteggiati nella quota di indici per tabella. Per ulteriori informazioni, consulta Errori comuni.
- `DELETING`— Sta eliminando attivamente l'indice dopo che un utente lo ha eliminato.

- `message`— Il messaggio di errore che descrive il motivo per cui l'indice ha uno`FAILED` stato. Questo campo è incluso se non è valido.

Utilizzo della console

Puoi inoltre utilizzareAWS Management Console per verificare lo stato di un indice.

Verifica dello stato di un indice (console)

1. Accedere allaAWS Management Console e aprire la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel riquadro di navigazione, seleziona Registri.
3. Nell'elenco dei libri contabili, scegli il nome del libro contabile di cui desideri gestire gli indici.
4. Nella pagina dei dettagli del libro contabile, nella scheda Tabelle, scegli il nome della tabella di cui desideri controllare l'indice.
5. Nella pagina dei dettagli della tabella, individua la scheda Campi indicizzati. La colonna Stato dell'indice mostra lo stato corrente di ogni indice della tabella.

Abbassamento degli indici

Usa l'DROP INDEXistruzione per eliminare un indice. Quando si elimina un indice, questo viene eliminato definitivamente dalla tabella.

Innanzitutto, trova l'ID dell'indice da`information_schema.user_tables`. Ad esempio, la seguente query restituisce il valore`indexId` del`LicensePlateNumber` campo indicizzato della`VehicleRegistration` tabella.

```
SELECT indexes.indexId
```

```
FROM information_schema.user_tables info, info.indexes indexes
WHERE info.name = 'VehicleRegistration' and indexes.expr = '[LicensePlateNumber]'
```

Quindi, usa questo ID per eliminare l'indice. Di seguito è riportato un esempio che elimina l'ID dell'indice`4tPW3fUhaVhDinRgKRLhGU`. L'ID dell'indice è un identificatore univoco che deve essere racchiuso tra virgolette.

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

> **Note**
>
> La clausola`WITH (purge = true)` è obbligatoria per tutte le`DROP INDEX` istruzioni ed`true` è attualmente l'unico valore supportato.
> La parola chiave`purge` fa distinzione tra maiuscole e minuscole e deve essere interamente in caratteri minuscoli.

Utilizzo della console

Puoi anche usare il AWS Management Console per eliminare un indice.

Per eliminare un indice (console)

1. Accedere alla AWS Management Console e aprire la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel riquadro di navigazione, seleziona Registri.
3. Nell'elenco dei libri contabili, scegli il nome del libro contabile di cui desideri gestire gli indici.
4. Nella pagina dei dettagli del libro contabile, nella scheda Tabelle, scegli il nome della tabella di cui desideri eliminare l'indice.
5. Nella pagina dei dettagli della tabella, individua la scheda Campi indicizzati. Seleziona l'indice che desideri eliminare, quindi scegli Elimina indice.

Errori comuni

Questa sezione descrive gli errori più comuni che potresti riscontrare durante la creazione di indici e suggerisce possibili soluzioni.

> **Note**
>
> Gli indici con lo stato`FAILED` ancora vengono conteggiati nella quota di indici per tabella. Un indice non riuscito impedisce inoltre di modificare o eliminare i documenti che hanno causato il fallimento della creazione dell'indice nella tabella.
> È necessario eliminare esplicitamente l'indice per rimuoverlo dalla quota.

Il documento contiene più valori per il campo indicizzato: ***`fieldName`***.

QLDB non è in grado di creare un indice per il nome del campo specificato perché la tabella contiene un documento con più valori per lo stesso campo (ovvero nomi di campo duplicati).

È necessario innanzitutto eliminare l'indice non riuscito. Quindi, assicurati che tutti i documenti nella tabella abbiano un solo valore per ogni nome di campo prima di riprovare a creare l'indice. Puoi anche creare un indice per un altro campo che non ha duplicati.

QLDB restituisce questo errore anche se si tenta di inserire un documento che contiene più valori per un campo già indicizzato nella tabella.

Limite di indici superato: ***`Table TableName`*** ha già ***`n`*** indici e non può crearne altri.

QLDB impone un limite di cinque indici per tabella, inclusi gli indici con errori. È necessario eliminare un indice esistente prima di crearne uno nuovo.

Nessun indice definito con identificatore: ***`indexID`***.

Hai provato a eliminare un indice che non esiste per la combinazione di tabella e ID indice specificata. Per informazioni su come controllare gli indici esistenti, consultaDescrizione degli indici.

ID univoci in Amazon QLDB

Questa sezione descrive le proprietà e le linee guida per l'uso degli identificatori univoci assegnati dal sistema in Amazon QLDB. Fornisce anche alcuni esempi di ID univoci QLDB.

Argomenti

- Proprietà
- Utilizzo

Proprietà

Tutti gli ID assegnati dal sistema in QLDB sono identificatori univoci universali (UUID). Ogni ID presenta le seguenti proprietà:

- Numero UUID a 128 bit
- Rappresentato in testo codificato in Base62
- Stringa alfanumerica a lunghezza fissa di 22 caratteri (ad esempio:`3Qv67yjXEwB9SjmvkuG6Cp`)

Utilizzo

Quando usi ID univoci QLDB nella tua applicazione, tieni presente le seguenti linee guida:

Fare

- Tratta l'ID come una stringa.

Non

- Prova a decodificare la stringa.
- Assegna un significato semantico alla stringa (come derivare un componente temporale).
- Ordina le stringhe in ordine semantico.

Esempi

I seguenti attributi sono alcuni esempi di ID univoci in QLDB:

- ID del documento
- ID dell'indice
- ID del filo
- ID tabella
- ID della transazione

Modello di concorrenza Amazon QLDB

Amazon QLDB è una valida scelta per i carichi di transazioni online (OLTP) ad alte prestazioni QLDB. QLDB supporta funzionalità di query di tipo SQL e fornisce transazioni ACID complete. Inoltre, gli elementi di dati QLDB sono documenti che offrono flessibilità dello schema e una modellazione intuitiva dei dati. Con un diario come base, puoi utilizzare QLDB per accedere alla cronologia completa e verificabile di tutte le modifiche ai tuoi dati e trasmettere transazioni coerenti ad altri servizi di dati, se necessario.

Argomenti

Controllo ottimistico della concorrenza

In QLDB, il controllo della concorrenza viene implementato utilizzando il controllo della concorrenza ottimistica (OCC). L'OCC opera secondo il principio che più transazioni possono spesso essere completate senza interferire l'una con l'altra.

Utilizzando OCC, le transazioni in QLDB non acquisiscono blocchi sulle risorse del database e funzionano con un isolamento serializzabile completo. QLDB esegue transazioni simultanee in modo seriale, in modo tale da produrre lo stesso effetto come se tali transazioni fossero avviate in serie.

Prima di eseguire il commit, ogni transazione esegue un controllo di convalida per garantire che nessun'altra transazione confermata abbia modificato i dati a cui accede. Se questo controllo rivela modifiche in conflitto o lo stato dei dati cambia, la transazione di conferma viene rifiutata. Tuttavia, la transazione può essere riavviata.

Quando una transazione scrive su QLDB, i controlli di convalida del modello OCC vengono implementati da QLDB stesso. Se una transazione non può essere scritta nel journal a causa di un errore nella fase di verifica di OCC, QLDB restituisce an`OccConflictException` al livello

dell'applicazione. Il software applicativo è responsabile del riavvio della transazione. L'applicazione deve interrompere la transazione rifiutata e quindi riprovare l'intera transazione dall'inizio.

Per informazioni su come il driver QLDB gestisce e riprova i conflitti OCC e altre eccezioni transitorie, vedereInformazioni sulla politica di riprova con il driver in Amazon QLDB.

Utilizzo degli indici per evitare la scansione completa delle tabelle

In QLDB, ogni istruzione PartiQL (inclusa ogni`SELECT` query) viene elaborata in una transazione ed è soggetta a un limite di timeout della transazione.

Come procedura consigliata, è consigliabile eseguire istruzioni con una clausola`WHERE` predicativa che filtra in base a un campo indicizzato o all'ID di un documento. QLDB richiede un operatore di uguaglianza su un campo indicizzato per cercare in modo efficiente un documento; ad esempio,`WHERE indexedField = 123` o`WHERE indexedField IN (456, 789)`.

Senza questa ricerca indicizzata, QLDB deve eseguire una scansione completa della tabella durante la lettura dei documenti. Ciò può causare latenza delle query e timeout delle transazioni e aumentare anche le possibilità di un conflitto OCC con transazioni concorrenti.

Ad esempio, considera una tabella denominata`Vehicle` che ha un indice solo nel`VIN` campo. Contiene i seguenti documenti.

VIN	Make	Modello	Colore
`"1N4AL11D 75C109151"`	`"Audi"`	`"A5"`	`"Silver"`
`"KM8SRDHF 6EU074761"`	`"Tesla"`	`"Model S"`	`"Blue"`
`"3HGGK5G5 3FM761765"`	`"Ducati"`	`"Monster 1200"`	`"Yellow"`
`"1HVBBAAN XWH544237"`	`"Ford"`	`"F 150"`	`"Black"`
`"1C4RJFAG 0FC625797"`	`"Mercedes"`	`"CLK 350"`	`"White"`

Due utenti simultanei di nome Alice e Bob stanno lavorando con la stessa tabella in un libro mastro. Vogliono aggiornare due documenti diversi, come segue.

Alice:

```
UPDATE Vehicle AS v
SET v.Color = 'Blue'
WHERE v.VIN = '1N4AL11D75C109151'
```

Bob:

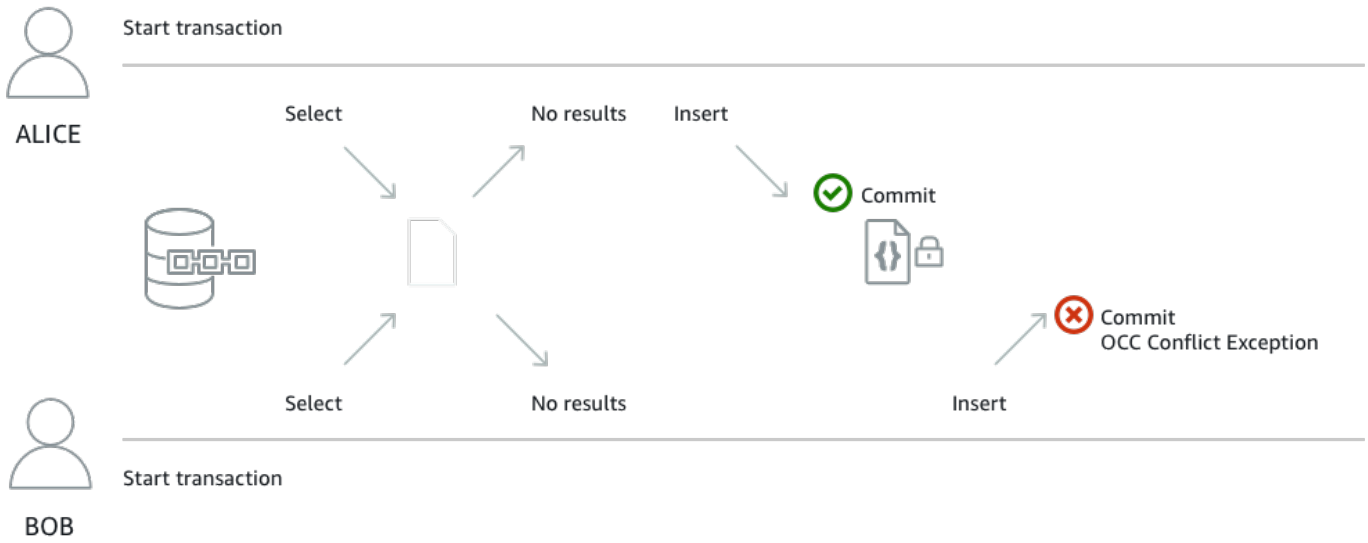
```
UPDATE Vehicle AS v
SET v.Color = 'Red'
WHERE v.Make = 'Tesla' AND v.Model = 'Model S'
```

Supponiamo che Alice e Bob inizino le loro transazioni contemporaneamente. La`UPDATE` dichiarazione di Alice esegue una ricerca indicizzata sul`VIN` campo, quindi deve leggere solo quel documento. Alice termina e porta a termine con successo la sua transazione per prima.

L'istruzione di Bob filtra i campi non indicizzati, quindi esegue una scansione della tabella e incontra un`OccConflictException`. Questo perché la transazione effettuata da Alice ha modificato i dati a cui accede la dichiarazione di Bob, che includono tutti i documenti della tabella, non solo il documento che Bob sta aggiornando.

Conflitti di inserimento OCC

I conflitti OCC possono includere documenti appena inseriti, non solo documenti che esistevano in precedenza. Si consideri il diagramma seguente, in cui due utenti in contemporanea (Alice e Bob) stanno lavorando con la stessa tabella in un registro. Entrambi vogliono inserire un nuovo documento solo a condizione che non esista ancora un valore predicato.



In questo esempio, sia Alice che Bob eseguono le seguentiINSERT istruzioniSELECT e all'interno di una singola transazione. La loro applicazione esegue l'INSERTistruzione solo se l'SELECTistruzione non restituisce risultati.

```
SELECT * FROM Vehicle v WHERE v.VIN = 'ABCDE12345EXAMPLE'
```

```
INSERT INTO Vehicle VALUE
{
    'VIN' : 'ABCDE12345EXAMPLE',
    'Type' : 'Wagon',
    'Year' : 2019,
    'Make' : 'Subaru',
    'Model' : 'Outback',
    'Color' : 'Gray'
}
```

Supponiamo che Alice e Bob inizino le loro transazioni contemporaneamente. Entrambe le loroSELECT domande non restituiscono alcun documento esistente con unVIN diABCDE12345EXAMPLE. Quindi, le loro candidature procedono con laINSERT dichiarazione.

Alice termina e porta a termine con successo la sua transazione per prima. Quindi, Bob prova a confermare la sua transazione, ma QLDB la rifiuta e lancia un0ccConflictException. Questo perché la transazione confermata da Alice ha modificato il set di risultati dellaSELECT query di Bob e OCC rileva questo conflitto prima di confermare la transazione di Bob.

L'`SELECT`interrogazione è necessaria affinché questo esempio di transazione sia idempotente. Bob può quindi riprovare l'intera transazione dall'inizio. Ma la sua prossima`SELECT` interrogazione restituirà il documento inserito da Alice, quindi l'applicazione di Bob non eseguirà il`INSERT`.

Rendere le transazioni idempotenti

La transazione di inserimento nella sezione precedente è anche un esempio di transazione idempotente. In altre parole, eseguire la stessa transazione più volte produce risultati identici. Se Bob esegue il file`INSERT` senza prima verificare se un particolare esiste`VIN` già, la tabella potrebbe finire con documenti con`VIN` valori duplicati.

Prendi in considerazione altri scenari di nuovo tentativo oltre ai conflitti OCC. Ad esempio, è possibile che QLDB esegua correttamente una transazione sul lato server, ma il client scada in attesa di una risposta. Come buona prassi, rendi le tue transazioni di scrittura idempotenti per evitare effetti collaterali imprevisti in caso di concorrenza o nuovi tentativi.

Conflitti OCC di redazione

QLDB impedisce le redazioni simultanee delle revisioni sullo stesso blocco del diario. Consideriamo un esempio in cui due utenti simultanei (Alice e Bob) desiderano redigere due diverse revisioni di documenti che vengono salvate sullo stesso blocco di un libro mastro. Innanzitutto, Alice richiede la redazione di una revisione eseguendo la procedura`REDACT_REVISION` memorizzata, come segue.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

Quindi, mentre la richiesta di Alice è ancora in elaborazione, Bob richiede la redazione di un'altra revisione, come segue.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

QLDB respinge la richiesta di Bob con un`OccConflictException` anche se stanno cercando di oscurare due diverse revisioni del documento. Questo perché la revisione di Bob si trova nello stesso blocco della revisione che Alice sta oscurando. Al termine dell'elaborazione della richiesta di Alice, Bob può quindi riprovare la sua richiesta di redazione.

Allo stesso modo, se due transazioni simultanee tentano di oscurare la stessa revisione, può essere elaborata una sola richiesta. L'altra richiesta fallisce con un'eccezione di conflitto OCC fino al

completamento della redazione. In seguito, qualsiasi richiesta di rimozione della stessa revisione genererà un errore che indica che la revisione è già stata redatta.

Gestione delle sessioni concorrenti

Se hai esperienza nell'uso di un sistema di gestione di database relazionali (RDBMS, Relational Database Management System), potresti avere una valida scelta per i limiti di connessioni simultanee. QLDB non ha lo stesso concetto di connessione RDBMS tradizionale perché le transazioni vengono eseguite con messaggi di richiesta e risposta HTTP.

In QLDB, il concetto analogo è una sessione attiva. Una sessione è concettualmente simile all'accesso di un utente: gestisce le informazioni sulle richieste di transazione di dati su un libro mastro. Una sessione attiva è una sessione che esegue attivamente una transazione. Può anche trattarsi di una sessione che ha recentemente terminato una transazione in cui il servizio prevede che ne avvierà immediatamente un'altra. QLDB supporta una transazione in esecuzione attiva per sessione.

Il limite di sessioni attive simultanee per registro è definito inQuote e limiti in Amazon QLDB. Una volta raggiunto questo limite, qualsiasi sessione che tenti di avviare una transazione genererà un errore (`LimitExceededException`).

Per informazioni sul ciclo di vita di una sessione e su come il driver QLDB gestisce le sessioni durante l'esecuzione di transazioni di dati, vedereGestione delle sessioni con il conducente. Per le best practice per configurare un pool di sessioni nell'applicazione utilizzando il driver QLDB, consulta iConfigurazione della QldbDriver oggetto consigli sui driver Amazon QLDB.

Verifica dei dati in Amazon QLDB

Con Amazon QLDB, puoi essere certo che la cronologia delle modifiche ai dati delle tue applicazioni sia accurata. QLDB utilizza un log transazionale immutabile, noto come journal, per l'archiviazione dei dati. Il diario tiene traccia di ogni modifica ai dati impegnati e mantiene una cronologia completa e verificabile delle modifiche nel tempo.

QLDB utilizza la funzione hash SHA-256 con un modello basato sull'albero Merkle per generare una rappresentazione crittografica del diario, nota come digest. Il digest funge da firma univoca dell'intera cronologia delle modifiche dei dati a partire da un determinato momento. Il digest viene utilizzato per verificare l'integrità delle revisioni dei documenti relative a tale firma.

Argomenti

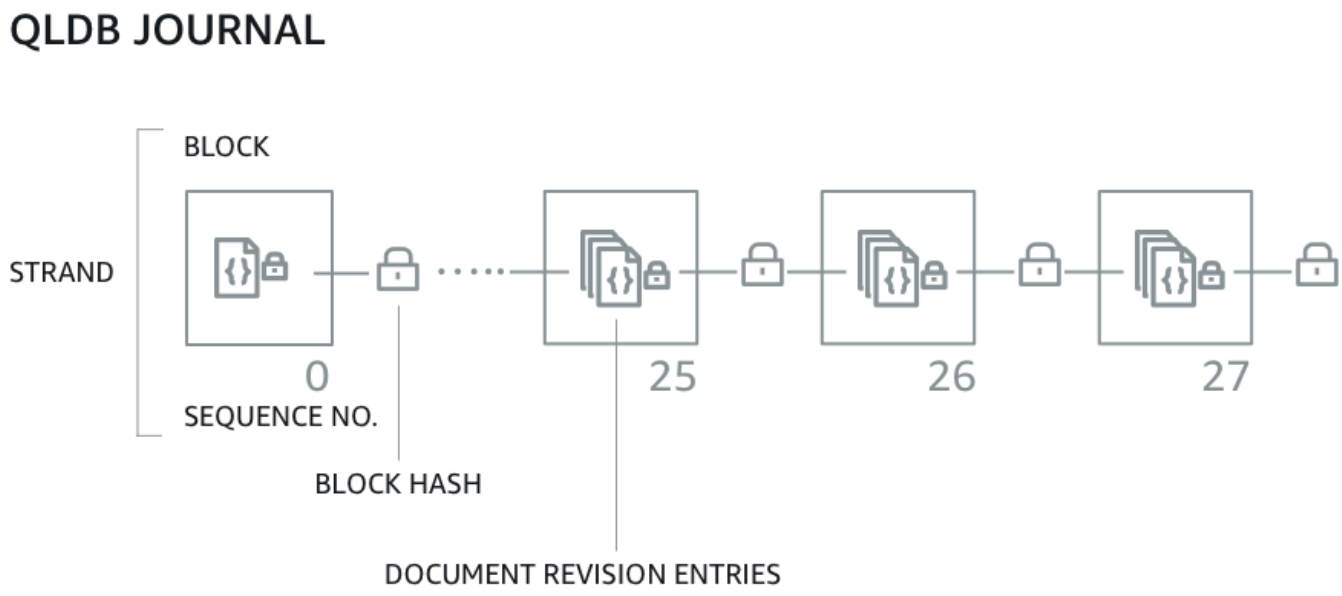
Che tipo di dati è possibile verificare in QLDB?

In QLDB, ogni libro mastro ha esattamente un giornale. Un diario può avere più filoni, che sono partizioni del diario.

> **Note**
>
> QLDB attualmente supporta solo riviste con un solo filone.

Un blocco è un oggetto che viene inserito nel filamento del journal durante una transazione. Questo blocco contiene oggetti di immissione, che rappresentano le revisioni del documento risultanti dalla transazione. È possibile verificare una singola revisione o un intero blocco di journal in QLDB.

Il diagramma seguente illustra questa struttura del diario.



Il diagramma mostra che le transazioni vengono salvate nel journal come blocchi contenenti le voci di revisione del documento. Mostra anche che ogni blocco è concatenato ad hash ai blocchi successivi e ha un numero di sequenza per specificare il suo indirizzo all'interno del filamento.

Per informazioni sul contenuto dei dati in un blocco, vedere. Contenuto del diario in Amazon QLDB

Cosa significa integrità dei dati?

L'integrità dei dati in QLDB significa che il diario del registro è di fatto immutabile. In altre parole, i tuoi dati (in particolare, ogni revisione del documento) si trovano in uno stato in cui sono vere le seguenti condizioni:

1. Esiste nella stessa posizione del diario in cui è stato scritto per la prima volta.
2. Non è stato alterato in alcun modo da quando è stato scritto.

Come funziona la verifica?

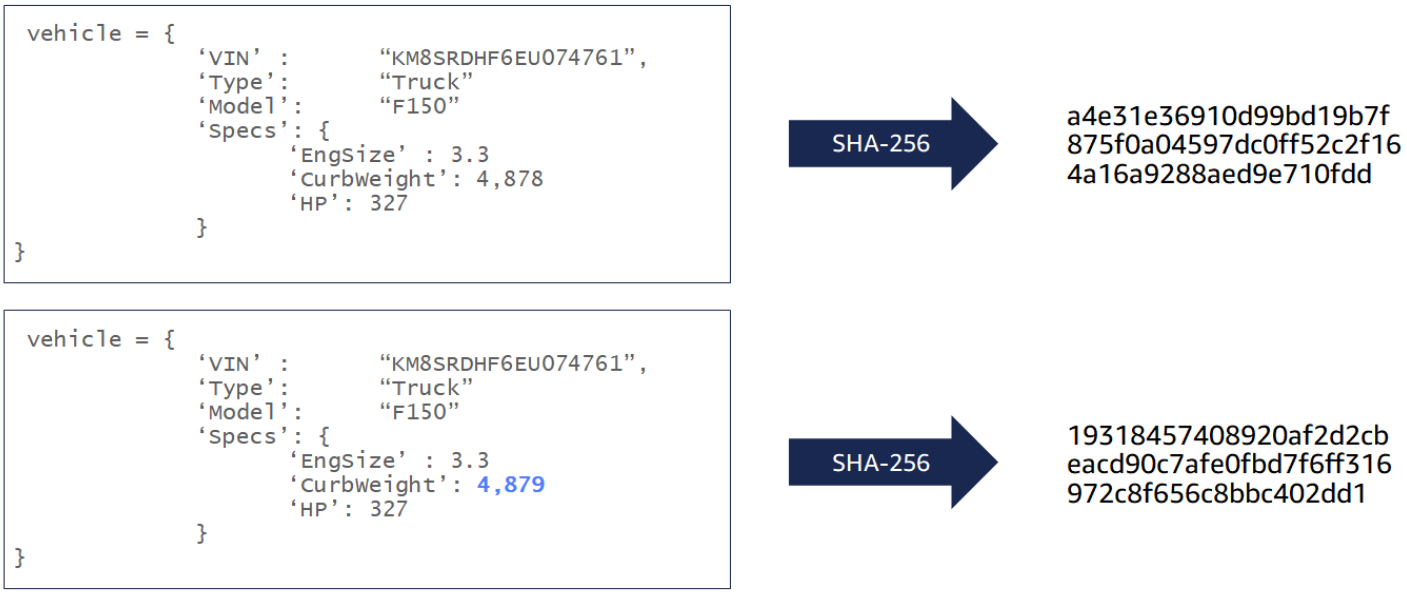
Per capire come funziona la verifica in Amazon QLDB, puoi suddividere il concetto in quattro componenti di base.

- Hashing
- Digest
- Albero Merkle
- Prova

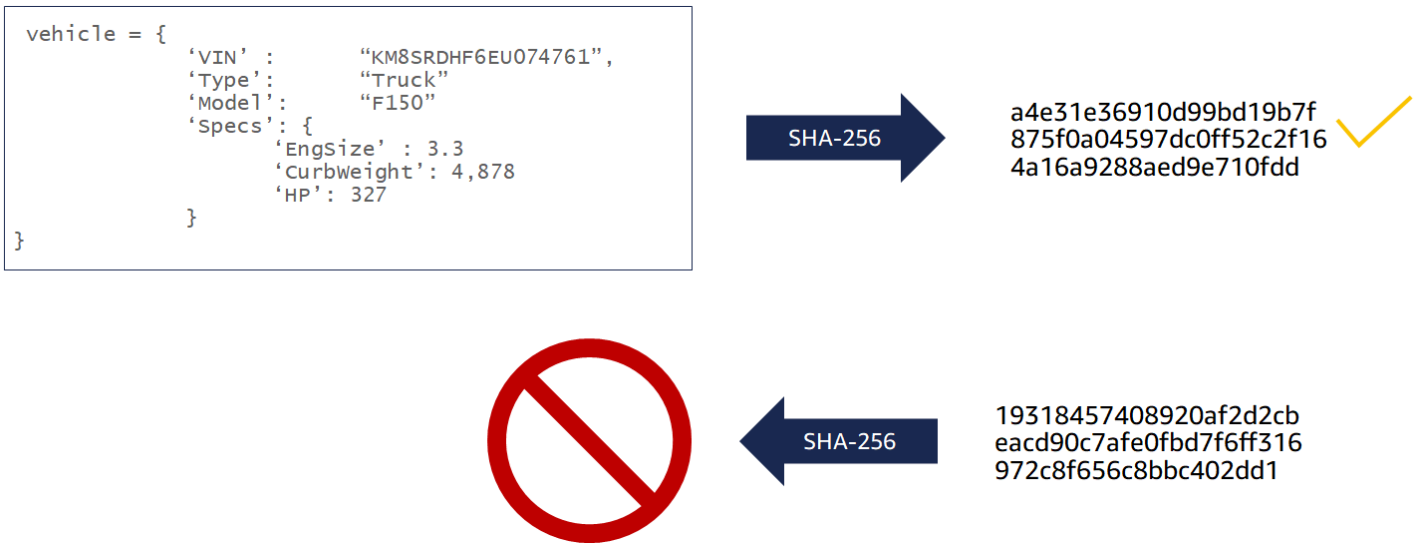
Hashing

QLDB utilizza la funzione hash crittografica SHA-256 per creare valori hash a 256 bit. Un hash funge da firma unica a lunghezza fissa di qualsiasi quantità arbitraria di dati di input. Se si modifica qualsiasi parte dell'input, anche un singolo carattere o bit, l'hash di output cambia completamente.

Il diagramma seguente mostra che la funzione hash SHA-256 crea valori hash completamente unici per due documenti QLDB che differiscono solo di una cifra.



La funzione hash SHA-256 è unidirezionale, il che significa che non è matematicamente possibile calcolare l'input quando viene fornito un output. Il diagramma seguente mostra che non è possibile calcolare il documento QLDB di input quando viene fornito un valore hash di output.



I seguenti input di dati vengono sottoposti a hash in QLDB a scopo di verifica:

- Revisioni del documento
- Istruzioni PartiQL
- Voci di revisione
- Blocchi diario

Digest

Un digest è una rappresentazione crittografica dell'intero diario del registro in un determinato momento. Un diario è di sola aggiunta e i blocchi di journal sono sequenziati e concatenati in hash in modo simile alle blockchain.

Puoi richiedere un riassunto per un libro mastro in qualsiasi momento. QLDB genera il digest e lo restituisce come file di output sicuro. Quindi si utilizza tale digest per verificare l'integrità delle revisioni dei documenti eseguite in un momento precedente. Se ricalcola gli hash iniziando con una revisione e terminando con il digest, dimostri che i dati non sono stati alterati nel frattempo.

Albero Merkle

Con l'aumentare delle dimensioni del registro, diventa sempre più inefficiente ricalcolare l'intera catena hash del diario a scopo di verifica. QLDB utilizza un modello ad albero Merkle per risolvere questa inefficienza.

Un albero Merkle è una struttura dati ad albero in cui ogni nodo foglia rappresenta un hash di un blocco di dati. Ogni nodo non foglia è un hash dei suoi nodi figli. Comunemente utilizzato nelle blockchain, un albero Merkle aiuta a verificare in modo efficiente set di dati di grandi dimensioni con un meccanismo a prova di audit. Per ulteriori informazioni sugli alberi Merkle, consulta la pagina Wikipedia di Merkle Tree. Per saperne di più sulle bozze di audit Merkle e per un esempio di utilizzo, consulta How Log Proofs Work sul sito Certificate Transparency.

L'implementazione QLDB dell'albero Merkle è costruita a partire dalla catena hash completa di un giornale. In questo modello, i nodi foglia sono l'insieme di tutti gli hash di revisione dei singoli documenti. Il nodo radice rappresenta il riepilogo dell'intero diario a partire da un determinato momento.

Utilizzando una bozza di verifica Merkle, puoi verificare una revisione controllando solo un piccolo sottoinsieme della cronologia delle revisioni del tuo libro mastro. Puoi farlo attraversando l'albero da un determinato nodo fogliare (revisione) alla sua radice (digest). Lungo questo percorso di attraversamento, esegui l'hash ricorsivo di coppie di nodi di pari livello per calcolare l'hash principale fino al termine del digest. Questo attraversamento presenta una complessità temporale dei nodi dell'albero. `log(n)`

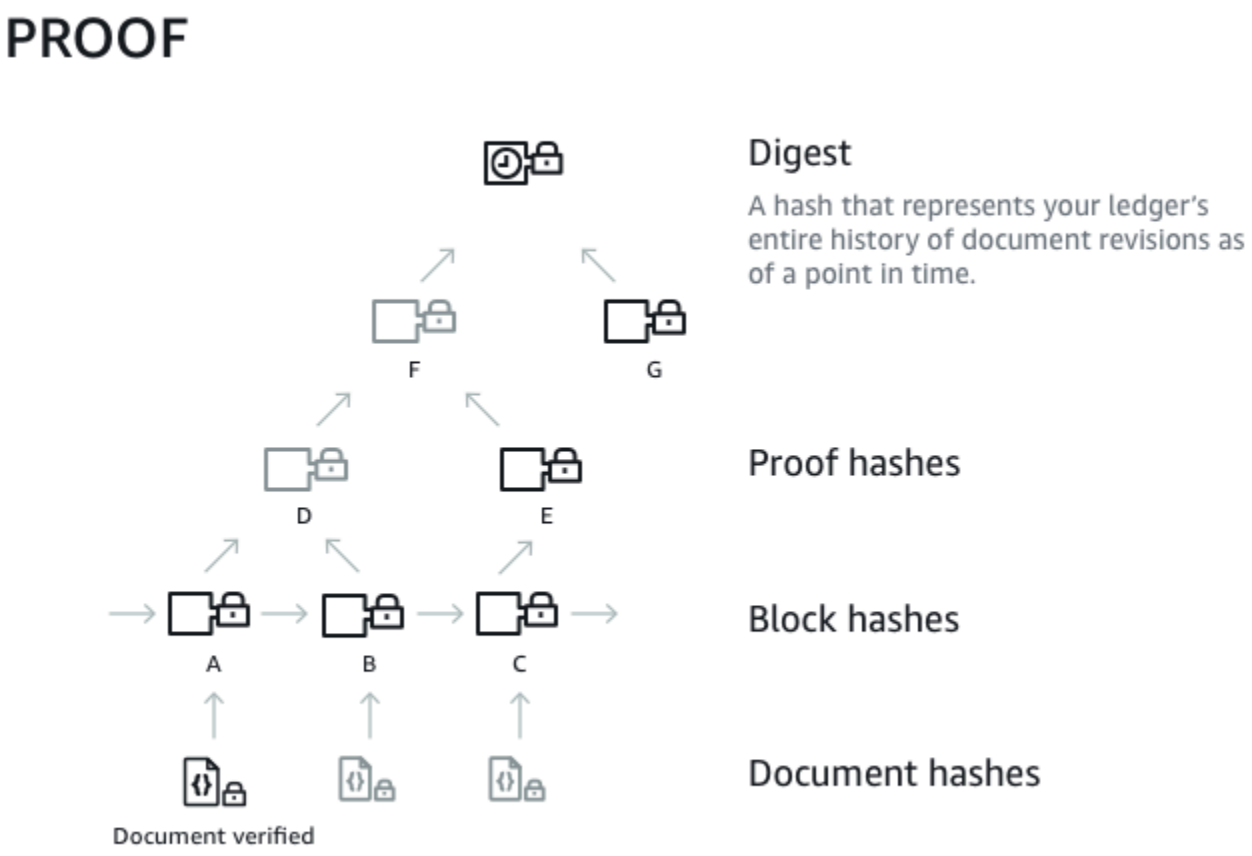
Prova

Una prova è l'elenco ordinato degli hash dei nodi che QLDB restituisce per un determinato digest e una determinata revisione del documento. È costituito dagli hash richiesti da un modello di albero di Merkle per concatenare l'hash del nodo foglia specificato (una revisione) all'hash root (il digest).

La modifica dei dati confermati tra una revisione e un digest interrompe la catena di hash del diario e rende impossibile la generazione di una bozza.

Esempio di verifica

Il diagramma seguente illustra il modello di albero hash di Amazon QLDB. Mostra una serie di hash a blocchi che arrivano al nodo principale principale, che rappresenta il digest di un filone di journal. In un registro con un diario a filamento singolo, questo nodo radice è anche il riepilogo dell'intero registro.



Supponiamo che il nodo A sia il blocco che contiene la revisione del documento di cui desideri verificare l'hash. I seguenti nodi rappresentano l'elenco ordinato di hash che QLDB fornisce nella dimostrazione: B, E, G. Questi hash sono necessari per ricalcolare il digest dall'hash A.

Per ricalcolare il digest, procedi come segue:

1. Iniziate con l'hash A e concatenatelo con l'hash B. Quindi, esegui l'hash del risultato per calcolare D.
2. Usa D ed E per calcolare F.
3. Usa F e G per calcolare il digest.

La verifica ha esito positivo se il digest ricalcolato corrisponde al valore previsto. Con un hash di revisione e un digest, non è possibile decodificare gli hash in una bozza. Pertanto, questo esercizio dimostra che la revisione è stata effettivamente scritta in questa posizione del diario rispetto al digest.

In che modo la redazione dei dati influisce sulla verifica?

In Amazon QLDB, `DELETE` un'istruzione elimina logicamente un documento solo creando una nuova revisione che lo contrassegna come eliminato. QLDB supporta anche un'operazione di redazione dei dati che consente di eliminare definitivamente le revisioni inattive dei documenti nella cronologia di una tabella.

L'operazione di redazione elimina solo i dati utente nella revisione specificata e lascia invariati la sequenza del diario e i metadati del documento. Dopo la redazione di una revisione, i dati utente nella revisione (rappresentati dalla `data` struttura) vengono sostituiti da un nuovo campo. `dataHash` Il valore di questo campo è l'hash Amazon Ion della `data` struttura rimossa. Per ulteriori informazioni e un esempio di operazione di redazione, consulta. Redazione delle revisioni dei documenti

Di conseguenza, il registro mantiene l'integrità complessiva dei dati e rimane verificabile crittograficamente tramite le operazioni API di verifica esistenti. Puoi comunque utilizzare queste operazioni API come previsto per richiedere un digest (GetDigest), richiedere una prova (GetBlocko GetRevision) e quindi eseguire l'algoritmo di verifica utilizzando gli oggetti restituiti.

Ricalcolo dell'hash di revisione

Se intendi verificare la revisione di un singolo documento ricalcolandone l'hash, devi verificare in modo condizionale se la revisione è stata oscurata. Se la revisione è stata redatta, puoi utilizzare il valore hash fornito nel campo. `dataHash` Se non è stata oscurata, puoi ricalcolare l'hash utilizzando il campo. `data`

Eseguendo questo controllo condizionale, puoi identificare le revisioni redatte e intraprendere le azioni appropriate. Ad esempio, è possibile registrare gli eventi di manipolazione dei dati per scopi di monitoraggio.

Guida introduttiva alla verifica

Prima di poter verificare i dati, è necessario richiedere un riepilogo dal registro e salvarlo per utilizzarlo in un secondo momento. Qualsiasi revisione del documento effettuata prima dell'ultimo blocco coperto dal digest è idonea per la verifica rispetto a tale digest.

Quindi, richiedi una bozza ad Amazon QLDB per una revisione idonea che desideri verificare. Utilizzando questa bozza, richiami un'API lato client per ricalcolare il digest, a partire dall'hash della revisione. Finché il digest salvato in precedenza è noto e affidabile al di fuori di QLDB, l'integrità del documento è dimostrata se l'hash del digest ricalcolato corrisponde all'hash del digest salvato.

Important

- Ciò che stai dimostrando in particolare è che la revisione del documento non è stata alterata tra il momento in cui hai salvato questo riassunto e quello in cui hai eseguito la verifica. Puoi richiedere e salvare un digest non appena una revisione che desideri verificare in un secondo momento viene salvata nel diario.
- Come procedura ottimale, si consiglia di richiedere regolarmente i riassunti e di conservarli lontano dal registro. Determina la frequenza con cui richiedi i riassunti in base alla frequenza con cui esegui le revisioni nel registro.

  Per un post dettagliato AWS sul blog che illustra il valore della verifica crittografica nel contesto di un caso d'uso realistico, consulta Verifica crittografica nel mondo reale con Amazon QLDB.

Per step-by-step guide su come richiedere un riepilogo dal registro e quindi verificare i dati, consulta quanto segue:

- Fase 1: Richiedere un digest in QLDB
- Fase 2: Verifica dei dati in QLDB

Fase 1: Richiedere un digest in QLDB

Amazon QLDB fornisce un'API per richiedere un digest che copra l'ultima parte del diario del registro. Il tip of the journal si riferisce all'ultimo blocco eseguito nel momento in cui QLDB riceve la tua richiesta. Puoi usare il AWS Management Console, un AWS SDK o il AWS Command Line Interface (AWS CLI) per ottenere un digest.

Argomenti

- AWS Management Console
- API QLDB

AWS Management Console

Segui questi passaggi per richiedere un digest utilizzando la console QLDB.

Per richiedere un digest (console)

1. Accedi a e apri AWS Management Console la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel pannello di navigazione, scegli Ledgers.
3. Nell'elenco dei libri contabili, seleziona il nome del libro contabile per il quale desideri richiedere un digest.
4. Scegli Get digest. La finestra di dialogo Get digest mostra i seguenti dettagli del digest:
   - Digest: il valore hash SHA-256 del digest richiesto.
   - Indirizzo del suggerimento del digest: l'ultima posizione del blocco nel diario coperta dal digest richiesto. Un indirizzo ha i due campi seguenti:
     - `strandId`— L'ID univoco del filone di giornale che contiene il blocco.
     - `sequenceNo`— Il numero indice che specifica la posizione del blocco all'interno del trefolo.
   - Libro contabile: il nome del libro contabile per il quale è stato richiesto un digest.
   - Data: il timestamp in cui è stato richiesto il digest.
5. Esamina le informazioni del riepilogo. Quindi scegli Save (Salva). È possibile mantenere il nome di file predefinito o immettere un nuovo nome.

> **Note**
>
> Potresti notare che i valori dell'hash del digest e dell'indirizzo tip cambiano anche quando non modifichi alcun dato nel tuo registro. Questo perché la console recupera il catalogo di sistema del registro ogni volta che si esegue una query nell'editor PartiQL. Si tratta di una transazione di lettura che viene salvata nel journal e causa la modifica dell'ultimo indirizzo di blocco.

Questo passaggio salva un file di testo semplice con contenuti in formato Amazon Ion. Il file ha un'estensione di `.ion.txt` e contiene tutte le informazioni digest elencate nella finestra di dialogo precedente. Di seguito è riportato un esempio del contenuto di un file digest. L'ordine dei campi può variare a seconda del browser utilizzato.

```
{
  "digest": "42zaJOfV8iGutVGNaIuzQWhD5Xb/5B9lScHnvxPXm9E=",
  "digestTipAddress": "{strandId:\"BlFTjlSXze9BIh1KOszcE3\",sequenceNo:73}",
```

```
  "ledger": "my-ledger",
  "date": "2019-04-17T16:57:26.749Z"
}
```

6. Salva questo file dove potrai accedervi in futuro. Successivamente, puoi utilizzare questo file per verificare la revisione di un documento.

> **Important**
>
> La revisione del documento che verificherete in seguito deve essere inclusa nel digest salvato. Cioè, il numero di sequenza dell'indirizzo del documento deve essere inferiore o uguale al numero di sequenza dell'indirizzo del suggerimento del Digest.

API QLDB

Puoi anche richiedere un digest dal tuo registro utilizzando l'API Amazon QLDB con un SDK o il. AWS AWS CLI L'API QLDB fornisce le seguenti operazioni per l'uso da parte dei programmi applicativi:

- GetDigest— Restituisce il riassunto di un libro mastro all'ultimo blocco commesso nel diario. La risposta include un valore hash a 256 bit e un indirizzo di blocco.

Per informazioni sulla richiesta di un digest utilizzando il AWS CLI, vedete il comando get-digest nel Command Reference.AWS CLI

Applicazione di esempio

Per esempi di codice Java, consulta il repository aws-samples/ -java. GitHub amazon-qldb-dmv-sample Per istruzioni su come scaricare e installare questa applicazione di esempio, consulta. Installazione dell'applicazione di esempio Java Amazon QLDB Prima di richiedere un digest, assicurati di seguire i passaggi da 1 a 3 Tutorial su Java per creare un registro di esempio e caricarlo con dati di esempio.

Il codice del tutorial in classe GetDigestfornisce un esempio di richiesta di un digest dal registro di esempio. `vehicle-registration`

Per verificare la revisione di un documento utilizzando il digest salvato, procedi a. Fase 2: Verifica dei dati in QLDB

Fase 2: Verifica dei dati in QLDB

Amazon QLDB fornisce un'API per richiedere una prova per un ID documento specifico e il blocco associato. È inoltre necessario fornire l'indirizzo tip di un digest salvato in precedenza, come descritto in. Fase 1: Richiedere un digest in QLDB Puoi usare il AWS Management Console, un AWS SDK o il AWS CLI per ottenere una prova.

Quindi, puoi utilizzare la bozza restituita da QLDB per verificare la revisione del documento rispetto al digest salvato, utilizzando un'API lato client. In questo modo puoi controllare l'algoritmo che utilizzi per verificare i tuoi dati.

Argomenti

- AWS Management Console
- API QLDB

AWS Management Console

Questa sezione descrive i passaggi per verificare una revisione del documento rispetto a un digest salvato in precedenza utilizzando la console Amazon QLDB.

Prima di iniziare, assicurati di seguire i passaggi riportati di seguito. Fase 1: Richiedere un digest in QLDB La verifica richiede un digest salvato in precedenza che copra la revisione che desideri verificare.

Per verificare la revisione di un documento (console)

1. Apri la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Per prima cosa, interroga il registro per verificare `id` la `blockAddress` fine della revisione che desideri verificare. Questi campi sono inclusi nei metadati del documento, sui quali è possibile interrogare nella visualizzazione confermata.

   Il documento `id` è una stringa ID univoca assegnata dal sistema. `blockAddress`È una struttura ionica che specifica la posizione del blocco in cui è stata eseguita la revisione.

   Nel pannello di navigazione, scegli Editor PartiQl.
3. Scegli il nome del libro mastro in cui desideri verificare una revisione.
4. Nella finestra dell'editor di query, inserisci un'**`SELECT`**istruzione con la seguente sintassi, quindi scegli Esegui.

```
SELECT metadata.id, blockAddress FROM _ql_committed_table_name
WHERE criteria
```

Ad esempio, la seguente query restituisce un documento dalla `VehicleRegistration` tabella del libro mastro di esempio creato in. Nozioni di base sulla console Amazon QLDB

```
SELECT r.metadata.id, r.blockAddress FROM _ql_committed_VehicleRegistration AS r
WHERE r.data.VIN = 'KM8SRDHF6EU074761'
```

5. Copia e salva i `blockAddress` valori `id` and restituiti dalla tua query. Assicurati di omettere le virgolette doppie per il `id` campo. In Amazon Ion, i tipi di dati stringa sono delimitati da virgolette doppie. Ad esempio, devi copiare solo il testo alfanumerico nel frammento seguente.

   "*LtMNJYNjSwzBLgf7sLifrG*"

6. Ora che avete selezionato una revisione del documento, potete iniziare il processo di verifica.

   Nel riquadro di navigazione, scegli Verifica.

7. Nel modulo Verifica documento, in Specificare il documento che desideri verificare, inserisci i seguenti parametri di input:

   - Libro contabile: il libro mastro in cui si desidera verificare una revisione.
   - Indirizzo di blocco: il `blockAddress` valore restituito dalla query nel passaggio 4.
   - ID documento: il `id` valore restituito dalla query nel passaggio 4.

8. In Specificare il digest da utilizzare per la verifica, seleziona il digest salvato in precedenza scegliendo Scegli digest. Se il file è valido, compila automaticamente tutti i campi digest sulla console. In alternativa, puoi copiare e incollare manualmente i seguenti valori direttamente dal tuo file digest:

   - Digest: il `digest` valore del file digest.
   - Indirizzo del suggerimento digest: il `digestTipAddress` valore del file digest.

9. Esamina i parametri di input del documento e digest, quindi scegli Verifica.

   La console automatizza due passaggi per te:

   a. Richiedi una bozza a QLDB per il documento specificato.

b. Usa la prova restituita da QLDB per chiamare un'API lato client, che verifica la revisione del documento rispetto al digest fornito. Per esaminare questo algoritmo di verifica, consulta la sezione seguente per scaricare l'esempio di codice. API QLDB

La console visualizza i risultati della richiesta nella scheda dei risultati della verifica. Per ulteriori informazioni, consulta Risultati della verifica.

API QLDB

Puoi anche verificare la revisione di un documento utilizzando l'API Amazon QLDB con AWS un SDK o il. AWS CLI L'API QLDB fornisce le seguenti operazioni per l'uso da parte dei programmi applicativi:

- `GetDigest`— Restituisce il riassunto di un libro mastro all'ultimo blocco commesso nel diario. La risposta include un valore hash a 256 bit e un indirizzo di blocco.
- `GetBlock`— Restituisce un oggetto blocco a un indirizzo specificato in un diario. Restituisce inoltre una prova del blocco specificato per la verifica, se `DigestTipAddress` fornita.
- `GetRevision`— Restituisce un oggetto di dati di revisione per un ID di documento e un indirizzo di blocco specificati. Restituisce inoltre una prova della revisione specificata per la verifica, se `DigestTipAddress` fornita.

Per una descrizione completa di queste operazioni API, consultaDocumentazione di riferimento dell'API Amazon QLDB.

Per informazioni sulla verifica dei dati utilizzando il AWS CLI, consulta il AWS CLI Command Reference.

Applicazione di esempio

Per esempi di codice Java, consulta il GitHub repository amazon-qldb-dmv-sampleaws-samples/ -java. Per istruzioni su come scaricare e installare questa applicazione di esempio, consulta. Installazione dell'applicazione di esempio Java Amazon QLDB Prima di effettuare una verifica, assicurati di seguire i passaggi da 1 a 3 Tutorial su Java per creare un registro di esempio e caricarlo con dati di esempio.

Il codice del tutorial in classe GetRevisionfornisce un esempio di richiesta di bozza per la revisione di un documento e quindi di verifica di tale revisione. Questa classe esegue i seguenti passaggi:

1. Richiede un nuovo digest dal registro `vehicle-registration` di esempio.
2. Richiede una bozza per un esempio di revisione di un documento dalla `VehicleRegistration` tabella del libro mastro. `vehicle-registration`
3. Verifica la revisione del campione utilizzando il digest e la bozza restituiti.

Risultati della verifica

Questa sezione descrive i risultati restituiti da una richiesta di verifica dei dati Amazon QLDB su. AWS Management Console Per i passaggi dettagliati su come inviare una richiesta di verifica, consulta. Fase 2: Verifica dei dati in QLDB

Nella pagina di verifica della console QLDB, i risultati della richiesta vengono visualizzati nella scheda dei risultati della verifica. La scheda Prova mostra il contenuto della bozza restituita da QLDB per la revisione e il riassunto del documento specificati. Include i seguenti dettagli:

- Hash di revisione: il valore SHA-256 che rappresenta in modo univoco la revisione del documento che stai verificando.
- Hash di prova: l'elenco ordinato di hash fornito da QLDB che vengono utilizzati per ricalcolare il digest specificato. La console inizia con l'hash Revision e lo combina in sequenza con ogni hash proof fino a quando non termina con un digest ricalcolato.

  L'elenco è compresso per impostazione predefinita, quindi puoi espanderlo per mostrare i valori hash. Facoltativamente, puoi provare tu stesso i calcoli hash seguendo i passaggi descritti in. Utilizzo di una bozza per ricalcolare il riassunto
- Digest calcolato: l'hash risultante dalla serie di calcoli Hash eseguiti sull'hash di revisione. Se questo valore corrisponde al Digest salvato in precedenza, la verifica ha esito positivo.

La scheda Blocca mostra il contenuto del blocco che contiene la revisione che stai verificando. Include i seguenti dettagli:

- ID transazione: l'ID univoco della transazione che ha commesso questo blocco.
- Ora della transazione: il timestamp in cui questo blocco è stato salvato nel filamento.
- Block hash: il valore SHA-256 che rappresenta in modo univoco questo blocco e tutto il suo contenuto.
- Indirizzo del blocco: la posizione nel diario del registro in cui è stato salvato questo blocco. Un indirizzo ha i due campi seguenti:

  - Strand ID: l'ID univoco del filone del diario che contiene questo blocco.
  - Numero di sequenza: il numero indice che specifica la posizione di questo blocco all'interno del trefolo.
- Dichiarazioni — Le istruzioni PartiQL che sono state eseguite per eseguire il commit delle voci in questo blocco.

> **Note**
>
> Se esegui istruzioni con parametri a livello di codice, queste vengono registrate nei blocchi di journal con parametri bind anziché dati letterali. Ad esempio, potreste vedere la seguente istruzione in un blocco di diario, in cui il punto interrogativo (?) è un segnaposto variabile per il contenuto del documento.
>
> ```
> INSERT INTO Vehicle ?
> ```

- Voci di documenti: le revisioni del documento che sono state salvate in questo blocco.

Se la richiesta non è riuscita a verificare la revisione del documento, consulta Errori comuni di verifica per informazioni sulle possibili cause.

Utilizzo di una bozza per ricalcolare il riassunto

Dopo che QLDB avrà restituito una prova per la tua richiesta di verifica del documento, puoi provare a eseguire tu stesso i calcoli hash. Questa sezione descrive i passaggi principali per ricalcolare il digest utilizzando la dimostrazione fornita.

Innanzitutto, associa il tuo hash Revision al primo hash dell'elenco degli hash Proof. Quindi, procedi nel seguente modo.

1. Ordina i due hash. Confronta gli hash in base ai valori dei byte firmati in ordine little-endian.
2. Concatena i due hash in ordine ordinato.
3. Effettua l'hash della coppia concatenata con un generatore di hash SHA-256.
4. Associa il tuo nuovo hash all'hash successivo nella bozza e ripeti i passaggi da 1 a 3. Dopo aver elaborato l'ultimo proof hash, il nuovo hash è il riassunto ricalcolato.

Se il digest ricalcolato corrisponde al digest salvato in precedenza, il documento viene verificato con successo.

Per un step-by-step tutorial con esempi di codice che illustrano questi passaggi di verifica, procedi a. Tutorial: verifica dei dati utilizzando un SDK AWS

Tutorial: verifica dei dati utilizzando un SDK AWS

In questo tutorial, verifichi un hash di revisione del documento e un hash del blocco journal in un registro Amazon QLDB utilizzando l'API QLDB tramite un SDK. AWS È inoltre possibile utilizzare il driver QLDB per interrogare la revisione del documento.

Si consideri un esempio in cui si dispone di una revisione del documento che contiene i dati di un veicolo con un numero di identificazione del veicolo (VIN) di. `KM8SRDHF6EU074761` La revisione del documento si trova in una `VehicleRegistration` tabella che si trova in un registro denominato. `vehicle-registration` Supponiamo di voler verificare l'integrità sia della revisione del documento per questo veicolo sia del blocco del giornale che contiene la revisione.

> **Note**
>
> Per un post dettagliato AWS sul blog che illustra il valore della verifica crittografica nel contesto di un caso d'uso realistico, consulta Verifica crittografica nel mondo reale con Amazon QLDB.

Argomenti

Prerequisiti

Prima di iniziare, assicurati di fare quanto segue:

1. Configura il driver QLDB per una lingua a tua scelta completando i rispettivi prerequisiti sotto. Nozioni base sul driver Amazon QLDB Ciò include la registrazione AWS, la concessione dell'accesso programmatico per lo sviluppo e la configurazione dell'ambiente di sviluppo.
2. Segui i passaggi da 1 Nozioni di base sulla console Amazon QLDB a 2 per creare un registro denominato `vehicle-registration` e caricarlo con dati di esempio predefiniti.

Successivamente, esamina i passaggi seguenti per scoprire come funziona la verifica, quindi esegui l'esempio di codice completo dall'inizio alla fine.

Passaggio 1: richiedi un riassunto

Prima di poter verificare i dati, devi prima richiedere un riepilogo dal registro `vehicle-registration` per utilizzarlo in un secondo momento.

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')
```

Passaggio 2: interrogare la revisione del documento

Usa il driver QLDB per interrogare gli indirizzi di blocco, gli hash e gli ID dei documenti associati al 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)
```

Fase 3: Richiedere una bozza della revisione

Esegui un'iterazione dei risultati della query e utilizza ogni indirizzo di blocco e ID del documento insieme al nome del libro mastro per inviare una richiesta. `GetRevision` Per ottenere una bozza della revisione, devi anche fornire l'indirizzo del suggerimento contenuto nel digest salvato in precedenza. Questa operazione API restituisce un oggetto che include la revisione del documento e la bozza della revisione.

Per informazioni sulla struttura di revisione e sul relativo contenuto, vedere. Interrogazione dei metadati dei documenti

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)
```

Recuperate quindi la bozza della revisione richiesta.

L'API QLDB restituisce la dimostrazione come rappresentazione in formato stringa dell'elenco ordinato degli hash dei nodi. Per convertire questa stringa in un elenco della rappresentazione binaria degli hash del nodo, puoi utilizzare un lettore di ioni dalla libreria Amazon Ion. Per ulteriori informazioni sull'uso della libreria Ion, consulta Amazon Ion Cookbook.

Java

In questo esempio, si utilizza `IonReader` per eseguire la conversione binaria.

```
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

In questo esempio, si utilizza `IonLoader` per caricare la bozza in un datagramma ionico.

```
string proofText = revisionResponse.Proof.IonText;
IIonDatagram proofValue = IonLoader.Default.Load(proofText);
```

Go

In questo esempio, si utilizza un lettore Ion per convertire la dimostrazione in binario e per scorrere l'elenco degli hash dei nodi della bozza.

```
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

In questo esempio, si utilizza la `load` funzione per eseguire la conversione binaria.

```
let proofValue: dom.Value = load(revisionResponse.Proof.IonText);
```

Python

In questo esempio, si utilizza la `loads` funzione per eseguire la conversione binaria.

```
proof_text = proof_response.get('Proof').get('IonText')
proof_hashes = loads(proof_text)
```

Fase 4: Ricalcola il digest dalla revisione

Usa l'elenco di hash della bozza per ricalcolare il digest, iniziando dall'hash della revisione. Finché il digest salvato in precedenza è noto e affidabile al di fuori di QLDB, l'integrità della revisione del documento è dimostrata se l'hash del digest ricalcolato corrisponde all'hash del digest salvato.

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))
```

Fase 5: Richiedere una bozza per il blocco journal

Successivamente, si verifica il blocco del diario che contiene la revisione del documento.

Utilizza l'indirizzo di blocco e l'indirizzo tip del digest che hai salvato nel passaggio 1 per inviare una `GetBlock` richiesta. Analogamente alla `GetRevision` richiesta nella Fase 2, devi fornire nuovamente l'indirizzo del suggerimento contenuto nel digest salvato per ottenere una prova dell'esistenza del blocco. Questa operazione API restituisce un oggetto che include il blocco e la bozza del blocco.

Per informazioni sulla struttura del blocco journal e sul relativo contenuto, vedereContenuto del diario in 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)
```

Quindi, recupera l'hash del blocco e la dimostrazione dal risultato.

Java

In questo esempio, si utilizza `IonLoader` per caricare l'oggetto blocco in un `IonDatagram` contenitore.

```
String blockText = getBlockResult.getBlock().getIonText();

IonDatagram datagram = SYSTEM.getLoader().load(blockText);
ionStruct = (IonStruct)datagram.get(0);

final byte[] blockHash = ((IonBlob)ionStruct.get("blockHash")).getBytes();
```

Utilizzate anche `IonLoader` per caricare la bozza in un file`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

In questo esempio, si utilizza `IonLoader` per caricare il blocco e la bozza in un datagramma ionico per ciascuno.

```
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

In questo esempio, utilizzate un lettore Ion per convertire la bozza in binario e per scorrere l'elenco degli hash dei nodi della bozza.

```
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

In questo esempio, si utilizza la `load` funzione per convertire il blocco e la dimostrazione in binario.

```
const blockValue: dom.Value = load(getBlockResponse.Block.IonText)
let blockHash: Uint8Array = blockValue.get("blockHash").uInt8ArrayValue();

proofValue = load(getBlockResponse.Proof.IonText);
```

Python

In questo esempio, si utilizza la `loads` funzione per convertire il blocco e la dimostrazione in binario.

```
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)
```

Fase 6: Ricalcola il digest dal blocco

Usa l'elenco di hash del proof per ricalcolare il digest, iniziando dall'hash del blocco. Finché il digest salvato in precedenza è noto e affidabile al di fuori di QLDB, l'integrità del blocco viene dimostrata se l'hash del digest ricalcolato corrisponde all'hash del digest salvato.

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))
```

Gli esempi di codice precedenti utilizzano la seguente funzione per ricalcolare il digest. `dot` Questa funzione accetta l'input di due hash, li ordina, li concatena e quindi restituisce l'hash dell'array concatenato.

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
```

Esegui l'esempio completo di codice

Eseguite l'esempio di codice completo come segue per eseguire tutti i passaggi precedenti dall'inizio alla fine.

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))
```

Errori comuni di verifica

Questa sezione descrive gli errori di runtime generati da Amazon QLDB per le richieste di verifica.

Di seguito è riportato un elenco di eccezioni comuni restituite dal servizio. Ogni eccezione include il messaggio di errore specifico, seguito dalle operazioni API che possono generarlo, una breve descrizione e suggerimenti per possibili soluzioni.

IllegalArgumentException

Messaggio: il valore Ion fornito non è valido e non può essere analizzato.

Operazioni API: `GetDigest, GetBlock, GetRevision`

Assicurati di fornire un valore Amazon Ion valido prima di ritentare la richiesta.

IllegalArgumentException

Messaggio: l'indirizzo di blocco fornito non è valido.

Operazioni API: `GetDigest, GetBlock, GetRevision`

Assicurati di fornire un indirizzo di blocco valido prima di riprovare la richiesta. Un indirizzo di blocco è una struttura Amazon Ion con due campi: `strandId` e `sequenceNo`.

Ad esempio: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`

IllegalArgumentException

Messaggio: il numero progressivo dell'indirizzo digest tip fornito non corrisponde all'ultimo record registrato del filone.

Operazioni API: `GetDigest, GetBlock, GetRevision`

L'indirizzo digest tip fornito deve avere un numero di sequenza inferiore o uguale al numero di sequenza dell'ultimo record registrato della sezione del diario. Prima di riprovare la richiesta, assicurati di fornire un indirizzo digest tip con un numero di sequenza valido.

IllegalArgumentException

Messaggio: lo Strand ID dell'indirizzo di blocco fornito non è valido.

Operazioni API: `GetDigest, GetBlock, GetRevision`

L'indirizzo di blocco fornito deve avere un ID di filamento che corrisponda all'ID di strand del diario. Prima di riprovare la richiesta, assicurati di fornire un indirizzo di blocco con un Strand ID valido.

IllegalArgumentException

Messaggio: il numero di sequenza dell'indirizzo di blocco fornito non corrisponde all'ultimo record registrato del filamento.

Operazioni API: `GetBlock, GetRevision`

L'indirizzo di blocco fornito deve avere un numero di sequenza inferiore o uguale al numero di sequenza dell'ultimo record confermato del filamento. Prima di riprovare la richiesta, assicurati di fornire un indirizzo di blocco con un numero di sequenza valido.

IllegalArgumentException

Messaggio: lo Strand ID dell'indirizzo di blocco fornito deve corrispondere allo Strand ID dell'indirizzo digest tip fornito.

Operazioni API: `GetBlock, GetRevision`

È possibile verificare la revisione o il blocco di un documento solo se presente nella stessa sezione del diario del digest fornito.

IllegalArgumentException

Messaggio: il numero di sequenza dell'indirizzo di blocco fornito non deve essere maggiore del numero di sequenza dell'indirizzo digest tip fornito.

Operazioni API: `GetBlock, GetRevision`

Puoi verificare la revisione o il blocco di un documento solo se è incluso nel digest che fornisci. Ciò significa che è stato inserito nella rivista prima dell'indirizzo del riepilogo.

IllegalArgumentException

Messaggio: l'ID del documento fornito non è stato trovato nel blocco all'indirizzo di blocco specificato.

Funzionamento dell'API: `GetRevision`

L'ID del documento fornito deve esistere nell'indirizzo di blocco fornito. Prima di riprovare la richiesta, assicurati che questi due parametri siano coerenti.

Esportazione dei dati del diario da Amazon QLDB

Amazon QLDB utilizza un log transazionale immutabile, noto come journal, per l'archiviazione dei dati. Il diario tiene traccia di ogni modifica ai dati impegnati e mantiene una cronologia completa e verificabile delle modifiche nel tempo.

È possibile accedere ai contenuti del diario nel registro per vari scopi, tra cui analisi, controllo, conservazione dei dati, verifica ed esportazione verso altri sistemi. I seguenti argomenti descrivono come esportare blocchi di journal dal tuo registro in un bucket Amazon Simple Storage Service (Amazon S3) del tuo. Account AWS Un processo di esportazione del diario scrive i dati in Amazon S3 come oggetti nella rappresentazione testuale o binaria del formato Amazon Ion o nel formato di testo JSON Lines.

Nel formato JSON Lines, ogni blocco di un oggetto dati esportato è un oggetto JSON valido delimitato da una nuova riga. Puoi utilizzare questo formato per integrare direttamente le esportazioni JSON con strumenti di analisi come Amazon Athena AWS Glue e perché questi servizi possono analizzare automaticamente JSON delimitato da nuove righe. Per ulteriori informazioni sul formato, consulta JSON Lines.

Per informazioni su Amazon S3, consulta la Guida per l'utente di Amazon Simple Storage Service.

> **Note**
>
> Se specifichi JSON come formato di output del processo di esportazione, QLDB converte i dati del journal Ion in JSON negli oggetti dati esportati. Per ulteriori informazioni, consulta Conversione verso il basso in JSON.

Argomenti

- Richiedere l'esportazione di un diario in QLDB
- Uscita di esportazione del diario in QLDB
- Autorizzazioni di esportazione del diario in QLDB
- Errori comuni relativi all'esportazione delle riviste

Richiedere l'esportazione di un diario in QLDB

Amazon QLDB fornisce un'API per richiedere l'esportazione dei blocchi di journal per un intervallo di data e ora specificato e una destinazione di bucket Amazon S3 specificata. Un processo di esportazione del diario può scrivere gli oggetti dati nella rappresentazione testuale o binaria del formato Amazon Ion o nel formato di testo JSON Lines. Puoi utilizzare il AWS Management Console, un AWS SDK o il AWS Command Line Interface (AWS CLI) per creare un processo di esportazione.

Argomenti

AWS Management Console

Segui questi passaggi per inviare una richiesta di esportazione del journal in QLDB utilizzando la console QLDB.

Per richiedere un'esportazione (console)

1. Accedi a e apri AWS Management Console la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel pannello di navigazione, scegli Esporta.
3. Scegli Crea processo di esportazione.
4. Nella pagina Crea processo di esportazione, inserisci le seguenti impostazioni di esportazione:
   - Libro contabile: il libro contabile di cui si desidera esportare i blocchi del diario.
   - Data e ora di inizio: il timestamp di inizio incluso, in formato UTC (Coordinated Universal Time), dell'intervallo di blocchi di diario da esportare. Questo timestamp deve essere anteriore alla data e all'ora di fine. Se fornisci un timestamp di inizio precedente a quello del libro mastro, `CreationDateTime` QLDB lo imposta come predefinito su quello del libro mastro. `CreationDateTime`
   - Data e ora di fine: l'esclusivo timestamp di fine (UTC) dell'intervallo di blocchi di journal da esportare. Questa data e ora non possono appartenere al futuro.
   - Destinazione per i blocchi journal: il bucket Amazon S3 e il nome del prefisso in cui il processo di esportazione scrive gli oggetti dati. Utilizza il seguente formato URI Amazon S3.

```
s3://DOC-EXAMPLE-BUCKET/prefix/
```

È necessario specificare un nome di bucket S3 e un nome di prefisso opzionale per gli oggetti di output. Di seguito è riportato un esempio.

```
s3://DOC-EXAMPLE-BUCKET/journalExport/
```

Il nome e il prefisso del bucket devono entrambi rispettare le regole e le convenzioni di denominazione di Amazon S3. Per ulteriori informazioni sulla denominazione dei bucket, consulta Restrizioni e limitazioni dei bucket nella Amazon S3 Developer Guide. Per ulteriori informazioni sui prefissi dei nomi chiave, consulta Chiave dell'oggetto e metadati.

> **Note**
>
> Le esportazioni tra regioni non sono supportate. Il bucket Amazon S3 specificato deve trovarsi nello Regione AWS stesso registro.

- Crittografia S3: le impostazioni di crittografia utilizzate dal processo di esportazione per scrivere dati in un bucket Amazon S3. Per ulteriori informazioni sulle opzioni di crittografia lato server in Amazon S3, consulta Protezione dei dati utilizzando la crittografia lato server nella Amazon S3 Developer Guide.
  - Crittografia predefinita del bucket: utilizza le impostazioni di crittografia predefinite del bucket Amazon S3 specificato.
  - AES-256: utilizza la crittografia lato server con le chiavi gestite di Amazon S3 (SSE-S3).
  - AWS-KMS: utilizza la crittografia lato server con chiavi AWS KMS gestite (SSE-KMS).

    Se scegli questo tipo insieme all' AWS KMS key opzione Scegli un altro, devi anche specificare una chiave KMS di crittografia simmetrica nel seguente formato Amazon Resource Name (ARN).

    ```
    arn:aws:kms:aws-region:account-id:key/key-id
    ```
- Accesso al servizio: il ruolo IAM che concede le autorizzazioni di scrittura QLDB nel bucket Amazon S3. Se applicabile, il ruolo IAM deve inoltre concedere le autorizzazioni QLDB per utilizzare la chiave KMS.

Per passare un ruolo a QLDB quando si richiede l'esportazione di un journal, è necessario disporre delle autorizzazioni per eseguire *`iam:PassRole`* l'azione sulla risorsa del ruolo IAM.

- Crea e usa un nuovo ruolo di servizio: consenti alla console di creare un nuovo ruolo per te con le autorizzazioni richieste per il bucket Amazon S3 specificato.
- Usa un ruolo di servizio esistente: per scoprire come creare manualmente questo ruolo in IAM, consulta. Autorizzazioni di esportazione

- Formato di output: il formato di output dei dati del diario esportati
  - Ion text — (impostazione predefinita) Rappresentazione testuale di Amazon Ion
  - Ion binary: rappresentazione binaria di Amazon Ion
  - JSON: formato di testo JSON delimitato da newline

    Se scegli JSON, QLDB converte i dati del journal Ion in JSON negli oggetti dati esportati. Per ulteriori informazioni, consulta Conversione verso il basso in JSON.

5. Quando le impostazioni sono quelle che desideri, scegli Crea processo di esportazione.

   Il tempo necessario per completare il processo di esportazione varia a seconda della dimensione dei dati. Se l'invio della richiesta ha esito positivo, la console torna alla pagina principale di esportazione ed elenca i lavori di esportazione con il loro stato attuale.

6. Puoi vedere i tuoi oggetti di esportazione sulla console Amazon S3.

   Apri la console Amazon S3 all'indirizzo https://console.aws.amazon.com/s3/.

   Per ulteriori informazioni sul formato di questi oggetti di output, consultaUscita di esportazione del diario in QLDB.

> **Note**
>
> I lavori di esportazione scadono sette giorni dopo il completamento. Per ulteriori informazioni, consulta Scadenza del lavoro di esportazione.

API QLDB

Puoi anche richiedere l'esportazione di un journal utilizzando l'API Amazon QLDB con AWS un SDK o il. AWS CLI L'API QLDB fornisce le seguenti operazioni per l'uso da parte dei programmi applicativi:

- `ExportJournalToS3`— Esporta i contenuti del diario entro un intervallo di data e ora da un determinato registro in un bucket Amazon S3 specificato. Un processo di esportazione può scrivere i dati come oggetti nella rappresentazione testuale o binaria del formato Amazon Ion o nel formato di testo JSON Lines.
- `DescribeJournalS3Export`— Restituisce informazioni dettagliate su un processo di esportazione di un diario. L'output include lo stato corrente, l'ora di creazione e i parametri della richiesta di esportazione originale.
- `ListJournalS3Exports`— Restituisce un elenco di descrizioni dei lavori di esportazione delle scritture contabili per tutti i libri contabili associati alla regione corrente Account AWS e alla regione. L'output di ogni descrizione del lavoro di esportazione include gli stessi dettagli restituiti da`DescribeJournalS3Export`.
- `ListJournalS3ExportsForLedger`— Restituisce un elenco di descrizioni dei lavori di esportazione delle riviste per un determinato libro mastro. L'output di ogni descrizione del lavoro di esportazione include gli stessi dettagli restituiti da`DescribeJournalS3Export`.

Per una descrizione completa di queste operazioni API, consultaDocumentazione di riferimento dell'API Amazon QLDB.

Per informazioni sull'esportazione dei dati del diario utilizzando AWS CLI, vedere il AWS CLI Command Reference.

Applicazione di esempio (Java)

Per esempi di operazioni di esportazione di base in codice Java, consulta il GitHub repository amazon-qldb-dmv-sampleaws-samples/ -java. Per istruzioni su come scaricare e installare questa applicazione di esempio, consulta. Installazione dell'applicazione di esempio Java Amazon QLDB Prima di richiedere un'esportazione, assicurati di seguire i passaggi da 1 a 3 Tutorial su Java per creare un registro di esempio e caricarlo con dati di esempio.

Il codice tutorial nelle seguenti classi fornisce esempi di creazione di un'esportazione, controllo dello stato di un'esportazione ed elaborazione dell'output di un'esportazione.

Classe	Descrizione
ExportJournal	Esporta i blocchi del diario dal registro di `vehicle-registration` esempio per un intervallo di timestamp compreso tra 10 minuti

Classe	Descrizione
	e ora. Scrive gli oggetti di output in un bucket S3 specificato o crea un bucket unico se non ne viene fornito uno.
DescribeJournalExport	Descrive un processo di esportazione del giornale per un registro specificato `exportId` nel registro di esempio. `vehicle-registration`
ListJournalExports	Restituisce un elenco di descrizioni dei lavori di esportazione delle riviste per il libro mastro `vehicle-registration` di esempio.
ValidateQldbHashChain	Convalida la catena hash del registro di `vehicle-registration` esempio utilizzando un determinato registro. `exportId` Se non viene fornita, richiede una nuova esportazione da utilizzare per la convalida della catena hash.

Scadenza del lavoro di esportazione

I lavori di esportazione del diario completati sono soggetti a un periodo di conservazione di 7 giorni. Vengono eliminati automaticamente dopo la scadenza di questo limite. Questo periodo di scadenza è un limite rigido e non può essere modificato.

Dopo l'eliminazione di un processo di esportazione completato, non è più possibile utilizzare la console QLDB o le seguenti operazioni API per recuperare i metadati relativi al lavoro:

- `DescribeJournalS3Export`
- `ListJournalS3Exports`
- `ListJournalS3ExportsForLedger`

Tuttavia, questa scadenza non ha alcun impatto sui dati esportati stessi. Tutti i metadati vengono conservati nei file manifesto scritti dalle tue esportazioni. Questa scadenza è progettata per fornire un'esperienza più fluida per le operazioni API che elencano i lavori di esportazione delle riviste.

QLDB rimuove i vecchi lavori di esportazione per garantire che vengano visualizzate solo le esportazioni recenti senza dover analizzare più pagine di lavori.

Uscita di esportazione del diario in QLDB

Un processo di esportazione del journal Amazon QLDB scrive due file manifest oltre agli oggetti dati che contengono i blocchi del journal. Questi vengono tutti salvati nel bucket Amazon S3 che hai fornito nella richiesta di esportazione. Nelle sezioni seguenti sono descritti il formato e il contenuto di ciascun oggetto di output.

> **Note**
>
> Se specifichi JSON come formato di output del processo di esportazione, QLDB converte i dati del journal Amazon Ion in JSON negli oggetti dati esportati. Per Conversione verso il basso in JSON ulteriori informazioni, procedi a.

Argomenti

File manifesto

Amazon QLDB crea due file manifest nel bucket S3 fornito per ogni richiesta di esportazione. Il file manifesto iniziale viene creato non appena si invia la richiesta di esportazione. Il file manifesto finale viene scritto dopo il completamento dell'esportazione. Puoi utilizzare questi file per verificare lo stato dei tuoi lavori di esportazione in Amazon S3.

Il formato per il contenuto dei file manifest corrisponde al formato di output richiesto per l'esportazione.

Manifesto iniziale

Il manifesto iniziale indica che il processo di esportazione è iniziato. Contiene i parametri di input che hai passato alla richiesta. Oltre alla destinazione di Amazon S3 e ai parametri dell'ora di inizio e fine

per l'esportazione, questo file contiene anche un. `exportId` `exportId`È un ID univoco che QLDB assegna a ogni processo di esportazione.

La convenzione di denominazione dei file è la seguente.

```
s3://DOC-EXAMPLE-BUCKET/prefix/exportId.started.manifest
```

Di seguito è riportato un esempio di file manifest iniziale e del relativo contenuto in formato di testo 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"
}
```

Il manifesto iniziale include `outputFormat` solo se è stato specificato nella richiesta di esportazione. Se non si specifica il formato di output, per impostazione predefinita viene utilizzato il formato dei dati esportati. `ION_TEXT`

Il funzionamento dell'API DescribeJournalS3Export e il tipo di contenuto degli oggetti Amazon S3 esportati indicano anche il formato di output.

Manifesto finale

Il manifesto finale indica che il processo di esportazione per una particolare sezione del diario è stato completato. Il processo di esportazione scrive un file manifesto finale separato per ogni filone.

> **Note**
>
> In Amazon QLDB, un filone è una partizione del diario del libro contabile. QLDB attualmente supporta solo riviste con un solo filone.

Il manifesto finale include un elenco ordinato di chiavi degli oggetti dati che sono state scritte durante l'esportazione. La convenzione di denominazione dei file è la seguente.

```
s3://DOC-EXAMPLE-BUCKET/prefix/exportId.strandId.completed.manifest
```

`strandId`È un ID univoco che QLDB assegna al filamento. Di seguito è riportato un esempio di file manifest finale e del relativo contenuto in formato di testo 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"
  ]
}
```

Oggetti dati

Amazon QLDB scrive oggetti di dati del journal nel bucket Amazon S3 fornito nella rappresentazione testuale o binaria del formato Amazon Ion o nel formato di testo JSON Lines.

Nel formato JSON Lines, ogni blocco in un oggetto dati esportato è un oggetto JSON valido delimitato da una nuova riga. Puoi utilizzare questo formato per integrare direttamente le esportazioni JSON con strumenti di analisi come Amazon Athena AWS Glue e perché questi servizi possono analizzare automaticamente JSON delimitato da nuove righe. Per ulteriori informazioni sul formato, consulta JSON Lines.

Nomi degli oggetti dati

Un processo di esportazione di un diario scrive questi oggetti dati con la seguente convenzione di denominazione.

```
s3://DOC-EXAMPLE-BUCKET/prefix/yyyy/mm/dd/hh/strandId.startSn-endSn.ion|.json
```

- I dati di output di ogni processo di esportazione sono suddivisi in blocchi.

- `yyyy/mm/dd/hh`— La data e l'ora in cui hai inviato la richiesta di esportazione. Gli oggetti che vengono esportati entro la stessa ora vengono raggruppati con lo stesso prefisso Amazon S3.
- `strandId`— L'ID univoco del particolare filamento che contiene il blocco journal che viene esportato.
- `startSn-endSn`— L'intervallo di numeri di sequenza incluso nell'oggetto. Un numero di sequenza specifica la posizione di un blocco all'interno di un filamento.

Si supponga, ad esempio, di specificare il percorso seguente.

```
s3://DOC-EXAMPLE-BUCKET/journalExport/
```

Il processo di esportazione crea un oggetto dati Amazon S3 simile al seguente. Questo esempio mostra il nome di un oggetto in formato Ion.

```
s3://DOC-EXAMPLE-BUCKET/journalExport/2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.1-5.ion
```

Contenuto dell'oggetto dati

Ogni oggetto dati contiene oggetti Journal Block con il seguente formato.

```
{
  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 blocco è un oggetto che viene salvato nel journal durante una transazione. Un blocco contiene i metadati delle transazioni insieme alle voci che rappresentano le revisioni dei documenti che sono state eseguite nella transazione e le istruzioni PartiQL che le hanno salvate.

Di seguito è riportato un esempio di blocco con dati di esempio in formato testo Ion. Per informazioni sui campi in un oggetto blocco, vedereContenuto del diario in Amazon QLDB.

> **Note**
>
> Questo esempio di blocco viene fornito solo a scopo informativo. Gli hash mostrati non sono valori hash calcolati reali.

```
{
  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"
      }
    }
  ]
}
```

Nel `revisions` campo, alcuni oggetti di revisione potrebbero contenere solo un `hash` valore e nessun altro attributo. Si tratta di revisioni di sistema solo interne che non contengono dati utente. Un processo di esportazione include queste revisioni nei rispettivi blocchi perché gli hash di queste revisioni fanno parte dell'intera catena hash della rivista. La catena hash completa è necessaria per la verifica crittografica.

Conversione verso il basso in JSON

Se specifichi JSON come formato di output del processo di esportazione, QLDB converte i dati del journal Amazon Ion in JSON negli oggetti dati esportati. Tuttavia, la conversione di Ion in JSON comporta perdite in alcuni casi in cui i dati utilizzano tipi di ioni complessi che non esistono in JSON.

Per informazioni dettagliate sulle regole di conversione da Ion a JSON, consulta Down-converting to JSON nell'Amazon Ion Cookbook.

Libreria di processori di esportazione (Java)

QLDB fornisce un framework estensibile per Java che semplifica l'elaborazione delle esportazioni in Amazon S3. Questa libreria di framework gestisce il lavoro di lettura dell'output di un'esportazione e l'iterazione tra i blocchi esportati in ordine sequenziale. Per utilizzare questo processore di esportazione, consulta il GitHub repository awslabs/ -java. amazon-qldb-export-processor

Autorizzazioni di esportazione del diario in QLDB

Prima di inviare una richiesta di esportazione del journal in Amazon QLDB, devi fornire a QLDB le autorizzazioni di scrittura nel bucket Amazon S3 specificato. Se scegli un client managed AWS KMS

key come tipo di crittografia degli oggetti per il tuo bucket Amazon S3, devi anche fornire a QLDB le autorizzazioni per utilizzare la chiave di crittografia simmetrica specificata. Amazon S3 non supporta chiavi KMS asimmetriche.

Per fornire al processo di esportazione le autorizzazioni necessarie, puoi fare in modo che QLDB assuma un ruolo di servizio IAM con le politiche di autorizzazione appropriate. Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.

> **Note**
>
> Per passare un ruolo a QLDB quando si richiede l'esportazione di un journal, è necessario disporre delle autorizzazioni per eseguire `iam:PassRole` l'azione sulla risorsa del ruolo IAM. Ciò si aggiunge all'`qldb:ExportJournalToS3`autorizzazione sulla risorsa di registro QLDB.
> Per informazioni su come controllare l'accesso a QLDB utilizzando IAM, consulta. Come funziona Amazon QLDB con IAM Per un esempio di policy QLDB, vedere. Esempi di policy basate sull'identità per Amazon QLDB

In questo esempio, crei un ruolo che consente a QLDB di scrivere oggetti in un bucket Amazon S3 per tuo conto. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.

Se stai esportando un journal QLDB per Account AWS la prima volta, devi prima creare un ruolo IAM con le politiche appropriate procedendo come segue. In alternativa, puoi utilizzare la console QLDB per creare automaticamente il ruolo per te. Altrimenti, puoi scegliere un ruolo che hai creato in precedenza.

Argomenti

- Creazione di una policy di autorizzazione
- Creazione di un ruolo IAM

Creazione di una policy di autorizzazione

Completa i seguenti passaggi per creare una politica di autorizzazioni per un processo di esportazione di diari QLDB. Questo esempio mostra una policy sui bucket di Amazon S3 che concede le autorizzazioni QLDB per scrivere oggetti nel bucket specificato. Se applicabile, l'esempio mostra anche una politica chiave che consente a QLDB di utilizzare la chiave KMS di crittografia simmetrica.

Per ulteriori informazioni sulle policy dei bucket di Amazon S3, consulta Using bucket policy e user policy nella Amazon Simple Storage Service User Guide. Per ulteriori informazioni sulle politiche AWS KMS chiave, consulta Using key policy AWS KMS nella Developer Guide.AWS Key Management Service

> **Note**
>
> Il bucket Amazon S3 e la chiave KMS devono trovarsi entrambi nello stesso registro QLDB Regione AWS .

Come utilizzare l'editor di policy JSON per creare una policy

1. Accedi AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione a sinistra, seleziona Policies (Policy).

   Se è la prima volta che selezioni Policy, verrà visualizzata la pagina Benvenuto nelle policy gestite. Seleziona Inizia.
3. Nella parte superiore della pagina, scegli Crea policy.
4. Scegli la scheda JSON.
5. Specificare un documento della policy JSON.
   - Se utilizzi una chiave KMS gestita dal cliente per la crittografia degli oggetti Amazon S3, utilizza il seguente documento di policy di esempio. *Per utilizzare questa politica, sostituisci DOC-EXAMPLE-BUCKET, us-east-1, 123456789012 e 1234abcd-12ab-34cd-56ef-1234567890ab nell'esempio con le tue informazioni.*

     ```
     {
     ```

```
    "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"
        }
    ]
}
```

- Per altri tipi di crittografia, utilizzare il seguente documento di policy di esempio. Per utilizzare questa politica, sostituisci *DOC-EXAMPLE-BUCKET nell'esempio con il tuo nome di bucket* Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permission",
            "Action": [
                "s3:PutObjectAcl",
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

6. Scegli Verifica policy.

> **Note**
>
> È possibile passare tra le schede Visual editor (Editor visivo) e JSON in qualsiasi momento. Se tuttavia si apportano modifiche o se si seleziona Review policy (Rivedi policy) nella scheda Visual editor (Editor visivo), IAM potrebbe modificare la policy per ottimizzarla per l'editor visivo. Per ulteriori informazioni, consulta Modifica della struttura delle policy nella Guida per l'utente di IAM.

7. Nella pagina Review policy (Rivedi policy), inserisci i valori per Name (Nome) e Description (Descrizione) (facoltativa) per la policy che stai creando. Consulta il Summary (Riepilogo) della policy per visualizzare le autorizzazioni concesse dalla policy. Seleziona Create policy (Crea policy) per salvare il proprio lavoro.

Creazione di un ruolo IAM

Dopo aver creato una politica di autorizzazioni per il tuo processo di esportazione di riviste QLDB, puoi quindi creare un ruolo IAM e allegare la tua policy ad esso.

Per creare il ruolo di servizio per QLDB (console IAM)

1. Accedi AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione della console IAM, scegliere Ruoli e quindi Crea ruolo.
3. Per Trusted entity type (Tipo di entità attendibile), scegli Servizio AWS.
4. Per Servizio o caso d'uso, scegli QLDB, quindi scegli lo use case QLDB.
5. Seleziona Successivo.
6. Seleziona la casella accanto alla politica che hai creato nei passaggi precedenti.
7. (Facoltativo) Impostare un limite delle autorizzazioni. Questa è una caratteristica avanzata disponibile per i ruoli di servizio, ma non per i ruoli collegati ai servizi.

    a. Apri la sezione Imposta i limiti delle autorizzazioni, quindi scegli Usa un limite di autorizzazioni per controllare il numero massimo di autorizzazioni per i ruoli.

    IAM include un elenco delle politiche AWS gestite e gestite dal cliente nel tuo account.

    b. Selezionare la policy da utilizzare per il limite delle autorizzazioni.

8. Seleziona Successivo.
9. Inserisci il nome del ruolo o il suffisso del nome del ruolo per aiutarti a identificare lo scopo del ruolo.

> **Important**
>
> Quando assegnate un nome a un ruolo, tenete presente quanto segue:
>
> - I nomi dei ruoli devono essere univoci all'interno del tuo Account AWS account e non possono essere resi unici per caso.
>
> Ad esempio, non creare ruoli denominati entrambi **`PRODROLE`** e**`prodrole`**. Quando un nome di ruolo viene utilizzato in una policy o come parte di un ARN, il nome del ruolo fa distinzione tra maiuscole e minuscole, tuttavia quando un nome di ruolo viene visualizzato dai clienti nella console, ad esempio durante il processo di accesso, il nome del ruolo non fa distinzione tra maiuscole e minuscole.
> - Non è possibile modificare il nome del ruolo dopo averlo creato perché altre entità potrebbero fare riferimento al ruolo.

10. (Facoltativo) In Descrizione, inserisci una descrizione per il ruolo.
11. (Facoltativo) Per modificare i casi d'uso e le autorizzazioni per il ruolo, nelle sezioni Passo 1: Seleziona entità attendibili o Passaggio 2: Aggiungi autorizzazioni, scegli Modifica.
12. (Facoltativo) Per facilitare l'identificazione, l'organizzazione o la ricerca del ruolo, aggiungi tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Applicazione di tag alle risorse IAM nella Guida per l'utente di IAM.
13. Verificare il ruolo e quindi scegliere Create role (Crea ruolo).

Il seguente documento JSON è un esempio di policy di fiducia che consente a QLDB di assumere un ruolo IAM con autorizzazioni specifiche associate.

```
{
    "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**
>
> Questo esempio di politica di fiducia mostra come utilizzare le chiavi di contesto *aws:SourceArn* e di contesto della condizione *aws:SourceAccount* globale per evitare il confuso problema del vice. Con questa politica di fiducia, QLDB può assumere il ruolo di qualsiasi risorsa QLDB solo nell'account. 123456789012
> Per ulteriori informazioni, consulta Prevenzione del confused deputy tra servizi.

Dopo aver creato il tuo ruolo IAM, torna alla console QLDB e aggiorna la pagina Crea processo di esportazione in modo che possa trovare il tuo nuovo ruolo.

Errori comuni relativi all'esportazione delle riviste

Questa sezione descrive gli errori di runtime generati da Amazon QLDB per le richieste di esportazione dei journal.

Di seguito è riportato un elenco di eccezioni comuni restituite dal servizio. Ogni eccezione include il messaggio di errore specifico, seguito da una breve descrizione e suggerimenti per possibili soluzioni.

AccessDeniedException

Messaggio: Utente: userARN non è autorizzato a eseguire: iam: PassRole on resource: roLearn

Non disponi delle autorizzazioni per passare un ruolo IAM al servizio QLDB. QLDB richiede un ruolo per tutte le richieste di esportazione del journal e devi disporre delle autorizzazioni per passare questo ruolo a QLDB. Il ruolo fornisce a QLDB le autorizzazioni di scrittura nel bucket Amazon S3 specificato.

Verifica di definire una policy IAM che conceda l'autorizzazione a eseguire l'operazione `PassRole` API sulla risorsa del ruolo IAM specificata per il servizio QLDB (). `qldb.amazonaws.com` Per un esempio di policy, consulta Esempi di policy basate sull'identità per Amazon QLDB.

IllegalArgumentException

`Messaggio: QLDB ha riscontrato un errore durante la convalida della configurazione S3: errorCode ErrorMessage`

Una possibile causa di questo errore è che il bucket Amazon S3 fornito non esiste in Amazon S3. Oppure, QLDB non dispone di autorizzazioni sufficienti per scrivere oggetti nel bucket Amazon S3 specificato.

Verifica che il nome del bucket S3 fornito nella richiesta di lavoro di esportazione sia corretto. Per ulteriori informazioni sulla denominazione dei bucket, consulta Restrizioni e limitazioni dei bucket nella Amazon Simple Storage Service User Guide.

Inoltre, verifica di definire una policy per il bucket specificato che conceda `PutObject` e `PutObjectAcl` autorizzi il servizio QLDB (). `qldb.amazonaws.com` Per ulteriori informazioni, consulta Autorizzazioni di esportazione.

IllegalArgumentException

Messaggio: risposta inaspettata da Amazon S3 durante la convalida della configurazione S3. *`Risposta da S3: ErrorCode ErrorMessage`*

Il tentativo di scrivere i dati di esportazione del journal nel bucket S3 fornito non è riuscito con la risposta di errore di Amazon S3 fornita. Per ulteriori informazioni sulle possibili cause, consulta la sezione Risoluzione dei problemi di Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

IllegalArgumentException

Messaggio: il prefisso del bucket Amazon S3 non deve superare i 128 caratteri

Il prefisso fornito nella richiesta di esportazione del journal contiene più di 128 caratteri.

IllegalArgumentException

Messaggio: la data di inizio non deve essere successiva alla data di fine

Entrambi `InclusiveStartTime` `ExclusiveEndTime` devono essere nel formato di data e ora ISO 8601 e nel formato UTC (Coordinated Universal Time).

IllegalArgumentException

Messaggio: la data di fine non può essere futura

Entrambi `InclusiveStartTime` i formati `ExclusiveEndTime` devono essere in formato `ISO 8601` data e ora e UTC.

IllegalArgumentException

Messaggio: l'impostazione di crittografia degli oggetti (S3EncryptionConfiguration) fornita non è compatibile con una chiave AWS Key Management Service (AWS KMS)

Hai `KMSKeyArn` fornito una `ObjectEncryptionType` delle due opzioni `NO_ENCRYPTION` o`SSE_S3`. È possibile fornire a un cliente gestito AWS KMS key solo un tipo di crittografia degli oggetti di`SSE_KMS`. Per ulteriori informazioni sulle opzioni di crittografia lato server in Amazon S3, consulta Protezione dei dati utilizzando la crittografia lato server nella Amazon S3 Developer Guide.

LimitExceededException

Messaggio: è stato superato il limite di 2 job di esportazione di Journal eseguiti contemporaneamente

QLDB impone un limite predefinito di due processi di esportazione di giornali simultanei.

Streaming dei dati del diario da Amazon QLDB

Amazon QLDB utilizza un log transazionale immutabile, noto come journal, per l'archiviazione dei dati. Il diario tiene traccia di ogni modifica ai dati impegnati e mantiene una cronologia completa e verificabile delle modifiche nel tempo.

Puoi creare uno stream in QLDB che acquisisce ogni revisione del documento inserita nel tuo diario e invia questi dati ad Amazon Kinesis Data Streams quasi in tempo reale. Un flusso QLDB è un flusso continuo di dati dal diario del registro a una risorsa del flusso di dati Kinesis.

Quindi, utilizzi la piattaforma di streaming Kinesis o la Kinesis Client Library per consumare lo streaming, elaborare i record di dati e analizzare il contenuto dei dati. Uno stream QLDB scrive i dati su Kinesis Data Streams in tre tipi di record: controllo, riepilogo dei blocchi e dettagli di revisione. Per ulteriori informazioni, consulta Record di stream QLDB in Kinesis.

Argomenti

Casi di utilizzo comune

Lo streaming ti consente di utilizzare QLDB come un'unica fonte di verità verificabile, integrando al contempo i dati del tuo diario con altri servizi. Di seguito sono riportati alcuni dei casi d'uso più comuni supportati dai flussi di journal QLDB:

- Architettura basata sugli eventi: crea applicazioni in uno stile architettonico basato sugli eventi con componenti disaccoppiati. Ad esempio, una banca può utilizzare AWS Lambda le funzioni per

implementare un sistema di notifica che avvisa i clienti quando il saldo del conto scende al di sotto di una soglia. In un sistema di questo tipo, i saldi dei conti vengono conservati in un registro QLDB e tutte le modifiche al saldo vengono registrate nel giornale di registrazione. La AWS Lambda funzione può attivare la logica di notifica quando si consuma un evento di aggiornamento del saldo che viene salvato nel journal e inviato a un flusso di dati Kinesis.

- Analisi in tempo reale: crea applicazioni Kinesis consumer che eseguono analisi in tempo reale sui dati degli eventi. Con questa funzionalità, puoi ottenere informazioni dettagliate quasi in tempo reale e rispondere rapidamente a un ambiente aziendale in evoluzione. Ad esempio, un sito di e-commerce può analizzare i dati di vendita dei prodotti e interrompere la pubblicità di un prodotto scontato non appena le vendite raggiungono un limite.
- Analisi storica: sfrutta l'architettura orientata al journal di Amazon QLDB riproducendo i dati storici degli eventi. Puoi scegliere di avviare uno stream QLDB in qualsiasi momento del passato, in cui tutte le revisioni successive a quel momento vengono consegnate a Kinesis Data Streams. Utilizzando questa funzionalità, puoi creare applicazioni consumer Kinesis che eseguono processi di analisi su dati storici. Ad esempio, un sito di e-commerce può eseguire analisi in base alle esigenze per generare metriche di vendita passate che non erano state acquisite in precedenza.
- Replica su database creati appositamente: collega i registri QLDB ad altri archivi dati creati appositamente utilizzando flussi di journal QLDB. Ad esempio, utilizza la piattaforma di streaming di dati Kinesis per l'integrazione con Amazon OpenSearch Service, che può fornire funzionalità di ricerca di testo completo per documenti QLDB. Puoi anche creare applicazioni Kinesis consumer personalizzate per replicare i dati del tuo diario su altri database creati appositamente che forniscono diverse viste materializzate. Ad esempio, esegui la replica su Amazon Aurora per dati relazionali o su Amazon Neptune per dati basati su grafici.

Consumo del tuo streaming

Usa Kinesis Data Streams per consumare, elaborare e analizzare continuamente grandi flussi di record di dati. Oltre a Kinesis Data Streams, la piattaforma di streaming dati Kinesis include Amazon Data Firehose e Amazon Managed Service for Apache Flink. Puoi utilizzare questa piattaforma per inviare record di dati direttamente a servizi come Amazon OpenSearch Service, Amazon Redshift, Amazon S3 o Splunk. Per ulteriori informazioni, consulta i consumatori di Kinesis Data Streams nella Amazon Kinesis Data Streams Developer Guide.

Puoi anche utilizzare la Kinesis Client Library (KCL) per creare un'applicazione di streaming consumer per elaborare i record di dati in modo personalizzato. KCL semplifica la codifica fornendo

astrazioni utili sull'API Kinesis Data Streams di basso livello. Per ulteriori informazioni su KCL, consulta Using the Kinesis Client Library nella Amazon Kinesis Data Streams Developer Guide.

Garanzia di consegna

Gli stream QLDB offrono una garanzia di consegna. at-least-once Ogni record di dati prodotto da un flusso QLDB viene inviato a Kinesis Data Streams almeno una volta. Gli stessi record possono apparire più volte in un flusso di dati Kinesis. Pertanto, è necessario disporre di una logica di deduplicazione a livello di applicazione consumer se il caso d'uso lo richiede.

Inoltre, non ci sono garanzie di ordinazione. In alcune circostanze, i blocchi e le revisioni QLDB possono essere prodotti in un flusso di dati Kinesis non funzionante. Per ulteriori informazioni, consulta Gestione di duplicati e record out-of-order.

Considerazioni sulla latenza di consegna

Gli stream QLDB in genere forniscono aggiornamenti a Kinesis Data Streams quasi in tempo reale. Tuttavia, i seguenti scenari potrebbero creare una latenza aggiuntiva prima che i nuovi dati QLDB salvati vengano emessi in un flusso di dati Kinesis:

- Kinesis può limitare i dati trasmessi in streaming da QLDB, a seconda del provisioning di Kinesis Data Streams. Ad esempio, ciò potrebbe verificarsi se sono presenti più flussi QLDB che scrivono su un singolo flusso di dati Kinesis e la frequenza di richiesta di QLDB supera la capacità della risorsa Kinesis stream. La limitazione in Kinesis può verificarsi anche quando si utilizza il provisioning su richiesta se il throughput aumenta fino a più del doppio del picco precedente in meno di 15 minuti.

  Puoi misurare questo throughput superato monitorando la metrica Kinesis. `WriteProvisionedThroughputExceeded` Per ulteriori informazioni e possibili soluzioni, vedi Come posso risolvere gli errori di limitazione in Kinesis Data Streams? .
- Con gli stream QLDB, puoi creare uno stream indefinito con data e ora di inizio precedenti e senza data e ora di fine. In base alla progettazione, QLDB inizia a emettere i nuovi dati salvati su Kinesis Data Streams solo dopo che tutti i dati precedenti della data e dell'ora di inizio specificate sono stati consegnati correttamente. Se si percepisce una latenza aggiuntiva in questo scenario, potrebbe essere necessario attendere la consegna dei dati precedenti oppure è possibile avviare lo streaming da una data e un'ora di inizio successive.

Guida introduttiva agli stream

Di seguito è riportata una panoramica di alto livello dei passaggi necessari per iniziare a trasmettere i dati del journal su Kinesis Data Streams:

1. Crea una risorsa Kinesis Data Streams. Per istruzioni, consulta Creazione e aggiornamento di flussi di dati nella Amazon Kinesis Data Streams Developer Guide.
2. Crea un ruolo IAM che consenta a QLDB di assumere le autorizzazioni di scrittura per il flusso di dati Kinesis. Per istruzioni, consulta Autorizzazioni di streaming in QLDB.
3. Crea un flusso di journal QLDB. Per istruzioni, consulta Creazione e gestione di stream in QLDB.
4. Consuma il flusso di dati Kinesis, come descritto nella sezione precedente. Consumo del tuo streaming Per esempi di codice che mostrano come utilizzare la Kinesis Client Library oppure AWS Lambda, consulta. Sviluppo con stream in QLDB

Creazione e gestione di stream in QLDB

Amazon QLDB fornisce operazioni API per creare e gestire un flusso di dati del journal dal registro ad Amazon Kinesis Data Streams. Lo stream QLDB acquisisce ogni revisione del documento inserita nel diario e la invia a un flusso di dati Kinesis.

Puoi usare il AWS Management Console, un AWS SDK o il AWS Command Line Interface (AWS CLI) per creare un flusso di journal. Inoltre, puoi anche utilizzare un AWS CloudFormationmodello per creare stream. Per ulteriori informazioni, consulta la AWS::QLDB::Streamrisorsa nella Guida per l'AWS CloudFormation utente.

Argomenti

- Parametri dello stream
- Flusso ARN
- AWS Management Console
- Stati dello stream
- Gestione degli stream compromessi

Parametri dello stream

Per creare un flusso di journal QLDB, è necessario fornire i seguenti parametri di configurazione:

Nome del libro mastro

Il registro QLDB di cui desideri trasmettere i dati del diario a Kinesis Data Streams.

Nome del flusso

Il nome che si desidera assegnare al flusso del journal QLDB. I nomi definiti dall'utente possono aiutare a identificare e indicare lo scopo di un flusso.

Il nome del flusso deve essere univoco tra gli altri flussi attivi per un determinato libro mastro. I nomi degli stream hanno gli stessi vincoli di denominazione dei nomi dei libri contabili, come definito in. Quote e limiti in Amazon QLDB

Oltre al nome dello stream, QLDB assegna un ID stream a ogni stream QLDB che crei. L'ID dello stream è unico tra tutti gli stream di un determinato registro, indipendentemente dal loro stato.

Data e ora di inizio

La data e l'ora da cui iniziare lo streaming dei dati del diario. Questo valore può corrispondere a qualsiasi data e ora del passato, ma non a quelle future.

Data e ora di fine

(Facoltativo) La data e l'ora che specificano il termine dello streaming.

Se crei uno stream indefinito senza ora di fine, devi annullarlo manualmente per terminare lo streaming. Puoi anche annullare uno stream attivo e finito che non ha ancora raggiunto la data e l'ora di fine specificate.

Flusso di dati Kinesis di destinazione

La risorsa di destinazione Kinesis Data Streams su cui lo stream scrive i record di dati. Per informazioni su come creare un flusso di dati Kinesis, consulta Creazione e aggiornamento di flussi di dati nella Amazon Kinesis Data Streams Developer Guide.

> **Important**
>
> - I flussi tra regioni e tra account non sono supportati. Il flusso di dati Kinesis specificato deve trovarsi nello stesso Regione AWS account del registro.
> - L'aggregazione dei record in Kinesis Data Streams è abilitata per impostazione predefinita. Questa opzione consente a QLDB di pubblicare più record di dati in un singolo record Kinesis Data Streams, aumentando il numero di record inviati per chiamata API.

L'aggregazione dei record ha importanti implicazioni per l'elaborazione dei record e richiede la disaggregazione nello stream consumer. Per ulteriori informazioni, consulta i concetti chiave di KPL e la deaggregazione dei consumatori nella Amazon Kinesis Data Streams Developer Guide.

Ruolo IAM

Il ruolo IAM che consente a QLDB di assumere le autorizzazioni di scrittura per il flusso di dati Kinesis. Puoi utilizzare la console QLDB per creare automaticamente questo ruolo oppure puoi crearlo manualmente in IAM. Per informazioni su come crearlo manualmente, consulta. Autorizzazioni di streaming

Per passare un ruolo a QLDB quando si richiede un flusso journal, è necessario disporre delle autorizzazioni per eseguire l'operazione `iam:PassRole` sulla risorsa del ruolo IAM.

Flusso ARN

Ogni flusso di journal QLDB è una sottorisorsa di un registro ed è identificato in modo univoco da un Amazon Resource Name (ARN). Di seguito è riportato un esempio di ARN di un flusso QLDB con un ID di flusso pari a per un registro denominato. `IiPT4brpZCqCq3f4MTHbYy` `exampleLedger`

```
arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/IiPT4brpZCqCq3f4MTHbYy
```

La sezione seguente descrive come creare e annullare uno stream QLDB utilizzando. AWS Management Console

AWS Management Console

Segui questi passaggi per creare o annullare uno stream QLDB utilizzando la console QLDB.

Per creare uno stream (console)

1. Accedi a e apri AWS Management Console la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel pannello di navigazione, seleziona Streams (Flussi).
3. Scegli Crea stream QLDB.
4. Nella pagina Crea stream QLDB, inserisci le seguenti impostazioni:

- Nome dello stream: il nome che desideri assegnare allo stream QLDB.
- Registro: il registro di cui desideri trasmettere i dati del diario.
- Data e ora di inizio: il timestamp inclusivo in UTC (Coordinated Universal Time) da cui iniziare lo streaming dei dati del diario. Il valore predefinito di questo timestamp è la data e l'ora correnti. Non può essere nel futuro e deve essere precedente alla data e all'ora di fine.
- Data e ora di fine: (Facoltativo) Il timestamp esclusivo (UTC) che specifica quando termina lo streaming. Se mantieni vuoto questo parametro, lo stream viene eseguito a tempo indeterminato fino a quando non lo annulli.
- Flusso di destinazione: la risorsa di destinazione Kinesis Data Streams su cui lo stream scrive i record di dati. Utilizzate il seguente formato ARN.

  ```
  arn:aws:kinesis:aws-region:account-id:stream/kinesis-stream-name
  ```

  Di seguito è riportato un esempio.

  ```
  arn:aws:kinesis:us-east-1:123456789012:stream/stream-for-qldb
  ```

  Gli stream tra regioni e tra account non sono supportati. Il flusso di dati Kinesis specificato deve trovarsi nello stesso Regione AWS account del registro.
- Abilita l'aggregazione dei record in Kinesis Data Streams — (Attivata per impostazione predefinita) Consente a QLDB di pubblicare più record di dati in un singolo record Kinesis Data Streams, aumentando il numero di record inviati per chiamata API.
- Accesso al servizio: il ruolo IAM che concede le autorizzazioni di scrittura QLDB al flusso di dati Kinesis.

  Per passare un ruolo a QLDB quando si richiede un flusso di journal, è necessario disporre delle autorizzazioni per eseguire *`iam:PassRole`* l'azione sulla risorsa del ruolo IAM.

  - Crea e usa un nuovo ruolo di servizio: consenti alla console di creare un nuovo ruolo per te con le autorizzazioni richieste per il flusso di dati Kinesis specificato.
  - Usa un ruolo di servizio esistente: per scoprire come creare manualmente questo ruolo in IAM, consulta. Autorizzazioni di streaming
- Tag: (Facoltativo) Aggiungi metadati allo stream allegando i tag come coppie chiave-valore. Puoi aggiungere tag allo stream per organizzarli e identificarli. Per ulteriori informazioni, consulta Assegnazione di tag alle risorse Amazon QLDB.

Scegli Aggiungi tag, quindi inserisci le coppie chiave-valore appropriate.

5. Quando le impostazioni sono quelle che desideri, scegli Crea stream QLDB.

   Se l'invio della richiesta ha esito positivo, la console torna alla pagina principale Streams ed elenca gli stream QLDB con il loro stato corrente.

6. Dopo che lo streaming è attivo, usa Kinesis per elaborare i dati di streaming con un'applicazione consumer.

   Apri la console Kinesis Data Streams all'indirizzo https://console.aws.amazon.com/kinesis/.

Per informazioni sul formato dei record di dati di flusso, consulta. Record di stream QLDB in Kinesis

Per informazioni su come gestire gli stream che generano un errore, consultaGestione degli stream compromessi.

Per annullare uno stream (console)

Non puoi riavviare uno stream QLDB dopo averlo annullato. Per riprendere la consegna dei dati a Kinesis Data Streams, puoi creare un nuovo flusso QLDB.

1. Apri la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel pannello di navigazione, seleziona Streams (Flussi).
3. Nell'elenco degli stream QLDB, seleziona lo stream attivo che desideri annullare.
4. Scegli Annulla streaming. Confermalo inserendo **`cancel stream`** nell'apposita casella.

Per informazioni sull'utilizzo dell'API QLDB con AWS un SDK o per creare e gestire flussi AWS CLI di journal, consulta. Sviluppo con stream in QLDB

Stati dello stream

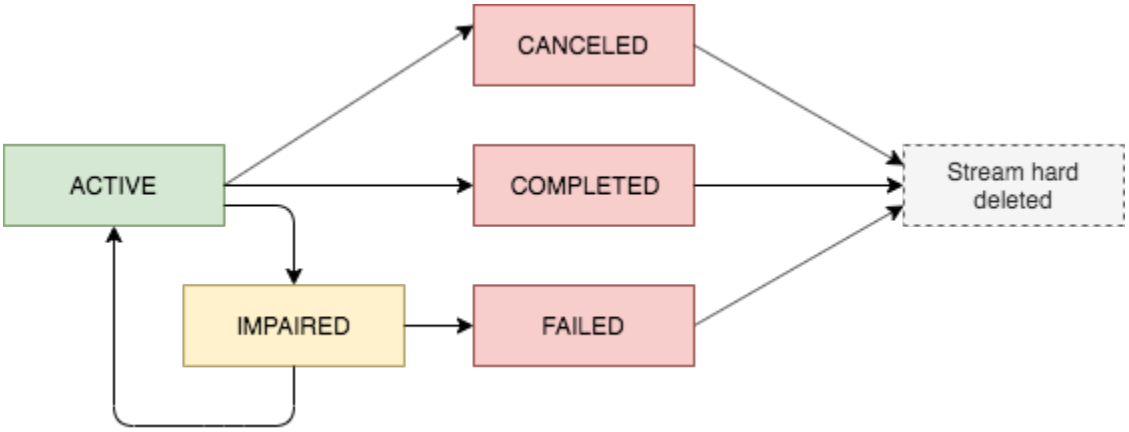
Lo stato di uno stream QLDB può essere uno dei seguenti:

- `ACTIVE`— È attualmente in streaming o è in attesa di trasmettere i dati (per uno streaming indefinito senza orario di fine).
- `COMPLETED`— Ha terminato con successo lo streaming di tutti i blocchi del journal entro l'intervallo di tempo specificato. Si tratta di uno stato terminale.

- `CANCELED`— È stato interrotto da una richiesta dell'utente prima dell'ora di fine specificata e non trasmette più attivamente i dati in streaming. Si tratta di uno stato terminale.
- `IMPAIRED`— Non è in grado di scrivere record su Kinesis a causa di un errore che richiede l'intervento dell'utente. Si tratta di uno stato non terminale ripristinabile.

  Se risolvi l'errore entro un'ora, lo stream passa automaticamente allo stato. `ACTIVE` Se l'errore rimane irrisolto dopo un'ora, lo stream passa automaticamente allo `FAILED` stato.
- `FAILED`— Non è in grado di scrivere record su Kinesis a causa di un errore e si trova in uno stato terminale irrecuperabile.

Il diagramma seguente illustra come una risorsa di flusso QLDB può passare da uno stato all'altro.



Scadenza per gli stream terminali

Le risorse di streaming che si trovano in uno stato terminale (`CANCELED`COMPLETED, e`FAILED`) sono soggette a un periodo di conservazione di 7 giorni. Vengono eliminate automaticamente dopo la scadenza di questo limite.

Dopo l'eliminazione di un flusso di terminale, non è più possibile utilizzare la console QLDB o l'API QLDB per descrivere o elencare la risorsa di flusso.

Gestione degli stream compromessi

Se il tuo stream riscontra un errore, passa prima allo stato. `IMPAIRED` QLDB continua a `IMPAIRED` riprovare lo streaming per un massimo di un'ora.

Se risolvi l'errore entro un'ora, lo stream passa automaticamente allo stato. `ACTIVE` Se l'errore rimane irrisolto dopo un'ora, lo stream passa automaticamente allo `FAILED` stato.

Uno stream danneggiato o non riuscito può avere una delle seguenti cause di errore:

- `KINESIS_STREAM_NOT_FOUND`— La risorsa Kinesis Data Streams di destinazione non esiste. Verifica che il flusso di dati Kinesis fornito nella richiesta di flusso QLDB sia corretto. Quindi, vai su Kinesis e crea il flusso di dati che hai specificato.
- `IAM_PERMISSION_REVOKED`— QLDB non dispone di autorizzazioni sufficienti per scrivere record di dati nel flusso di dati Kinesis specificato. Verifica di aver definito una policy per il flusso di dati Kinesis specificato che conceda al servizio QLDB () `qldb.amazonaws.com` le autorizzazioni per le seguenti azioni:
  - `kinesis:PutRecord`
  - `kinesis:PutRecords`
  - `kinesis:DescribeStream`
  - `kinesis:ListShards`

Monitoraggio dei flussi compromessi

Se uno stream viene interrotto, la console QLDB visualizza un banner che mostra i dettagli sullo stream e l'errore riscontrato. Puoi anche utilizzare l'operazione `DescribeJournalKinesisStream` API per ottenere lo stato di uno stream e la causa dell'errore sottostante.

Inoltre, puoi utilizzare Amazon CloudWatch per creare un allarme che monitora la `IsImpaired` metrica di uno stream. Per informazioni sul monitoraggio delle metriche CloudWatch QLDB con, vedere. Dimensioni e parametri di Amazon QLDB

Sviluppo con stream in QLDB

Questa sezione riassume le operazioni API che puoi utilizzare con un AWS SDK o AWS CLI per creare e gestire flussi di journal in Amazon QLDB. Descrive inoltre le applicazioni di esempio che dimostrano queste operazioni e utilizzano la Kinesis Client Library (KCL) o AWS Lambda per implementare uno stream consumer.

Puoi utilizzare KCL per creare applicazioni consumer per Amazon Kinesis Data Streams. KCL semplifica la codifica fornendo astrazioni utili sull'API Kinesis Data Streams di basso livello. Per ulteriori informazioni su KCL, consulta Using the Kinesis Client Library nella Amazon Kinesis Data Streams Developer Guide.

Indice

API QLDB Journal Stream

L'API QLDB fornisce le seguenti operazioni di flusso del diario per l'utilizzo da parte dei programmi applicativi:

- `StreamJournalToKinesis`— Crea un flusso di journal per un determinato registro QLDB. Lo stream acquisisce ogni revisione del documento inserita nel diario del registro e invia i dati a una risorsa Kinesis Data Streams specificata.
  - L'aggregazione dei record in Kinesis Data Streams è abilitata per impostazione predefinita. Questa opzione consente a QLDB di pubblicare più record di dati in un singolo record Kinesis Data Streams, aumentando il numero di record inviati per chiamata API.

    L'aggregazione dei record ha importanti implicazioni per l'elaborazione dei record e richiede la disaggregazione nello stream consumer. Per ulteriori informazioni, consulta i concetti chiave di KPL e la deaggregazione dei consumatori nella Amazon Kinesis Data Streams Developer Guide.
- `DescribeJournalKinesisStream`— Restituisce informazioni dettagliate su un determinato flusso di journal QLDB. L'output include l'ARN, il nome dello stream, lo stato corrente, l'ora di creazione e i parametri della richiesta di creazione dello stream originale.
- `ListJournalKinesisStreamsForLedger`— Restituisce un elenco di tutti i descrittori di stream del journal QLDB per un determinato registro. L'output di ogni descrittore di flusso include gli stessi dettagli restituiti da. `DescribeJournalKinesisStream`
- `CancelJournalKinesisStream`— Termina un determinato flusso di journal QLDB. Prima che uno stream possa essere annullato, deve essere il suo stato attuale. `ACTIVE`

  Non puoi riavviare uno stream dopo averlo annullato. Per riprendere la consegna dei dati a Kinesis Data Streams, puoi creare un nuovo flusso QLDB.

Per una descrizione completa di queste operazioni API, consulta la. Documentazione di riferimento dell'API Amazon QLDB

Per informazioni sulla creazione e la gestione di stream di journal utilizzando AWS CLI, vedere il AWS CLI Command Reference.

Applicazioni di esempio

QLDB fornisce applicazioni di esempio che dimostrano varie operazioni utilizzando stream di journal. Queste applicazioni sono open source sul sito Samples.AWS GitHub

Argomenti

- Operazioni di base (Java)
- Integrazione con OpenSearch Service (Python)
- Integrazione con Amazon SNS e Amazon SQS (Python)

Operazioni di base (Java)

Per un esempio di codice Java che dimostra le operazioni di base per i flussi di journal QLDB, consulta il repository aws-samples/ -java. GitHub amazon-qldb-dmv-sample Per istruzioni su come scaricare e installare questa applicazione di esempio, consulta. Installazione dell'applicazione di esempio Java Amazon QLDB

> **Note**
>
> Dopo aver installato l'applicazione, non procedete al passaggio 1 del tutorial su Java per creare un libro mastro. Questa applicazione di esempio per lo streaming crea il `vehicle-registration` registro per te.

Questa applicazione di esempio racchiude il codice sorgente completo di Tutorial su Java e le relative dipendenze, inclusi i seguenti moduli:

- AWS SDK for Java— Per creare ed eliminare le risorse QLDB e Kinesis Data Streams, inclusi i registri, i flussi di journal QLDB e i flussi di dati Kinesis.
- Driver Amazon QLDB per Java— Per eseguire transazioni di dati su un registro utilizzando istruzioni PartiQl, inclusa la creazione di tabelle e l'inserimento di documenti.
- Kinesis Client Library: per utilizzare ed elaborare i dati da un flusso di dati Kinesis.

Esecuzione del codice

La StreamJournalclasse contiene codice tutorial che illustra le seguenti operazioni:

1. Crea un registro denominato`vehicle-registration`, crea tabelle e caricale con dati di esempio.

   > **Note**
   >
   > Prima di eseguire questo codice, assicurati di non avere già un registro attivo denominato. `vehicle-registration`

2. Crea un flusso di dati Kinesis, un ruolo IAM che consente a QLDB di assumere le autorizzazioni di scrittura per il flusso di dati Kinesis e un flusso di journal QLDB.
3. Usa KCL per avviare un lettore di stream che elabora il flusso di dati Kinesis e registra ogni record di dati QLDB.
4. Usa i dati del flusso per convalidare la catena hash del registro di esempio. `vehicle-registration`
5. Pulisci tutte le risorse interrompendo lo stream reader, annullando lo stream del journal QLDB, eliminando il registro ed eliminando il flusso di dati Kinesis.

Per eseguire il codice del `StreamJournal` tutorial, inserisci il seguente comando Gradle dalla directory principale del tuo progetto.

```
./gradlew run -Dtutorial=streams.StreamJournal
```

Integrazione con OpenSearch Service (Python)

Per un'applicazione di esempio in Python che dimostra come integrare un flusso QLDB con Amazon OpenSearch Service, consulta il repository aws-samples/ -. GitHub amazon-qldb-streaming-amazon opensearch-service-sample-python Questa applicazione utilizza una AWS Lambda funzione per implementare un consumatore Kinesis Data Streams.

Per clonare il repository, inserisci il seguente comando. `git`

```
git clone https://github.com/aws-samples/amazon-qldb-streaming-amazon-opensearch-
service-sample-python.git
```

Per eseguire l'applicazione di esempio, consultate il file README su GitHub per le istruzioni.

Integrazione con Amazon SNS e Amazon SQS (Python)

Per un'applicazione di esempio in Python che dimostra come integrare uno stream QLDB con Amazon Simple Notification Service (Amazon SNS), consulta il repository aws-samples/ -. GitHub amazon-qldb-streams-dmv sample-lambda-python

Questa applicazione utilizza una AWS Lambda funzione per implementare un consumatore Kinesis Data Streams. Invia messaggi a un argomento Amazon SNS a cui è associata una coda Amazon Simple Queue Service (Amazon SQS).

Per clonare il repository, inserisci il seguente comando. `git`

```
git clone https://github.com/aws-samples/amazon-qldb-streams-dmv-sample-lambda-
python.git
```

Per eseguire l'applicazione di esempio, consultate il file README su GitHub per le istruzioni.

Record di stream QLDB in Kinesis

Uno stream Amazon QLDB scrive tre tipi di record di dati su una determinata risorsa Amazon Kinesis Data Streams: controllo, riepilogo dei blocchi e dettagli di revisione. Tutti e tre i tipi di record sono scritti nella rappresentazione binaria del formato Amazon Ion.

I record di controllo indicano l'inizio e il completamento dei tuoi stream QLDB. Ogni volta che viene eseguita una revisione nel diario, uno stream QLDB scrive tutti i dati dei blocchi di journal associati nei record di riepilogo dei blocchi e dei dettagli di revisione.

I tre tipi di record sono polimorfici. Sono tutti costituiti da un record comune di primo livello che contiene l'ARN del flusso QLDB, il tipo di record e il payload del record. Questo record di primo livello ha il seguente formato.

```
{
  qldbStreamArn: string,
  recordType: string,
  payload: {
    //control | block summary | revision details record
  }
}
```

Il `recordType` campo può avere uno dei tre valori seguenti:

- `CONTROL`
- `BLOCK_SUMMARY`
- `REVISION_DETAILS`

Le seguenti sezioni descrivono il formato e il contenuto di ogni singolo record di payload.

> **Note**
>
> QLDB scrive tutti i record di flusso su Kinesis Data Streams nella rappresentazione binaria di Amazon Ion. I seguenti esempi sono forniti nella rappresentazione testuale di Ion per illustrare il contenuto del record in un formato leggibile.

Argomenti

Registri di controllo

Uno stream QLDB scrive record di controllo per indicarne gli eventi di inizio e completamento. Di seguito sono riportati alcuni esempi di record di controllo con dati di esempio per ciascuno di essi: `controlRecordType`

- `CREATED`— Il primo record che uno stream QLDB scrive su Kinesis per indicare che lo stream appena creato è attivo.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
    controlRecordType:"CREATED"
  }
}
```

- `COMPLETED`— L'ultimo record che uno stream QLDB scrive su Kinesis per indicare che lo stream ha raggiunto la data e l'ora di fine specificate. Questo record non viene scritto se annulli lo streaming.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
    controlRecordType:"COMPLETED"
  }
}
```

Blocca i record di ri

Un record di riepilogo dei blocchi rappresenta un blocco del diario in cui vengono salvate le revisioni dei documenti. Un blocco è un oggetto che viene salvato nel tuo journal QLDB durante una transazione.

Il payload di un record di riepilogo dei blocchi contiene l'indirizzo del blocco, il timestamp e altri metadati della transazione che ha eseguito il blocco. Include anche gli attributi di riepilogo delle revisioni nel blocco e le istruzioni PartiQL che le hanno salvate. Di seguito è riportato un esempio di record di riepilogo dei blocchi con dati di esempio.

> **Note**
>
> Questo esempio di riepilogo dei blocchi viene fornito solo a scopo informativo. Gli hash mostrati non sono valori hash calcolati reali.

```
{
  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"
      }
    ]
  }
}
```

Sul `revisionSummaries` campo, alcune revisioni potrebbero non avere un. `documentId` Si tratta di revisioni di sistema solo interne che non contengono dati utente. Uno stream QLDB include queste revisioni nei rispettivi record di riepilogo dei blocchi perché gli hash di queste revisioni fanno parte dell'intera catena hash della rivista. La catena hash completa è necessaria per la verifica crittografica.

Solo le revisioni che dispongono di un ID documento vengono pubblicate in record di dettagli di revisione separati, come descritto nella sezione seguente.

Record dei dettagli delle revisioni

Un record dei dettagli di revisione rappresenta una revisione del documento salvata nel diario. Il payload contiene tutti gli attributi della visualizzazione salvata della revisione, insieme al nome e all'ID della tabella associati. Di seguito è riportato un esempio di record di revisione con dati di esempio.

```
{
  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"
      }
    }
  }
}
```

Gestione di duplicati e record out-of-order

Gli stream QLDB possono pubblicare duplicati e record su out-of-order Kinesis Data Streams. Pertanto, un'applicazione consumer potrebbe dover implementare una propria logica per identificare e gestire tali scenari. I record di riepilogo dei blocchi e dei dettagli di revisione includono campi che è possibile utilizzare a questo scopo. In combinazione con le funzionalità dei servizi a valle, questi campi possono indicare sia un'identità univoca che un ordine rigoroso per i record.

Ad esempio, considera uno stream che integra QLDB con OpenSearch un indice per fornire funzionalità di ricerca di testo completo sui documenti. In questo caso d'uso, è necessario evitare di indicizzare le revisioni stale () out-of-order di un documento. Per applicare l'ordinamento e la deduplicazione, è possibile utilizzare i campi ID del documento e versione esterna in OpenSearch, insieme ai campi ID documento e versione in un record di dettagli di revisione.

Per un esempio di logica di deduplicazione in un'applicazione di esempio che integra QLDB con OpenSearch Amazon Service, consulta il repository aws-samples/ -. GitHub amazon-qldb-streaming-amazon opensearch-service-sample-python

Autorizzazioni di streaming in QLDB

Prima di creare uno stream Amazon QLDB, devi fornire a QLDB le autorizzazioni di scrittura per la risorsa Amazon Kinesis Data Streams specificata. Se utilizzi un sistema gestito dal cliente AWS KMS key per la crittografia lato server del tuo flusso Kinesis, devi anche fornire a QLDB le autorizzazioni per utilizzare la chiave di crittografia simmetrica specificata. Kinesis Data Streams non supporta le chiavi KMS asimmetriche.

Per fornire al tuo stream QLDB le autorizzazioni necessarie, puoi fare in modo che QLDB assuma un ruolo di servizio IAM con le politiche di autorizzazione appropriate. Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.

> **Note**
>
> Per passare un ruolo a QLDB quando si richiede un flusso journal, è necessario disporre delle autorizzazioni per eseguire l'operazione `iam:PassRole` sulla risorsa del ruolo IAM. Ciò si aggiunge all'`qldb:StreamJournalToKinesis`autorizzazione sulla sottorisorsa del flusso QLDB.
>
> Per informazioni su come controllare l'accesso a QLDB utilizzando IAM, consulta. Come funziona Amazon QLDB con IAM Per un esempio di policy QLDB, vedere. Esempi di policy basate sull'identità per Amazon QLDB

In questo esempio, crei un ruolo che consente a QLDB di scrivere record di dati su un flusso di dati Kinesis per tuo conto. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.

Se stai trasmettendo in streaming un journal QLDB per Account AWS la prima volta, devi prima creare un ruolo IAM con le politiche appropriate procedendo come segue. In alternativa, puoi utilizzare la console QLDB per creare automaticamente il ruolo per te. Altrimenti, puoi scegliere un ruolo che hai creato in precedenza.

Argomenti

- Creazione di una policy di autorizzazione
- Creazione di un ruolo IAM

Creazione di una policy di autorizzazione

Completa i seguenti passaggi per creare una politica di autorizzazioni per uno stream QLDB. Questo esempio mostra una policy di Kinesis Data Streams che concede le autorizzazioni QLDB per scrivere record di dati nel flusso di dati Kinesis specificato. Se applicabile, l'esempio mostra anche una politica chiave che consente a QLDB di utilizzare la chiave KMS di crittografia simmetrica.

Per ulteriori informazioni sulle policy di Kinesis Data Streams, consulta Controllo dell'accesso alle risorse di Amazon Kinesis Data Streams tramite IAM e Permissions to use KMS key generate dall'utente nella Amazon Kinesis Data Streams Developer Guide. Per ulteriori informazioni sulle politiche chiave, consulta Using AWS KMS key policy nella Developer Guide. AWS KMSAWS Key Management Service

> **Note**
>
> Il flusso di dati Kinesis e la chiave KMS devono essere entrambi nello stesso account del Regione AWS registro QLDB.

Come utilizzare l'editor di policy JSON per creare una policy

1. Accedi AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione a sinistra, seleziona Policies (Policy).

   Se è la prima volta che selezioni Policy, verrà visualizzata la pagina Benvenuto nelle policy gestite. Seleziona Inizia.
3. Nella parte superiore della pagina, scegli Crea policy.
4. Scegli la scheda JSON.
5. Specificare un documento della policy JSON.
   - Se utilizzi una chiave KMS gestita dal cliente per la crittografia lato server del tuo flusso Kinesis, utilizza il seguente documento di policy di esempio. Per utilizzare questa politica, sostituisci *us-east-1*, 123456789012 e *kinesis-stream-name1234abcd-12ab-34cd-56ef-1234567890ab* nell'esempio con *le tue informazioni*.

     ```
     {
         "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"
        }
    ]
}
```

- Altrimenti, usa il seguente documento politico di esempio. Per utilizzare questa politica, sostituisci *us-east-1,* 123456789012 *kinesis-stream-name*e nell'esempio con le tue informazioni.

```
{
    "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. Scegli Verifica policy.

> **Note**
>
> È possibile passare tra le schede Visual editor (Editor visivo) e JSON in qualsiasi momento. Se tuttavia si apportano modifiche o se si seleziona Review policy (Rivedi

policy) nella scheda Visual editor (Editor visivo), IAM potrebbe modificare la policy per ottimizzarla per l'editor visivo. Per ulteriori informazioni, consulta Modifica della struttura delle policy nella Guida per l'utente di IAM.

7. Nella pagina Review policy (Rivedi policy), inserisci i valori per Name (Nome) e Description (Descrizione) (facoltativa) per la policy che stai creando. Consulta il Summary (Riepilogo) della policy per visualizzare le autorizzazioni concesse dalla policy. Seleziona Create policy (Crea policy) per salvare il proprio lavoro.

Creazione di un ruolo IAM

Dopo aver creato una politica di autorizzazioni per il tuo stream QLDB, puoi quindi creare un ruolo IAM e allegare la tua policy ad esso.

Per creare il ruolo di servizio per QLDB (console IAM)

1. Accedi AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione della console IAM, scegliere Ruoli e quindi Crea ruolo.
3. Per Trusted entity type (Tipo di entità attendibile), scegli Servizio AWS.
4. Per Servizio o caso d'uso, scegli QLDB, quindi scegli lo use case QLDB.
5. Seleziona Successivo.
6. Seleziona la casella accanto alla politica che hai creato nei passaggi precedenti.
7. (Facoltativo) Impostare un limite delle autorizzazioni. Questa è una caratteristica avanzata disponibile per i ruoli di servizio, ma non per i ruoli collegati ai servizi.

   a. Apri la sezione Imposta i limiti delle autorizzazioni, quindi scegli Usa un limite di autorizzazioni per controllare il numero massimo di autorizzazioni per i ruoli.

      IAM include un elenco delle politiche AWS gestite e gestite dal cliente nel tuo account.

   b. Selezionare la policy da utilizzare per il limite delle autorizzazioni.
8. Seleziona Successivo.
9. Inserisci il nome del ruolo o il suffisso del nome del ruolo per aiutarti a identificare lo scopo del ruolo.

> **Important**
>
> Quando assegnate un nome a un ruolo, tenete presente quanto segue:
>
> - I nomi dei ruoli devono essere univoci all'interno del tuo Account AWS account e non possono essere resi unici per caso.
>
> Ad esempio, non creare ruoli denominati entrambi **`PRODROLE`** e**`prodrole`**. Quando un nome di ruolo viene utilizzato in una policy o come parte di un ARN, il nome del ruolo fa distinzione tra maiuscole e minuscole, tuttavia quando un nome di ruolo viene visualizzato dai clienti nella console, ad esempio durante il processo di accesso, il nome del ruolo non fa distinzione tra maiuscole e minuscole.
> - Non è possibile modificare il nome del ruolo dopo averlo creato perché altre entità potrebbero fare riferimento al ruolo.

10. (Facoltativo) In Descrizione, inserisci una descrizione per il ruolo.
11. (Facoltativo) Per modificare i casi d'uso e le autorizzazioni per il ruolo, nelle sezioni Passo 1: Seleziona entità attendibili o Passaggio 2: Aggiungi autorizzazioni, scegli Modifica.
12. (Facoltativo) Per facilitare l'identificazione, l'organizzazione o la ricerca del ruolo, aggiungi tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Applicazione di tag alle risorse IAM nella Guida per l'utente di IAM.
13. Verificare il ruolo e quindi scegliere Create role (Crea ruolo).

Il seguente documento JSON è un esempio di policy di fiducia che consente a QLDB di assumere un ruolo IAM con autorizzazioni specifiche associate.

```
{
    "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**
>
> Questo esempio di politica di fiducia mostra come utilizzare le chiavi di contesto *aws:SourceArn* e di contesto della condizione *aws:SourceAccount* globale per evitare il confuso problema del vice. Con questa politica di fiducia, QLDB può assumere il ruolo di qualsiasi flusso QLDB presente nell'account solo per il registro. `123456789012` `myExampleLedger`
> Per ulteriori informazioni, consulta Prevenzione del confused deputy tra servizi.

Dopo aver creato il tuo ruolo IAM, torna alla console QLDB e aggiorna la pagina dello stream Crea QLDB in modo che possa trovare il tuo nuovo ruolo.

Errori comuni per i flussi di journal in QLDB

Questa sezione descrive gli errori di runtime generati da Amazon QLDB per le richieste di stream del journal.

Di seguito è riportato un elenco di eccezioni comuni restituite dal servizio. Ogni eccezione include il messaggio di errore specifico, seguito da una breve descrizione e suggerimenti per possibili soluzioni.

AccessDeniedException

Messaggio: Utente: userARN non è autorizzato a eseguire: iam: PassRole on resource: roLearn

Non disponi delle autorizzazioni per passare un ruolo IAM al servizio QLDB. QLDB richiede un ruolo per tutte le richieste di stream del journal e devi disporre delle autorizzazioni per passare

questo ruolo a QLDB. Il ruolo fornisce a QLDB le autorizzazioni di scrittura nella risorsa Amazon Kinesis Data Streams specificata.

Verifica di definire una policy IAM che conceda l'autorizzazione a eseguire l'operazione `PassRole` API sulla risorsa del ruolo IAM specificata per il servizio QLDB (). `qldb.amazonaws.com` Per un esempio di policy, consulta Esempi di policy basate sull'identità per Amazon QLDB.

IllegalArgumentException

`Messaggio: QLDB ha riscontrato un errore durante la convalida di Kinesis Data Streams: Response from Kinesis: ErrorCode ErrorMessage`

Una possibile causa di questo errore è che la risorsa Kinesis Data Streams fornita non esiste. Oppure, QLDB non dispone di autorizzazioni sufficienti per scrivere record di dati nel flusso di dati Kinesis specificato.

Verifica che il flusso di dati Kinesis fornito nella richiesta di streaming sia corretto. Per ulteriori informazioni, consulta Creazione e aggiornamento di flussi di dati nella Amazon Kinesis Data Streams Developer Guide.

Inoltre, verifica di aver definito una policy per il flusso di dati Kinesis specificato che conceda al servizio QLDB () `qldb.amazonaws.com` le autorizzazioni per le seguenti azioni. Per ulteriori informazioni, consulta Autorizzazioni di streaming.

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

Messaggio: risposta inaspettata da Kinesis Data Streams durante la convalida della configurazione Kinesis. *`Risposta di Kinesis: ErrorCode ErrorMessage`*

Il tentativo di scrivere record di dati nel flusso di dati Kinesis fornito non è riuscito con la risposta di errore Kinesis fornita. Per ulteriori informazioni sulle possibili cause, consulta la sezione Risoluzione dei problemi dei produttori di Amazon Kinesis Data Streams nella Amazon Kinesis Data Streams Developer Guide.

IllegalArgumentException

Messaggio: la data di inizio non deve essere successiva alla data di fine.

Entrambi `InclusiveStartTime` `ExclusiveEndTime` devono essere nel formato di data e ora ISO 8601 e nel formato UTC (Coordinated Universal Time).

IllegalArgumentException

Messaggio: la data di inizio non può essere futura.

Entrambi `InclusiveStartTime` i formati `ExclusiveEndTime` devono essere in formato `ISO 8601` data e ora e UTC.

LimitExceededException

Messaggio: superato il limite di 5 stream Journal in esecuzione simultanea su Kinesis Data Streams

QLDB impone un limite predefinito di cinque stream di journal simultanei.

Gestione dei registri in Amazon QLDB

Questo capitolo descrive come utilizzare l'API QLDB, ilAWS Command Line Interface (AWS CLI) e comeAWS CloudFormation eseguire operazioni di gestione dei libri contabili in Amazon QLDB.

Puoi eseguire queste stesse attività utilizzando la AWS Management Console. Per ulteriori informazioni, consulta Accesso ad Amazon QLDB tramite la console.

Argomenti

- Operazioni di base per i libri mastri Amazon QLDB
- Creazione di risorse Amazon QLDB conAWS CloudFormation
- Assegnazione di tag alle risorse Amazon QLDB

Operazioni di base per i libri mastri Amazon QLDB

Puoi usare l'API QLDB o ilAWS Command Line Interface (AWS CLI) per creare, aggiornare ed eliminare i registri in Amazon QLDB. Puoi anche elencare tutti i libri mastri nel tuo account o ottenere informazioni su un libro mastro specifico.

I seguenti argomenti forniscono brevi esempi di codice che mostrano i passaggi comuni per le operazioni di contabilità utilizzandoAWS SDK for Java e ilAWS CLI.

Argomenti

- Creazione di un registro
- Descrizione di un libro mastro
- Aggiornamento di un libro mastro
- Aggiornamento di una modalità di autorizzazione del registro
- Eliminazione di un libro mastro
- Registri di quotazione

Per esempi di codice che dimostrano queste operazioni in un'applicazione di esempio completa, consulta i seguentiNozioni base sul driver tutorial e GitHub repository:

- Java: Tutorial | GitHub repository
- Node.js: Tutorial | GitHub repository

- Python: Tutorial | GitHub repository

Creazione di un registro

Usa l'`CreateLedger`operazione per creare un libro mastro nel tuoAccount AWS. È necessario fornire le seguenti informazioni:

- Nome libro mastro: il nome della contabilità che desideri creare nel tuo account. Il nome deve essere univoco tra tutti i libri mastri nel momentoRegione AWS.

  I vincoli di denominazione per i nomi dei libri mastri sono definiti inQuote e limiti in Amazon QLDB.

- Modalità autorizzazioni: la modalità di autorizzazione da assegnare alla contabilità. Seleziona una delle seguenti opzioni:
  - Consenti tutto: una modalità di autorizzazione legacy che consente il controllo degli accessi con granularità a livello di API per i libri mastri.

    Questa modalità consente agli utenti che dispongono dell'autorizzazione API `SendCommand` per questo libro mastro per eseguire tutti i comandi PartiQL (quindi, `ALLOW_ALL`) su qualsiasi tabella nel libro mastro specificato. Questa modalità ignora tutte le policy di autorizzazione IAM a livello di tabella o di comando create per il libro mastro.
  - Standard: (impostazione consigliata) una modalità di autorizzazione che consente il controllo degli accessi con una granularità più fine per libri mastri, tabelle e comandi PartiQL. Ti consigliamo di utilizzare questa modalità di autorizzazione per incrementare la sicurezza dei dati nel libro mastro.

    Per impostazione predefinita, questa modalità nega tutte le richieste di eseguire qualsiasi comando PartiQL su qualsiasi tabella in questo libro mastro. Per consentire i comandi PartiQL, devi creare le policy di autorizzazione IAM per risorse di tabelle e operazioni PartiQL specifiche, oltre all'autorizzazione`SendCommand` API per il libro mastro. Per informazioni, consulta Guida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

- Protezione da eliminazione: (Facoltativo) il flag che impedisce a un utente qualsiasi di eliminare una contabilità. Se non la si specifica durante la creazione della contabilità, questa caratteristica è abilitata (`true`) per impostazione predefinita.

  Se la protezione dall'eliminazione è abilitata, è necessario innanzitutto disabilitarla prima di poter eliminare il libro mastro. Puoi disabilitarla usando l'`UpdateLedger`operazione per impostare il flag su`false`.

- AWS KMS key— (Facoltativo) La chiave inAWS Key Management Service (AWS KMS) da utilizzare per la crittografia dei dati inattivi. Scegliere uno dei seguenti tipi diAWS KMS keys:
  - AWSchiave KMS di proprietà: usa una chiave KMS di proprietà e gestita da per tuoAWS conto.

    Se non si definisce questo parametro durante la creazione del libro mastro, il libro mastro utilizza questo tipo di chiave per impostazione predefinita. Puoi anche usare la stringa`AWS_OWNED_KMS_KEY` per specificare questo tipo di chiave. Questa opzione non richiede alcuna configurazione aggiuntiva.
  - Chiave KMS gestita dal cliente: usa una chiave KMS di crittografia simmetrica nel tuo account che crei, possiedi e gestisci. QLDB non supporta chiavi asimmetriche.

    Questa opzione richiede la creazione di una chiave KMS o l'utilizzo di una chiave esistente nel tuo account. Per istruzioni sulla creazione di una chiave gestita dal cliente, consulta Creazione di chiavi KMS con crittografia simmetrica nella Guida per gliAWS Key Management Service sviluppatori.

    Puoi specificare una chiave KMS gestita dal cliente utilizzando un ID, un alias o un nome di risorsa Amazon (ARN). Per ulteriori informazioni, consulta Identificatori chiave (KeyId) nella Guida per gliAWS Key Management Service sviluppatori.

    > **Note**
    >
    > Le chiavi tra Regioni non sono supportate. La chiave KMS specificata deve trovarsi nelloRegione AWS stesso libro mastro.

  Per ulteriori informazioni, consulta Crittografia inattiva in Amazon QLDB.
- Tag: (Facoltativo) Aggiungere metadati alla contabilità collegando i tag come coppie chiave-valore. Puoi aggiungere tag al libro mastro per facilitarne l'organizzazione e l'identificazione. Per ulteriori informazioni, consulta Assegnazione di tag alle risorse Amazon QLDB.

Il libro mastro non è pronto per l'uso fino a quando QLDB lo crea e ne imposta lo stato su`ACTIVE`.

Creazione di un libro mastro (Java)

Per creare un registro utilizzandoAWS SDK for Java

1. Creare un'istanza della classe `AmazonQLDB`.

2. Crea un'istanza della classe `CreateLedgerRequest` per fornire le informazioni della richiesta.

   È necessario fornire il nome del registro e una modalità di autorizzazione.
3. Eseguire il metodo `createLedger` fornendo l'oggetto della richiesta come parametro.

La`createLedger` richiesta restituisce un`CreateLedgerResult` oggetto che contiene informazioni sul libro mastro. Consulta la sezione successiva per un esempio di utilizzo dell'`DescribeLedger`operazione per controllare lo stato del libro contabile dopo averlo creato.

I seguenti esempi dimostrano i passaggi precedenti.

Example — Utilizzo delle impostazioni di configurazione predefinite

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
CreateLedgerRequest request = new CreateLedgerRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
CreateLedgerResult result = client.createLedger(request);
```

> **Note**
>
> Il libro mastro utilizza le seguenti impostazioni predefinite se non le specificate:
>
> - Protezione dall'eliminazione — Attivata (`true`).
> - Chiave KMS: chiave KMS diAWS proprietà.

Example — Disattiva la protezione dall'eliminazione, utilizza una chiave KMS gestita dal cliente e allega tag

```
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);
```

Creazione di un libro mastro (AWS CLI)

Crea un nuovo libro mastro denominato`vehicle-registration` utilizzando le impostazioni di configurazione predefinite.

Example

```
aws qldb create-ledger --name vehicle-registration --permissions-mode STANDARD
```

> **Note**
>
> Il libro mastro utilizza le seguenti impostazioni predefinite se non le specificate:
>
> - Protezione dall'eliminazione — Attivata (`true`).
> - Chiave KMS: chiave KMS diAWS proprietà.

In alternativa, crea un nuovo libro mastro denominato`vehicle-registration` con protezione dall'eliminazione disattivata, con una chiave KMS specificata gestita dal cliente e con tag specifici.

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
```

Creazione di un libro mastro (AWS CloudFormation)

Puoi anche usare un AWS CloudFormationmodello per creare libri mastri. Per ulteriori informazioni, consulta la AWS::QLDB::Ledgerrisorsa nella Guida per l'AWS CloudFormationutente.

Descrizione di un libro mastro

Utilizzare l'`DescribeLedger`operazione per visualizzare i dettagli relativi a un libro mastro. È necessario specificare il nome della contabilità. Il formato dell'output da `DescribeLedger` è identico a quello da `CreateLedger`. Include le seguenti informazioni:

- Nome libro mastro: il nome della contabilità che desideri descrivere.
- ARN: il nome della risorsa Amazon (ARN) per il libro mastro nel seguente formato.

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

- Protezione dall'eliminazione: il contrassegno che indica se la funzione di protezione dall'eliminazione è abilitata.
- Data e ora di creazione: la data e l'ora, nel formato dell'ora dell'epoca, di quando è stato creato il libro mastro.
- Stato: lo stato attuale del libro mastro. Può essere uno dei seguenti valori:
  - `CREATING`
  - `ACTIVE`
  - `DELETING`
  - `DELETED`
- Modalità autorizzazioni: la modalità delle autorizzazioni assegnata al libro mastro. Può essere uno dei seguenti valori:
  - `ALLOW_ALL`— Una modalità di autorizzazione legacy che consente il controllo degli accessi con granularità a livello di API per i libri mastri.
  - `STANDARD`— Una modalità di autorizzazione che consente il controllo degli accessi con una granularità più fine per libri mastri, tabelle e comandi PartiQL.
- Descrizione della crittografia: informazioni sulla crittografia dei dati a riposo nel libro mastro. Ciò include i seguenti articoli:
  - AWS KMS keyARN: l'ARN della chiave KMS gestita dal cliente che il libro mastro utilizza per la crittografia a riposo. Se non è definita, il libro mastro utilizza una chiave KMSAWS di proprietà per la crittografia.
  - Stato di crittografia: lo stato corrente della crittografia inattiva per il libro mastro. Può essere uno dei seguenti valori:
    - `ENABLED`— La crittografia è completamente abilitata utilizzando la chiave specificata.

- `UPDATING`— La modifica della chiave specificata viene elaborata attivamente.

  Le modifiche chiave in QLDB sono asincrone. Il libro mastro è completamente accessibile senza alcun impatto sulle prestazioni durante l'elaborazione della modifica chiave. La quantità di tempo necessaria per aggiornare una chiave varia a seconda della dimensione della contabilità.

- `KMS_KEY_INACCESSIBLE`— La chiave KMS specificata gestita dal cliente non è accessibile e il registro è danneggiato. La chiave è stata disattivata o eliminata oppure le concessioni sulla chiave sono state revocate. Quando un libro mastro è danneggiato, non è accessibile e non accetta alcuna richiesta di lettura o scrittura.

  Un libro mastro danneggiato torna automaticamente allo stato attivo dopo aver ripristinato le concessioni sulla chiave o dopo aver riattivato la chiave che era stata disattivata. Tuttavia, l'eliminazione di una chiave KMS gestita dal cliente è irreversibile. Dopo l'eliminazione di una chiave, non è più possibile accedere ai libri mastri protetti da tale chiave e i dati diventano irrecuperabili in modo permanente.

- InaccessibileAWS KMS key: la data e l'ora, in formato epoch-time, in cui la chiave KMS è diventata inaccessibile per la prima volta, in caso di errore.

  Questo non è definito se la chiave KMS è accessibile.

Per ulteriori informazioni, consulta Crittografia inattiva in Amazon QLDB.

> **Note**
>
> Dopo aver creato un registro QLDB, diventa pronto per l'uso quando il suo stato cambia da`CREATING` a`ACTIVE`.

Descrizione di un libro mastro (Java)

Per descrivere un libro mastro usandoAWS SDK for Java

1. Creare un'istanza della classe `AmazonQLDB`. In alternativa, puoi utilizzare la stessa istanza del`AmazonQLDB` client che hai creato per la`CreateLedger` richiesta.
2. Creare un'istanza della`DescribeLedgerRequest` classe e specificare il nome della contabilità che si desidera descrivere.

3. Eseguire il metodo `describeLedger` fornendo l'oggetto della richiesta come parametro.
4. La`describeLedger` richiesta restituisce un`DescribeLedgerResult` oggetto che contiene informazioni correnti sul libro mastro.

Il seguente esempio di codice mostra le fasi precedenti. Puoi chiamare il`describeLedger` metodo del cliente per ottenere informazioni sul registro in qualsiasi momento.

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());
```

Descrivere un libro mastro (AWS CLI)

Descrizione della`vehicle-registration` contabilità appena creata.

Example

```
aws qldb describe-ledger --name vehicle-registration
```

Aggiornamento di un libro mastro

L'`UpdateLedger`operazione attualmente consente di modificare le seguenti impostazioni di configurazione per un libro contabile esistente:

- Protezione da eliminazione: il flag che impedisce a un utente qualsiasi di eliminare una contabilità. Se questa caratteristica è abilitata, è necessario innanzitutto disabilitarla impostando il flag su`false` prima di poter eliminare il libro mastro.

Se non si definisce questo parametro, non vengono apportate modifiche all'impostazione di protezione da eliminazione del libro mastro.

- AWS KMS key— La chiave inAWS Key Management Service (AWS KMS) da utilizzare per la crittografia dei dati inattivi. Se non si definisce questo parametro, non vengono apportate modifiche alla chiave KMS del libro mastro.

> **Note**
>
> Amazon QLDB ha lanciato il supporto per la gestioneAWS KMS keys dei clienti il 22 luglio 2021. Tutti i registri creati prima del lancio sono protetti per impostazioneChiavi di proprietà di AWS predefinita, ma al momento non sono idonei alla crittografia inattiva utilizzando chiavi gestite dal cliente.
> È possibile visualizzare l'ora di creazione del registro sulla console QLDB.

Utilizzare una delle opzioni seguenti:

- AWSchiave KMS di proprietà: usa una chiave KMS di proprietà e gestita da per tuoAWS conto. Per utilizzare questo tipo di chiave, specificate la stringa`AWS_OWNED_KMS_KEY` per questo parametro. Questa opzione non richiede alcuna configurazione aggiuntiva.
- Chiave KMS gestita dal cliente: usa una chiave KMS di crittografia simmetrica nel tuo account che crei, possiedi e gestisci. QLDB non supporta chiavi asimmetriche.

  Questa opzione richiede la creazione di una chiave KMS o l'utilizzo di una chiave esistente nel tuo account. Per istruzioni sulla creazione di una chiave gestita dal cliente, consulta Creazione di chiavi KMS con crittografia simmetrica nella Guida per gliAWS Key Management Service sviluppatori.

  Puoi specificare una chiave KMS gestita dal cliente utilizzando un ID, un alias o un nome di risorsa Amazon (ARN). Per ulteriori informazioni, consulta Identificatori chiave (KeyId) nella Guida per gliAWS Key Management Service sviluppatori.

  > **Note**
  >
  > Le chiavi tra Regioni non sono supportate. La chiave KMS specificata deve trovarsi nelloRegione AWS stesso libro mastro.

Le modifiche chiave in QLDB sono asincrone. Il libro mastro è completamente accessibile senza alcun impatto sulle prestazioni durante l'elaborazione della modifica chiave.

È possibile cambiare chiave tutte le volte che è necessario, ma il tempo necessario per aggiornare una chiave varia a seconda delle dimensioni del libro contabile. È possibile utilizzare l'`DescribeLedger`operazione per verificare lo stato di inattività della crittografia.

Per ulteriori informazioni, consulta Crittografia inattiva in Amazon QLDB.

Il formato dell'output da `UpdateLedger` è identico a quello da `CreateLedger`.

Aggiornamento di un libro mastro (Java)

Per aggiornare un libro contabile utilizzandoAWS SDK for Java

1. Creare un'istanza della classe `AmazonQLDB`.
2. Crea un'istanza della classe `UpdateLedgerRequest` per fornire le informazioni della richiesta.

   È necessario fornire il nome del libro contabile insieme a un nuovo valore booleano per la protezione dall'eliminazione o un nuovo valore di stringa per la chiave KMS.
3. Eseguire il metodo `updateLedger` fornendo l'oggetto della richiesta come parametro.

I seguenti esempi di codice illustrano i passaggi precedenti. La`updateLedger` richiesta restituisce un`UpdateLedgerResult` oggetto che contiene informazioni aggiornate sul libro mastro.

Example — Disabilita protezione da eliminazione

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withDeletionProtection(false);
UpdateLedgerResult result = client.updateLedger(request);
```

Example — Utilizzo di una chiave KMS gestita dal cliente

```
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 — Usa una chiave KMS diAWS proprietà

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withKmsKey("AWS_OWNED_KMS_KEY")
UpdateLedgerResult result = client.updateLedger(request);
```

Aggiornamento di un libro mastro (AWS CLI)

Se la`vehicle-registration` protezione dall'eliminazione è abilitata, è necessario innanzitutto disabilitarla prima di poterlo eliminare.

Example

```
aws qldb update-ledger --name vehicle-registration --no-deletion-protection
```

Puoi anche modificare le impostazioni di crittografia del registro a riposo per utilizzare una chiave KMS gestita dal cliente.

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

In alternativa, puoi modificare le impostazioni di crittografia a riposo per utilizzare una chiave KMS diAWS proprietà.

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key AWS_OWNED_KMS_KEY
```

Aggiornamento di una modalità di autorizzazione del registro

L'`UpdateLedgerPermissionsMode`operazione consente di modificare la modalità delle autorizzazioni di un libro mastro esistente. Seleziona una delle seguenti opzioni:

- Consenti tutto: una modalità di autorizzazione legacy che consente il controllo degli accessi con granularità a livello di API per i libri mastri.

  Questa modalità consente agli utenti che dispongono dell'autorizzazione API `SendCommand` per questo libro mastro per eseguire tutti i comandi PartiQL (quindi, `ALLOW_ALL`) su qualsiasi tabella nel libro mastro specificato. Questa modalità ignora tutte le policy di autorizzazione IAM a livello di tabella o di comando create per il libro mastro.

- Standard: (impostazione consigliata) una modalità di autorizzazione che consente il controllo degli accessi con una granularità più fine per libri mastri, tabelle e comandi PartiQL. Ti consigliamo di utilizzare questa modalità di autorizzazione per incrementare la sicurezza dei dati nel libro mastro.

  Per impostazione predefinita, questa modalità nega tutte le richieste di eseguire qualsiasi comando PartiQL su qualsiasi tabella in questo libro mastro. Per consentire i comandi PartiQL, devi creare le policy di autorizzazione IAM per risorse di tabelle e operazioni PartiQL specifiche, oltre all'autorizzazione`SendCommand` API per il libro mastro. Per informazioni, consulta Guida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

> **Important**
>
> Prima di passare alla modalità di`STANDARD` autorizzazione, devi prima creare tutte le politiche IAM e i tag di tabella necessari per evitare interruzioni per gli utenti. Per saperne di più, procedi aMigrazione alla modalità di autorizzazione standard.

Aggiornamento di una modalità di autorizzazione del registro (Java)

Per aggiornare la modalità di autorizzazione di un registro utilizzandoAWS SDK for Java

1. Creare un'istanza della classe `AmazonQLDB`.
2. Crea un'istanza della classe `UpdateLedgerPermissionsModeRequest` per fornire le informazioni della richiesta.

   È necessario fornire il nome del libro contabile insieme a un nuovo valore di stringa per la modalità autorizzazioni.
3. Eseguire il metodo `updateLedgerPermissionsMode` fornendo l'oggetto della richiesta come parametro.

I seguenti esempi di codice illustrano i passaggi precedenti. La`updateLedgerPermissionsMode` richiesta restituisce un`UpdateLedgerPermissionsModeResult` oggetto che contiene informazioni aggiornate sul libro mastro.

Example — Assegnazione della modalità di autorizzazione standard

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerPermissionsModeRequest request = new UpdateLedgerPermissionsModeRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
UpdateLedgerPermissionsModeResult result = client.updateLedgerPermissionsMode(request);
```

Aggiornamento di una modalità di autorizzazione del registro (AWS CLI)

Assegna la modalità`STANDARD` delle autorizzazioni al tuo`vehicle-registration` libro mastro.

Example

```
aws qldb update-ledger-permissions-mode --name vehicle-registration --permissions-mode
 STANDARD
```

Migrazione alla modalità di autorizzazione standard

Per passare alla modalità di`STANDARD` autorizzazione, ti consigliamo di analizzare i tuoi modelli di accesso QLDB e di aggiungere politiche IAM che concedano agli utenti le autorizzazioni appropriate per accedere alle loro risorse.

Prima di passare alla modalità di`STANDARD` autorizzazione, devi prima creare tutte le politiche IAM e i tag di tabella richiesti. In caso contrario, il passaggio dalla modalità delle autorizzazioni può disturbare gli utenti fino a quando non si creano le politiche IAM corrette o si ripristina la modalità delle autorizzazioni`ALLOW_ALL`. Per informazioni sulla creazione di queste policy, consultaGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

È inoltre possibile utilizzare una policyAWS gestita per garantire l'accesso completo a tutte le risorse QLDB. Le`AmazonQLDBFullAccess` politiche`AmazonQLDBConsoleFullAccess` gestite includono tutte le azioni QLDB, comprese tutte le azioni PartiQL. Collegare una di queste politiche a un principale equivale alla modalità di`ALLOW_ALL` autorizzazione per quel preside. Per ulteriori informazioni, consulta AWS politiche gestite per Amazon QLDB.

Eliminazione di un libro mastro

Utilizzate l'`DeleteLedger`operazione per eliminare un libro mastro e tutto il suo contenuto. L'eliminazione di un libro mastro è un'operazione irrecuperabile.

Se la protezione dall'eliminazione è abilitata per il libro mastro, è necessario innanzitutto disabilitarla prima di poter eliminare il libro mastro.

Quando si invia una`DeleteLedger` richiesta, lo stato del libro mastro cambia da`ACTIVE` a`DELETING`. L'eliminazione del libro mastro potrebbe richiedere del tempo, a seconda della quantità di spazio di archiviazione utilizzato. A conclusione dell'`DeleteLedger`operazione, il libro mastro non esiste più in QLDB.

Eliminazione di un libro mastro (Java)

Per eliminare un libro contabile utilizzandoAWS SDK for Java

1. Creare un'istanza della classe `AmazonQLDB`.
2. Creare un'istanza della`DeleteLedgerRequest` classe e specificare il nome della contabilità che si desidera eliminare.
3. Eseguire il metodo `deleteLedger` fornendo l'oggetto della richiesta come parametro.

Il seguente esempio di codice mostra le fasi precedenti.

Example

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
DeleteLedgerRequest request = new DeleteLedgerRequest().withName(ledgerName);
DeleteLedgerResult result = client.deleteLedger(request);
```

Eliminazione di un libro mastro (AWS CLI)

Elimina il tuo`vehicle-registration` libro mastro.

Example

```
aws qldb delete-ledger --name vehicle-registration
```

Registri di quotazione

L'`ListLedgers`operazione restituisce informazioni riepilogative di tutti i registri QLDB per l'attualeAccount AWS e la regione.

Registri di quotazione (Java)

Per elencare i libri contabili nel tuo account utilizzandoAWS SDK for Java

1. Creare un'istanza della classe `AmazonQLDB`.
2. Creare un'istanza della classe `ListLedgersRequest`.

   Se hai ricevuto un valore per`NextToken` nella risposta a una`ListLedgers` chiamata precedente, devi fornire quel valore in questa richiesta per ottenere la pagina successiva di risultati.
3. Eseguire il metodo `listLedgers` fornendo l'oggetto della richiesta come parametro.
4. La`listLedgers` richiesta restituisce un`ListLedgersResult` oggetto. Questo oggetto ha un elenco di`LedgerSummary` oggetti e un token di impaginazione che indica se sono disponibili altri risultati:

   - Se`NextToken` è vuota, l'ultima pagina dei risultati è stata elaborata e non ci sono altri risultati.
   - Se non`NextToken` è vuoto, sono disponibili altri risultati. Per recuperare la pagina successiva di risultati, usa il valore di`NextToken` in una`ListLedgers` chiamata successiva.

Il seguente esempio di codice mostra le fasi precedenti.

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);
```

Registri di quotazione (AWS CLI)

Elenca tutti i registri della correnteAccount AWS e della regione.

Example

```
aws qldb list-ledgers
```

Creazione di risorse Amazon QLDB conAWS CloudFormation

Amazon QLDB è integrato conAWS CloudFormation, un servizio che ti consente di modellare e configurareAWS le tue risorse in modo da dedicare meno tempo alla creazione e alla gestione delle risorse e dell'infrastruttura. Puoi creare un modello che descrive tutte leAWS risorse desiderate (come i registri QLDB) e si occuperà delAWS CloudFormation provisioning e della configurazione di queste risorse per tuo conto.

Quando usiAWS CloudFormation, puoi riutilizzare il modello per configurare le risorse QLDB in modo coerente e continuo. Basta descrivere le risorse una volta sola, dopodiché si può effettuare il provisioning di tali risorse quante volte si vuole in più Account AWS e regioni.

QLDB eAWS CloudFormation modelli

Per eseguire il provisioning e la configurazione delle risorse per QLDB e i servizi correlati, devi conoscere i AWS CloudFormationmodelli. I modelli sono file di testo formattati in JSON o YAML. Questi modelli descrivono le risorse di cui intendi effettuare il provisioning negli stack AWS CloudFormation. Se non hai familiarità con JSON o YAML, puoi usare AWS CloudFormation Designer per iniziare a utilizzare i modelli AWS CloudFormation. Per ulteriori informazioni, consulta Che cos'è AWS CloudFormation Designer? nella Guida per l'utente di AWS CloudFormation.

QLDB supporta la creazione di libri contabili e flussi di scritture contabili inAWS CloudFormation. Per ulteriori informazioni, inclusi esempi di modelli JSON e YAML per libri contabili e stream, consulta Riferimento dei tipi di risorse seguenti nella Guida per l'AWS CloudFormationutente:

- AWS: :QLDB: :QLDB: :QLDB
- AWS: :QLDB: :QLDB: :Stream

Ulteriori informazioni su AWS CloudFormation

Per ulteriori informazioni su AWS CloudFormation, consulta le seguenti risorse:

- AWS CloudFormation
- Guida per l'utente di AWS CloudFormation
- Documentazione di riferimento dell'API AWS CloudFormation
- Guida per l'utente dell'interfaccia a riga di comando di AWS CloudFormation

Assegnazione di tag alle risorse Amazon QLDB

Un tag è un'etichetta di attributi personalizzata assegnata dall'utente o da AWS a una risorsa AWS. Ogni tag è costituito da due parti:

- Una chiave del tag (ad esempio, `CostCenter`, `Environment` o `Project`). Le chiavi dei tag prevedono una distinzione tra lettere maiuscole e minuscole.
- Un campo facoltativo noto come valore del tag (ad esempio, `111122223333` o `Production`). Non specificare il valore del tag equivale a utilizzare una stringa vuota. Analogamente alle chiavi dei tag, i valori dei tag prevedono una distinzione tra lettere maiuscole e minuscole.

I tag consentono di eseguire le seguenti operazioni:

- Identificare e organizzare le risorse AWS. Molti Servizi AWS supportano l'assegnazione di tag, perciò è possibile assegnare lo stesso tag a risorse di diversi servizi per indicare che queste sono correlate. Ad esempio, puoi assegnare a un Contabilità Amazon QLDB lo stesso tag che si assegna a un bucket Amazon S3.
- Tenere traccia dei costi AWS. Questi tag vengono attivati nel pannello di controllo AWS Billing and Cost Management. AWS usa i tag per categorizzare i costi e fornire un report di allocazione dei costi mensili. Per ulteriori informazioni, consulta Utilizzo dei tag per l'allocazione dei costi nella Guida per l'utente di AWS Billing.
- Controlla l'accesso alle tueAWS risorse conAWS Identity and Access Management (IAM). Per informazioni, consultaControllo degli accessi basato sugli attributi (ABAC) con QLDB questa guida per sviluppatori e Controlla l'accesso tramite i tag IAM nella Guida per l'utente IAM.

Per suggerimenti sull'utilizzo dei tag, consulta il post AWS Tagging Strategies nel blog AWS Answers.

Nelle sezioni seguenti vengono fornite ulteriori informazioni sui tag per Amazon QLDB.

Argomenti

Risorse supportate in Amazon QLDB

Le seguenti risorse in Amazon QLDB supportano l'assegnazione di tag:

- libro mastro
- table
- flusso di diari

Per informazioni sull'aggiunta e la gestione dei tag, consulta Gestione dei tag.

Convenzioni di denominazione e utilizzo dei tag

Le seguenti convenzioni di base di denominazione e di utilizzo si applicano all'utilizzo dei tag con le risorse Amazon QLDB:

- Ogni risorsa può avere un massimo di 50 tag.
- Per ciascuna risorsa, ogni chiave del tag deve essere univoca e ogni chiave del tag può avere un solo valore.
- La lunghezza massima delle chiavi di tag è 128 caratteri Unicode in UTF-8.
- Il valore massimo dei tag è 256 caratteri Unicode in UTF-8.
- I caratteri consentiti sono lettere, numeri, spazi rappresentabili in formato UTF-8, oltre ai seguenti caratteri: . : + = @ _ / - (trattino).
- i valori e le chiavi dei tag rispettano la distinzione tra maiuscole e minuscole; Come best practice, è consigliabile definire una strategia per l'uso delle lettere maiuscole e minuscole nei tag e implementarla costantemente in tutti i tipi di risorse. Ad esempio, puoi decidere se utilizzare `Costcenter`, `costcenter` o `CostCenter` e utilizzare la stessa convenzione per tutti i tag. Non utilizzare tag simili con lettere maiuscole o minuscole incoerenti.
- Il prefisso `aws:` è riservato per l'uso di AWS. Non puoi modificare o eliminare la chiave o il valore di un tag quando il tag ha una chiave tag con il prefisso `aws:`. I tag con questo prefisso non vengono conteggiati per il limite del numero di tag per risorsa.

Gestione dei tag

I tag sono formati dalle proprietà `Key` e `Value` in una risorsa. Per aggiungere, modificare o eliminare i valori di tali proprietàAWS CLI, puoi anche usare la console Amazon QLDB, la o l'API QLDB. Puoi anche usare ilAWS Resource Groups Tag Editor per gestire i tag.

Per informazioni sull'utilizzo dei tag, consulta le seguenti operazioni API:

- ListTagsForResourcenella sezione di riferimento dell'API Amazon QLDB
- TagResourcenella sezione di riferimento dell'API Amazon QLDB
- UntagResourcenella sezione di riferimento dell'API Amazon QLDB

Per utilizzare il pannello di tagging QLDB (console)

1. Accedere e aprireAWS Management Console la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel riquadro di navigazione, scegliere Contabilità.
3. Nell'elenco dei libri contabili, scegli il nome del libro contabile di cui desideri gestire i tag.
4. Nella pagina dei dettagli del libro contabile, individua la scheda Tag e scegli Gestisci tag.
5. Nella pagina Gestisci i tag, puoi aggiungere, modificare o rimuovere qualsiasi tag appropriato per il tuo libro mastro. Dopo aver selezionato tutte le chiavi e i valori del tag, scegli Salva.

Assegnazione di tag alle risorse durante la creazione

Per le risorse QLDB che supportano l'etichettatura, puoi definire i tag durante la creazione della risorsa utilizzando l'AWS Management ConsoleAPIAWS CLI, la o QLDB. L'aggiunta di tag alle risorse in fase di creazione consente di evitare di eseguire script di tagging personalizzati dopo la creazione delle risorse.

Dopo aver taggato una risorsa, puoi controllare l'accesso alla risorsa in base a tali tag. Ad esempio, puoi concedere l'accesso completo solo alle risorse delle tabelle con un tag specifico. Per un esempio di policy JSON, vedereAccesso completo a tutte le azioni basate sui tag della tabella.

> **Note**
>
> Le risorse delle tabelle e dei flussi non ereditano i tag della rispettiva risorsa di registro principale.

È inoltre possibile definire i tag delle tabelle specificandoli in un'istruzione`CREATE TABLE` PartiQL. Per ulteriori informazioni, consulta Tabelle di etichettatura.

Sicurezza in Amazon QLDB

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gira Servizi AWS su Cloud AWS. AWS fornisce inoltre servizi che è possibile utilizzare in modo sicuro. I revisori di terze parti testano e verificano regolarmente l'efficacia della sicurezza come parte dei programmi di conformitàAWS. Per informazioni sui programmi di conformità applicabili ad Amazon QLDB, AWS consulta Services in Scope by Compliance Program.
- Sicurezza nel cloud: la tua responsabilità è determinata dal materiale Servizio AWS che utilizzi. Inoltre, sei responsabile anche di altri fattori, tra cui la riservatezza dei dati, i requisiti dell'azienda e le leggi e le normative applicabili.

Questa documentazione aiuta a capire come applicare il modello di responsabilità condivisa quando si utilizza QLDB. I seguenti argomenti mostrano come configurare QLDB per soddisfare i tuoi obiettivi di sicurezza e conformità. Imparerai anche a usarne altri Servizi AWS che ti aiutano a monitorare e proteggere le tue risorse QLDB.

Argomenti

Protezione dei dati in Amazon QLDB

Il modello di responsabilità AWS condivisa Modello si applica alla protezione dei dati in Amazon QLDB. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura

globale che gestisce tutti i. Cloud AWS L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, vedi le Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al Modello di responsabilità condivisa AWS e GDPR nel Blog sulla sicurezza AWS .

Ai fini della protezione dei dati, consigliamo di proteggere Account AWS le credenziali e configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Usa SSL/TLS per comunicare con le risorse. AWS È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con. AWS CloudTrail
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-2 per l'accesso AWS tramite un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-2.

Ti consigliamo vivamente di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori con QLDB o Servizi AWS altro utilizzando la console, l'API AWS CLI o gli SDK. AWS I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando fornisci un URL a un server esterno, ti suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta al server.

> **Note**
>
> Questa guida su come evitare tag o campi in formato libero per informazioni sensibili si riferisce ai metadati di una risorsa di registro QLDB, piuttosto che ai dati archiviati all'interno

del registro. I dati archiviati all'interno di una risorsa di registro QLDB non vengono utilizzati per i registri di fatturazione o diagnostica.

Argomenti

Crittografia inattiva in Amazon QLDB

Per impostazione predefinita, tutti i dati archiviati in Amazon QLDB sono completamente crittografati quando sono inattivi. La crittografia QLDB a riposo offre una maggiore sicurezza crittografando tutti i dati del registro inattivo utilizzando le chiavi di crittografia in (). AWS Key Management Service AWS KMS Questa funzionalità consente di ridurre gli oneri operativi e la complessità associati alla protezione dei dati sensibili. Con la crittografia a riposo, puoi creare applicazioni di registro sensibili alla sicurezza che soddisfano i rigorosi requisiti normativi e di conformità alla crittografia.

Encryption at rest si integra con AWS KMS la gestione della chiave di crittografia utilizzata per proteggere i registri QLDB. Per ulteriori informazioni in merito AWS KMS, consulta i AWS Key Management Service concetti nella Guida per gli sviluppatori.AWS Key Management Service

In QLDB, è possibile specificare il tipo di risorsa del registro per ogni AWS KMS key risorsa. Quando crei un nuovo libro contabile o aggiorni un libro mastro esistente, puoi scegliere uno dei seguenti tipi di chiavi KMS per proteggere i dati del registro:

- Chiave di proprietà di AWS— Il tipo di crittografia predefinito. La chiave è di proprietà di QLDB (senza costi aggiuntivi).
- Chiave gestita dal cliente: la chiave viene archiviata presso di te Account AWS e viene creata, posseduta e gestita da te. Hai il pieno controllo della chiave (a AWS KMS pagamento).

> **Note**
>
> Amazon QLDB ha lanciato il supporto per la AWS KMS keys gestione dei clienti il 22 luglio 2021. Tutti i registri creati prima del lancio sono protetti per impostazione Chiavi di proprietà di AWS predefinita, ma al momento non sono idonei per la crittografia a riposo utilizzando chiavi gestite dal cliente.

Puoi visualizzare l'ora di creazione del tuo libro mastro sulla console QLDB.

Quando si accede a un registro, QLDB decripta i dati in modo trasparente. È possibile passare dalla chiave gestita dal cliente a quella Chiave di proprietà di AWS gestita dal cliente in qualsiasi momento. Non è necessario modificare alcun codice o applicazione per utilizzare o gestire dati crittografati.

È possibile specificare una chiave di crittografia quando si crea un nuovo registro o si modifica la chiave di crittografia su un registro esistente utilizzando l' AWS Management Console API QLDB o (). AWS Command Line Interface AWS CLI Per ulteriori informazioni, consulta Utilizzo di chiavi gestite dal cliente in Amazon QLDB.

Note

Per impostazione predefinita, Amazon QLDB abilita automaticamente la crittografia a riposo senza costi Chiavi di proprietà di AWS aggiuntivi. Tuttavia, l'utilizzo di una chiave gestita dal cliente comporta dei AWS KMS costi. Per informazioni sui prezzi, consultare Prezzi di AWS Key Management Service.
La crittografia QLDB a riposo è disponibile Regioni AWS ovunque sia disponibile QLDB.

Argomenti

Crittografia a riposo: come funziona in Amazon QLDB

La crittografia QLDB a riposo crittografa i dati utilizzando l'Advanced Encryption Standard (AES-256) a 256 bit. Questo aiuta a proteggere i dati dall'accesso non autorizzato allo storage sottostante. Per impostazione predefinita, tutti i dati archiviati nei registri QLDB sono crittografati quando sono inattivi. La crittografia lato server è trasparente, il che significa che non sono necessarie modifiche alle applicazioni.

Encryption at rest si integra con AWS Key Management Service (AWS KMS) per la gestione della chiave di crittografia utilizzata per proteggere i registri QLDB. Quando si crea un nuovo registro o si aggiorna un registro esistente, è possibile scegliere uno dei seguenti tipi di chiavi: AWS KMS

- Chiave di proprietà di AWS— Il tipo di crittografia predefinito. La chiave è di proprietà di QLDB (senza costi aggiuntivi).
- Chiave gestita dal cliente: la chiave viene archiviata presso di te Account AWS e viene creata, posseduta e gestita da te. Hai il pieno controllo della chiave (a AWS KMS pagamento).

Argomenti

Chiave di proprietà di AWS

Chiavi di proprietà di AWS non sono archiviati nel tuo Account AWS. Fanno parte di una raccolta di chiavi KMS che AWS possiede e gestisce per essere utilizzate in più Account AWS lingue. Servizi AWS può essere utilizzato Chiavi di proprietà di AWS per proteggere i tuoi dati.

Non è necessario creare o gestire Chiavi di proprietà di AWS. Tuttavia, non puoi visualizzarne Chiavi di proprietà di AWS, tracciarne o controllarne l'utilizzo. Non ti viene addebitata una tariffa mensile o una tariffa di utilizzo per Chiavi di proprietà di AWS e non vengono conteggiate nelle AWS KMS quote del tuo account.

Per ulteriori informazioni, consulta la sezione Chiavi di proprietà di AWS nella Guida per gli sviluppatori di AWS Key Management Service .

Chiave gestita dal cliente

Le chiavi gestite dal cliente sono chiavi KMS Account AWS che crei, possiedi e gestisci. Hai il pieno controllo su queste chiavi KMS. QLDB supporta solo chiavi KMS con crittografia simmetrica.

Utilizza una chiave gestita dal cliente per ottenere le seguenti caratteristiche:

- Impostazione e gestione delle politiche chiave, delle politiche IAM e delle concessioni per il controllo dell'accesso alla chiave
- Abilitazione e disabilitazione della chiave
- Materiale crittografico rotante per la chiave

- Creazione di tag e alias chiave
- Pianificazione della chiave per l'eliminazione
- Importazione del materiale chiave personalizzato o utilizzo di un archivio di chiavi personalizzato di tua proprietà e gestione
- Utilizzo AWS CloudTrail di Amazon CloudWatch Logs per tenere traccia delle richieste a cui QLDB invia per tuo conto AWS KMS

Per ulteriori informazioni, consulta Customer managed keys nella Guida per sviluppatori AWS Key Management Service .

Le chiavi gestite dal cliente comportano un addebito per ogni chiamata API e a queste chiavi KMS si applicano AWS KMS delle quote. Per ulteriori informazioni, consulta le AWS KMS risorse o richiedi quote.

Quando si specifica una chiave gestita dal cliente come chiave KMS per un registro, tutti i dati contabili presenti sia nell'archiviazione delle scritture contabili che nell'archiviazione indicizzata vengono protetti con la stessa chiave gestita dal cliente.

Chiavi gestite dal cliente inaccessibili

Se disabiliti la chiave gestita dal cliente, pianifichi l'eliminazione della chiave o revochi le concessioni sulla chiave, lo stato della crittografia del registro diventa. `KMS_KEY_INACCESSIBLE` In questo stato, il registro è danneggiato e non accetta richieste di lettura o scrittura. Una chiave inaccessibile impedisce a tutti gli utenti e al servizio QLDB di crittografare o decrittografare i dati e di eseguire operazioni di lettura e scrittura nel registro. QLDB deve avere accesso alla tua chiave KMS per assicurarti di poter continuare ad accedere al tuo registro e prevenire la perdita di dati.

> **Important**
>
> Un registro danneggiato torna automaticamente allo stato attivo dopo aver ripristinato le autorizzazioni sulla chiave o dopo aver riattivato la chiave che era disabilitata.
> Tuttavia, l'eliminazione di una chiave gestita dal cliente è irreversibile. Dopo l'eliminazione di una chiave, non è più possibile accedere ai registri protetti con tale chiave e i dati diventano irrecuperabili in modo permanente.

Per verificare lo stato di crittografia di un registro, utilizza l'operazione o l' AWS Management Console API. DescribeLedger

In che modo Amazon QLDB utilizza le sovvenzioni in AWS KMS

QLDB richiede sovvenzioni per utilizzare la chiave gestita dal cliente. Quando crei un registro protetto con una chiave gestita dal cliente, QLDB crea sovvenzioni per tuo conto inviando richieste a. CreateGrant AWS KMS Le sovvenzioni AWS KMS vengono utilizzate per consentire a QLDB di accedere a una chiave KMS in un cliente. Account AWS Per ulteriori informazioni sulle autorizzazioni, consulta Utilizzo delle concessioni nella Guida per gli sviluppatori di AWS Key Management Service .

QLDB richiede le sovvenzioni per utilizzare la chiave gestita dal cliente per le seguenti operazioni: AWS KMS

- DescribeKey— Verifica che la chiave KMS di crittografia simmetrica specificata sia valida.
- GenerateDataKey— Genera una chiave dati simmetrica unica che QLDB utilizza per crittografare i dati inattivi nel registro.
- Decrittografa: decrittografa la chiave dati che è stata crittografata dalla chiave gestita dal cliente.
- Crittografa: crittografa il testo non crittografato in testo cifrato utilizzando la chiave gestita dal cliente.

Puoi revocare una concessione per rimuovere l'accesso del servizio alla chiave gestita dal cliente in qualsiasi momento. In tal caso, la chiave diventa inaccessibile e QLDB perde l'accesso a tutti i dati del registro protetti dalla chiave gestita dal cliente. In questo stato, il registro è danneggiato e non accetta alcuna richiesta di lettura o scrittura finché non ripristini le autorizzazioni sulla chiave.

Ripristino delle sovvenzioni in AWS KMS

Per ripristinare le concessioni su una chiave gestita dal cliente e ripristinare l'accesso a un registro in QLDB, puoi aggiornare il registro e specificare la stessa chiave KMS. Per istruzioni, consulta Aggiornamento AWS KMS key di un libro mastro esistente.

Considerazioni sulla crittografia a riposo

Considera quanto segue quando utilizzi la crittografia a riposo in QLDB:

- La crittografia lato server a riposo è abilitata per impostazione predefinita su tutti i dati del registro QLDB e non può essere disabilitata. Non è possibile crittografare solo un sottoinsieme di dati in un registro.
- La crittografia dei dati inattivi crittografa i dati mentre è statica (dati inattivi) su media di storage persistente. Se la sicurezza dei dati è un problema per i dati in transito o per i dati in uso, potrebbe essere necessario adottare le seguenti misure aggiuntive:

- Dati in transito: tutti i tuoi dati in QLDB sono crittografati durante il transito. Per impostazione predefinita, le comunicazioni da e verso QLDB utilizzano il protocollo HTTPS, che protegge il traffico di rete utilizzando la crittografia Secure Sockets Layer (SSL) /Transport Layer Security (TLS).
- Dati in uso: proteggi i tuoi dati prima di inviarli a QLDB utilizzando la crittografia lato client.

Per scoprire come implementare le chiavi gestite dai clienti per i registri, procedi a. Utilizzo di chiavi gestite dal cliente in Amazon QLDB

Utilizzo di chiavi gestite dal cliente in Amazon QLDB

Puoi utilizzare l'API AWS Management Console, the AWS Command Line Interface (AWS CLI) o QLDB per specificare AWS KMS key i registri nuovi e quelli esistenti in Amazon QLDB. I seguenti argomenti descrivono come gestire e monitorare l'utilizzo delle chiavi gestite dai clienti in QLDB.

Argomenti

Prerequisiti

Prima di poter proteggere un registro QLDB con una chiave gestita dal cliente, devi prima creare la chiave in (). AWS Key Management Service AWS KMSÈ inoltre necessario specificare una politica chiave che consenta a QLDB di creare sovvenzioni in tal senso per conto AWS KMS key dell'utente.

Creazione di una chiave gestita dal cliente

Per creare una chiave gestita dal cliente, segui i passaggi descritti in Creazione di chiavi KMS con crittografia simmetrica nella Guida per gli AWS Key Management Service sviluppatori. QLDB non supporta le chiavi asimmetriche.

Impostazione di una policy delle chiavi

Le policy chiave sono il modo principale per controllare l'accesso alle chiavi gestite dai clienti. AWS KMS Ogni chiave gestita dal cliente deve avere esattamente una politica chiave. Le

istruzioni nel documento di policy delle chiavi determina chi è autorizzato a utilizzare la chiave KMS e come. Per ulteriori informazioni, vedere Utilizzo delle politiche chiave in AWS KMS.

Puoi specificare una politica chiave quando crei la tua chiave gestita dai clienti. Per modificare una politica chiave per una chiave gestita dal cliente esistente, vedi Modifica di una politica chiave.

Per consentire a QLDB di utilizzare la chiave gestita dal cliente, la policy chiave deve includere le autorizzazioni per le seguenti azioni: AWS KMS

- kms: CreateGrant — Aggiunge una concessione a una chiave gestita dal cliente. Concede il controllo dell'accesso a una chiave KMS specificata.

  Quando si crea o si aggiorna un registro con una chiave gestita dal cliente specificata, QLDB crea concessioni che consentono l'accesso alle operazioni di concessione richieste. Le operazioni di concessione includono quanto segue:

  - GenerateDataKey
  - Decrypt
  - Encrypt
- kms: DescribeKey — Restituisce informazioni dettagliate su una chiave gestita dal cliente. QLDB utilizza queste informazioni per convalidare la chiave.

Esempio di politica chiave

Di seguito è riportato un esempio di policy chiave che è possibile utilizzare per QLDB. Questa politica consente ai responsabili autorizzati a utilizzare QLDB dall'111122223333account di richiamare le operazioni `CreateGrant` e `DescribeKey` sulla risorsa. `arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`

Per utilizzare questa politica, sostituisci *`us-east-1`*, 111122223333 *e* *`1234abcd-12ab-34cd-56ef-1234567890ab`* nell'esempio con le tue informazioni.

```
{
    "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"
                }
            }
        }
    ]
}
```

AWS KMS key Specificare il per un nuovo libro contabile

Segui questi passaggi per specificare una chiave KMS quando crei un nuovo libro mastro utilizzando la console QLDB o il. AWS CLI

Puoi specificare una chiave gestita dal cliente utilizzando un ID, un alias o Amazon Resource Name (ARN). Per ulteriori informazioni, consulta Key identifiers (KeyId) nella Developer Guide.AWS Key Management Service

> **Note**
>
> Le chiavi interregionali non sono supportate. La chiave KMS specificata deve trovarsi nella Regione AWS stessa cartella del registro.

Creazione di un libro mastro (console)

1. Accedi a e apri AWS Management Console la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Scegli Create Ledger.
3. Nella pagina Crea libro contabile, procedi come segue:
   - Informazioni sul libro contabile: immettere un nome di libro contabile univoco tra tutti i libri contabili dell'area corrente e di quella regione. Account AWS

- Modalità autorizzazioni: scegli una modalità di autorizzazione da assegnare al registro:
  - Consenti tutto
  - Standard (consigliato)
- Crittografa i dati inattivi: scegli il tipo di chiave KMS da utilizzare per la crittografia a riposo:
  - Usa una chiave KMS di AWS proprietà: utilizza una chiave KMS di proprietà e gestita da per AWS tuo conto. Questa è l'opzione predefinita e non richiede alcuna configurazione aggiuntiva.
  - Scegli una AWS KMS chiave diversa: utilizza una chiave KMS con crittografia simmetrica nel tuo account che crei, possiedi e gestisci.

    Per creare una nuova chiave utilizzando la AWS KMS console, scegli Crea una chiave. AWS KMS Per ulteriori informazioni, consulta Creazione di chiavi KMS di crittografia simmetrica nella Guida per gli sviluppatori di AWS Key Management Service .

    Per utilizzare una chiave KMS esistente, scegline una dall'elenco a discesa o specifica un ARN per la chiave KMS.

4. Quando le impostazioni sono quelle che desideri, scegli Crea registro.

   Puoi accedere al registro QLDB quando il suo stato diventa Attivo. Questo processo può richiedere diversi minuti.

Creazione di un libro mastro ()AWS CLI

Utilizzare il AWS CLI per creare un registro in QLDB con la chiave Chiave di proprietà di AWS predefinita o gestita dal cliente.

Example — Per creare un libro mastro con il valore predefinito Chiave di proprietà di AWS

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

Example — Per creare un libro mastro con una chiave gestita dal cliente

```
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
```

Aggiornamento AWS KMS key di un libro mastro esistente

Puoi anche utilizzare la console QLDB o aggiornare AWS CLI la chiave KMS di un registro esistente con una chiave gestita dal cliente in qualsiasi momento. Chiave di proprietà di AWS

> **Note**
>
> Amazon QLDB ha lanciato il supporto per la AWS KMS keys gestione dei clienti il 22 luglio 2021. Tutti i registri creati prima del lancio sono protetti per impostazione Chiavi di proprietà di AWS predefinita, ma al momento non sono idonei per la crittografia a riposo utilizzando chiavi gestite dal cliente.
> Puoi visualizzare l'ora di creazione del tuo libro mastro sulla console QLDB.

Le modifiche principali in QLDB sono asincrone. Il registro è completamente accessibile senza alcun impatto sulle prestazioni durante l'elaborazione della modifica chiave. Il tempo necessario per aggiornare una chiave varia a seconda delle dimensioni del registro.

Puoi specificare una chiave gestita dal cliente utilizzando un ID, un alias o Amazon Resource Name (ARN). Per ulteriori informazioni, consulta Key identifiers (KeyId) nella Developer Guide.AWS Key Management Service

> **Note**
>
> Le chiavi interregionali non sono supportate. La chiave KMS specificata deve trovarsi nella Regione AWS stessa cartella del registro.

Aggiornamento di un libro mastro (console)

1. Accedi a e apri AWS Management Console la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel pannello di navigazione, scegli Ledgers.
3. Nell'elenco dei libri contabili, seleziona il libro contabile che desideri aggiornare, quindi scegli Modifica libro contabile.
4. Nella pagina Modifica libro contabile, scegli il tipo di chiave KMS da usare per la crittografia a riposo:

- Usa una chiave KMS di AWS proprietà: utilizza una chiave KMS di proprietà e gestita da per AWS tuo conto. Questa è l'opzione predefinita e non richiede alcuna configurazione aggiuntiva.
- Scegli una AWS KMS chiave diversa: utilizza una chiave KMS con crittografia simmetrica nel tuo account che crei, possiedi e gestisci.

  Per creare una nuova chiave utilizzando la AWS KMS console, scegli Crea una chiave. AWS KMS Per ulteriori informazioni, consulta Creazione di chiavi KMS di crittografia simmetrica nella Guida per gli sviluppatori di AWS Key Management Service .

  Per utilizzare una chiave KMS esistente, scegline una dall'elenco a discesa o specifica un ARN per la chiave KMS.

5. Scegli Conferma modifiche.

Aggiornamento di un registro ()AWS CLI

Utilizzare il AWS CLI per aggiornare un registro esistente in QLDB con la chiave Chiave di proprietà di AWS predefinita o gestita dal cliente.

Example — Per aggiornare un registro con quello predefinito Chiave di proprietà di AWS

```
aws qldb update-ledger --name my-example-ledger --kms-key AWS_OWNED_KMS_KEY
```

Example — Aggiornare un registro con una chiave gestita dal cliente

```
aws qldb update-ledger \
    --name my-example-ledger \
    --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Monitoraggio di AWS KMS keys

Se utilizzi una chiave gestita dal cliente per proteggere i tuoi registri Amazon QLDB, puoi utilizzare AWS CloudTrail CloudWatch Amazon Logs per tenere traccia delle richieste a cui QLDB invia per tuo conto. AWS KMS Per ulteriori informazioni, consulta Monitoring nella Developer Guide. AWS KMS keysAWS Key Management Service

Gli esempi seguenti sono le voci di CloudTrail registro relative alle operazioni `CreateGrant`GenerateDataKey,`Decrypt`,`Encrypt`, e`DescribeKey`.

CreateGrant

Quando specifichi una chiave gestita dal cliente per proteggere il tuo registro, QLDB `CreateGrant` invia richieste AWS KMS a tuo nome per consentire l'accesso alla tua chiave KMS. Inoltre, QLDB utilizza `RetireGrant` l'operazione per rimuovere le concessioni quando si elimina un registro.

Le sovvenzioni create da QLDB sono specifiche di un registro. Il principale nella `CreateGrant` richiesta è l'utente che ha creato la tabella.

L'evento che registra l'operazione `CreateGrant` è simile a quello del seguente evento di esempio. I parametri includono l'Amazon Resource Name (ARN) della chiave gestita dal cliente, il titolare del beneficiario e il destinatario del ritiro (il servizio QLDB) e le operazioni coperte dalla sovvenzione.

```
{
    "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

Quando si specifica una chiave gestita dal cliente per proteggere il registro, QLDB crea una chiave dati unica. Invia una `GenerateDataKey` richiesta a AWS KMS cui specifica la chiave gestita dal cliente per il registro.

L'evento che registra l'operazione `GenerateDataKey` è simile a quello del seguente evento di esempio. L'utente è l'account del servizio QLDB. I parametri includono l'ARN della chiave gestita

dal cliente, un identificatore di chiave dati che richiede una lunghezza di 32 byte e il contesto di crittografia che identifica il nodo della gerarchia delle chiavi interna.


```
{
    "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

Quando si accede a un registro, QLDB richiama `Decrypt` l'operazione per decrittografare la chiave dati memorizzata nel registro in modo che possa accedere ai dati crittografati nel registro.

L'evento che registra l'operazione `Decrypt` è simile a quello del seguente evento di esempio. L'utente è l'account del servizio QLDB. I parametri includono l'ARN della chiave gestita dal cliente e il contesto di crittografia che identifica il nodo della gerarchia delle chiavi interna.

```
{
    "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"
}
```

Crittografa

QLDB richiama `Encrypt` l'operazione per crittografare il testo non crittografato in testo cifrato utilizzando la chiave gestita dal cliente.

L'evento che registra l'operazione `Encrypt` è simile a quello del seguente evento di esempio. L'utente è l'account del servizio QLDB. I parametri includono l'ARN della chiave gestita dal cliente e il contesto di crittografia che specifica l'ID univoco interno del registro.

```
{
    "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 richiama `DescribeKey` l'operazione per determinare se la chiave KMS specificata esiste nella regione and. Account AWS

L'evento che registra l'operazione `DescribeKey` è simile a quello del seguente evento di esempio. Il principale è l'utente del tuo account Account AWS che ha specificato la chiave KMS. I parametri includono l'ARN della chiave gestita dal cliente.

```
{
    "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"
}
```

Crittografia in transito in Amazon QLDB

Amazon QLDB accetta solo connessioni sicure che utilizzano il protocollo HTTPS, che protegge il traffico di rete utilizzando Secure Sockets Layer (SSL) /Transport Layer Security (TLS). La crittografia in transito fornisce un ulteriore livello di protezione dei dati crittografando i dati mentre viaggiano da e verso QLDB. Le politiche organizzative, le normative di settore o governative e i requisiti di conformità spesso richiedono l'uso della crittografia in transito per aumentare la sicurezza dei dati delle applicazioni quando trasmettono dati sulla rete.

QLDB offre anche endpoint FIPS in regioni selezionate. A differenza degli endpoint AWS standard, gli endpoint FIPS utilizzano una libreria software TLS conforme agli standard FIPS (Federal Information Processing Standard) 140-2. Questi endpoint potrebbero essere necessari ad aziende

che interagiscono con il governo degli Stati Uniti. Per ulteriori informazioni, consulta Endpoint FIPS in. Riferimenti generali di AWS Per un elenco completo delle regioni e degli endpoint disponibili per QLDB, consulta Endpoint e quote Amazon QLDB.

Identity and Access Management per Amazon QLDB

AWS Identity and Access Management (IAM) è un software Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle risorse. AWS Gli amministratori IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (dispone delle autorizzazioni) a utilizzare le risorse QLDB. IAM è uno strumento Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Argomenti

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia a seconda del lavoro svolto in QLDB.

Utente del servizio: se utilizzi il servizio QLDB per svolgere il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. Man mano che utilizzi più funzionalità QLDB per svolgere il tuo lavoro, potresti aver bisogno di autorizzazioni aggiuntive. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non è possibile accedere a una funzionalità in QLDB, vedere. Risoluzione dei problemi relativi all'identità e all'accesso ad Amazon QLDB

Amministratore del servizio: se sei responsabile delle risorse QLDB della tua azienda, probabilmente hai pieno accesso a QLDB. È tuo compito determinare a quali funzionalità e risorse QLDB devono accedere gli utenti del servizio. Devi inviare le richieste all'amministratore IAM per cambiare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per ulteriori informazioni su come la tua azienda può utilizzare IAM con QLDB, consulta. Come funziona Amazon QLDB con IAM

Amministratore IAM: se sei un amministratore IAM, potresti voler conoscere i dettagli su come scrivere policy per gestire l'accesso a QLDB. Per visualizzare esempi di policy basate sull'identità QLDB che puoi utilizzare in IAM, consulta. Esempi di policy basate sull'identità per Amazon QLDB

Autenticazione con identità

L'autenticazione è il modo in cui accedi utilizzando le tue credenziali di identità. AWS Devi essere autenticato (aver effettuato l' Utente root dell'account AWS accesso AWS) come utente IAM o assumendo un ruolo IAM.

Puoi accedere AWS come identità federata utilizzando le credenziali fornite tramite una fonte di identità. AWS IAM Identity Center Gli utenti (IAM Identity Center), l'autenticazione Single Sign-On della tua azienda e le tue credenziali di Google o Facebook sono esempi di identità federate. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Quando accedi AWS utilizzando la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere al AWS Management Console o al portale di AWS accesso. Per ulteriori informazioni sull'accesso a AWS, vedi Come accedere al tuo Account AWS nella Guida per l'Accedi ad AWS utente.

Se accedi a AWS livello di codice, AWS fornisce un kit di sviluppo software (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le tue richieste utilizzando le tue credenziali. Se non utilizzi AWS strumenti, devi firmare tu stesso le richieste. Per ulteriori informazioni sull'utilizzo del metodo consigliato per firmare autonomamente le richieste, consulta Signing AWS API request nella IAM User Guide.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, ti AWS consiglia di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza del tuo account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'utente di AWS IAM Identity Center e Utilizzo dell'autenticazione a più fattori (MFA) in AWS nella Guida per l'utente di IAM.

Account AWS utente root

Quando si crea un account Account AWS, si inizia con un'identità di accesso che ha accesso completo a tutte Servizi AWS le risorse dell'account. Questa identità è denominata utente Account AWS root ed è accessibile effettuando l'accesso con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conservare le credenziali dell'utente root e utilizzarle per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione Attività che richiedono le credenziali dell'utente root nella Guida per l'utente di IAM.

Identità federata

Come procedura consigliata, richiedi agli utenti umani, compresi gli utenti che richiedono l'accesso come amministratore, di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente dell'elenco utenti aziendale, di un provider di identità Web AWS Directory Service, della directory Identity Center o di qualsiasi utente che accede utilizzando le Servizi AWS credenziali fornite tramite un'origine di identità. Quando le identità federate accedono Account AWS, assumono ruoli e i ruoli forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo di utilizzare AWS IAM Identity Center. Puoi creare utenti e gruppi in IAM Identity Center oppure puoi connetterti e sincronizzarti con un set di utenti e gruppi nella tua fonte di identità per utilizzarli su tutte le tue applicazioni. Account AWS Per ulteriori informazioni sul Centro identità IAM, consulta Cos'è Centro identità IAM? nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un utente IAM è un'identità interna Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, per casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente di IAM.

Un gruppo IAM è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti

alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, è possibile avere un gruppo denominato IAMAdmins e concedere a tale gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Quando creare un utente IAM (invece di un ruolo) nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. Puoi assumere temporaneamente un ruolo IAM in AWS Management Console cambiando ruolo. Puoi assumere un ruolo chiamando un'operazione AWS CLI o AWS API o utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente di IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Accesso utente federato: per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta Creazione di un ruolo per un provider di identità di terza parte nella Guida per l'utente di IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per ulteriori informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .
- Autorizzazioni utente IAM temporanee: un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, con alcuni Servizi AWS, è possibile allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.

- Accesso a più servizi: alcuni Servizi AWS utilizzano le funzionalità di altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è comune che tale servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
  - Sessioni di accesso diretto (FAS): quando utilizzi un utente o un ruolo IAM per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FAS utilizza le autorizzazioni del principale che chiama an Servizio AWS, combinate con la richiesta Servizio AWS per effettuare richieste ai servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Forward access sessions.
  - Ruolo di servizio: un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire azioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.
  - Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'operazione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.
- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un ruolo IAM per gestire le credenziali temporanee per le applicazioni in esecuzione su un'istanza EC2 e che AWS CLI effettuano richieste API. AWS Ciò è preferibile all'archiviazione delle chiavi di accesso nell'istanza EC2. Per assegnare un AWS ruolo a un'istanza EC2 e renderlo disponibile per tutte le sue applicazioni, crei un profilo di istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta Utilizzo di un ruolo IAM per concedere autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2 nella Guida per l'utente di IAM.

Per informazioni sull'utilizzo dei ruoli IAM, consulta Quando creare un ruolo IAM (invece di un utente) nella Guida per l'utente di IAM.

Gestione dell'accesso con policy

Puoi controllare l'accesso AWS creando policy e collegandole a AWS identità o risorse. Una policy è un oggetto AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un principale (utente, utente root o sessione di ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente di IAM.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. Successivamente l'amministratore può aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'azione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'azione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dall' AWS Management Console AWS CLI, dall'o dall' AWS API.

Policy basate su identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruoli IAM). Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono integrate direttamente in un singolo utente, gruppo o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più utenti, gruppi e ruoli nel tuo Account AWS. Le politiche gestite includono politiche AWS gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scelta fra policy gestite e policy inline nella Guida per l'utente di IAM.

Policy basate su risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarle per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non puoi utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Liste di controllo degli accessi (ACL)

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni per accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3 e Amazon VPC sono esempi di servizi che supportano gli ACL. AWS WAF Per maggiori informazioni sulle ACL, consulta Panoramica delle liste di controllo degli accessi (ACL) nella Guida per gli sviluppatori di Amazon Simple Storage Service.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite delle autorizzazioni è una funzione avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Politiche di controllo dei servizi (SCP): le SCP sono politiche JSON che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in. AWS Organizations AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più Account

AWS di proprietà dell'azienda. Se abiliti tutte le funzionalità in un'organizzazione, puoi applicare le policy di controllo dei servizi (SCP) a uno o tutti i tuoi account. L'SCP limita le autorizzazioni per le entità presenti negli account dei membri, inclusa ciascuna. Utente root dell'account AWS Per ulteriori informazioni su organizzazioni e policy SCP, consulta la pagina sulle Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations .

- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta Policy di sessione nella Guida per l'utente di IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella IAM User Guide.

Come funziona Amazon QLDB con IAM

Prima di utilizzare IAM per gestire l'accesso a QLDB, scopri quali funzionalità IAM sono disponibili per l'uso con QLDB.

Funzionalità IAM che puoi usare con Amazon QLDB

Funzionalità IAM	Supporto QLDB
Policy basate su identità	Sì
Policy basate su risorse	No
Azioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione delle policy	Sì

Funzionalità IAM	Supporto QLDB
Liste di controllo degli accessi (ACL)	No
ABAC (tag nelle policy)	Sì
Credenziali temporanee	Sì
Autorizzazioni del principale	No
Ruoli di servizio	Sì
Ruoli collegati al servizio	No

Per avere una panoramica di alto livello su come QLDB e Servizi AWS altri funzionano con la maggior parte delle funzionalità IAM, Servizi AWS consulta la sezione dedicata all'utilizzo di IAM nella IAM User Guide.

Politiche basate sull'identità per QLDB

Supporta le policy basate su identità	Sì

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Con le policy basate su identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Non è possibile specificare l'entità principale in una policy basata sull'identità perché si applica all'utente o al ruolo a cui è associato. Per informazioni su tutti gli elementi utilizzabili in una policy JSON, consulta Guida di riferimento agli elementi delle policy JSON IAM nella Guida per l'utente di IAM.

Esempi di policy basate sull'identità per QLDB

Per visualizzare esempi di policy basate sull'identità QLDB, vedere. Esempi di policy basate sull'identità per Amazon QLDB

Politiche basate sulle risorse all'interno di QLDB

Supporta le policy basate su risorse	No

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarle per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Per consentire l'accesso multi-account, puoi specificare un intero account o entità IAM in un altro account come principale in una policy basata sulle risorse. L'aggiunta di un principale multi-account a una policy basata sulle risorse rappresenta solo una parte della relazione di trust. Quando il principale e la risorsa sono diversi Account AWS, un amministratore IAM dell'account affidabile deve inoltre concedere all'entità principale (utente o ruolo) l'autorizzazione ad accedere alla risorsa. L'autorizzazione viene concessa collegando all'entità una policy basata sull'identità. Tuttavia, se una policy basata su risorse concede l'accesso a un principale nello stesso account, non sono richieste ulteriori policy basate su identità. Per ulteriori informazioni, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.

Azioni politiche per QLDB

Supporta le operazioni di policy	Sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso a un criterio. Le azioni politiche in genere hanno lo stesso nome dell'operazione AWS API associata. Ci sono alcune eccezioni, ad esempio le azioni di sola autorizzazione che non hanno un'operazione API corrispondente. Esistono anche alcune operazioni che richiedono più operazioni in una policy. Queste operazioni aggiuntive sono denominate operazioni dipendenti.

Includi le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Per visualizzare un elenco di azioni QLDB, consulta Azioni definite da Amazon QLDB nel Service Authorization Reference.

Le azioni politiche in QLDB utilizzano il seguente prefisso prima dell'azione:

```
qldb
```

Per specificare più operazioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [
      "qldb:action1",
      "qldb:action2"
]
```

È possibile specificare più azioni tramite caratteri jolly (*). Ad esempio, per specificare tutte le azioni che iniziano con la parola `Describe`, includi la seguente azione:

```
"Action": "qldb:Describe*"
```

Per interagire con l'API dei dati transazionali QLDB (sessione QLDB) eseguendo istruzioni PartiQL su un registro, è necessario concedere l'autorizzazione all'azione come segue. `SendCommand`

```
"Action": "qldb:SendCommand"
```

Per i registri in modalità `STANDARD` autorizzazioni, vedere le autorizzazioni aggiuntive richieste Riferimento alle autorizzazioni PartiQL per ogni comando PartiQL.

Per visualizzare esempi di policy basate sull'identità QLDB, vedere. Esempi di policy basate sull'identità per Amazon QLDB

Risorse politiche per QLDB

Supporta le risorse di policy	Sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'azione. Le istruzioni devono includere un elemento `Resource` o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Puoi eseguire questa operazione per azioni che supportano un tipo di risorsa specifico, note come autorizzazioni a livello di risorsa.

Per le azioni che non supportano le autorizzazioni a livello di risorsa, ad esempio le operazioni di elenco, utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

Per visualizzare un elenco dei tipi di risorse QLDB e dei relativi ARN, consulta Resources defined by Amazon QLDB nel Service Authorization Reference. Per sapere con quali azioni puoi specificare l'ARN di ogni risorsa, consulta Azioni definite da Amazon QLDB.

In QLDB, le risorse principali sono i registri. QLDB supporta anche tipi di risorse aggiuntivi: tabelle e stream. Tuttavia, è possibile creare tabelle e stream solo nel contesto di un registro esistente.

Una tabella QLDB è una visualizzazione materializzata di una raccolta non ordinata di revisioni di documenti dal giornale contabile. Nella modalità `STANDARD` autorizzazioni di un registro, è necessario creare politiche IAM che concedano le autorizzazioni per eseguire istruzioni PartiQL su questa risorsa della tabella. Con le autorizzazioni su una risorsa della tabella, puoi eseguire istruzioni che accedono allo stato corrente della tabella. È inoltre possibile interrogare la cronologia delle revisioni della tabella utilizzando la funzione integrata`history()`. Per ulteriori informazioni, consulta Guida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

> **Note**
>
> L'`CREATE TABLE`istruzione crea una tabella con un ID univoco e il nome della tabella fornito. Il nome della tabella fornito deve essere univoco tra tutte le tabelle attive. Tuttavia, QLDB consente di disattivare le tabelle, quindi potrebbero esserci più tabelle inattive che condividono lo stesso nome di tabella. Pertanto, gli ARN delle risorse della tabella si riferiscono all'ID univoco assegnato dal sistema anziché al nome della tabella definito dall'utente.

Ogni libro mastro fornisce anche una risorsa di catalogo definita dal sistema su cui è possibile interrogare per elencare tutte le tabelle e gli indici in un registro. Per ulteriori informazioni sul modello di oggetti dati QLDB, vedere. Concetti e terminologia fondamentali in Amazon QLDB

A queste risorse sono associati ARN univoci, come illustrato nella tabella seguente.

Tipo di risorsa	ARN
ledger	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}`
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}`

Ad esempio, per specificare la `myExampleLedger` risorsa nella dichiarazione, utilizzare il seguente ARN.

```
"Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
```

Per specificare più risorse in una singola istruzione, separa gli ARN con le virgole.

```
"Resource": [
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger1",
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger2"
]
```

Per visualizzare esempi di policy basate sull'identità QLDB, vedere. Esempi di policy basate sull'identità per Amazon QLDB

Chiavi relative alle condizioni delle policy per QLDB

Supporta le chiavi di condizione delle policy specifiche del servizio	Sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Condition` (o blocco `Condition`) consente di specificare le condizioni in cui un'istruzione è in vigore. L'elemento `Condition` è facoltativo. Puoi compilare espressioni condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta.

Se specifichi più elementi `Condition` in un'istruzione o più chiavi in un singolo elemento `Condition`, questi vengono valutati da AWS utilizzando un'operazione `AND` logica. Se si specificano più valori per una singola chiave di condizione, AWS valuta la condizione utilizzando un'operazione logica. `OR` Tutte le condizioni devono essere soddisfatte prima che le autorizzazioni dell'istruzione vengano concesse.

Puoi anche utilizzare variabili segnaposto quando specifichi le condizioni. Ad esempio, puoi autorizzare un utente IAM ad accedere a una risorsa solo se è stata taggata con il relativo nome utente IAM. Per ulteriori informazioni, consulta Elementi delle policy IAM: variabili e tag nella Guida per l'utente di IAM.

AWS supporta chiavi di condizione globali e chiavi di condizione specifiche del servizio. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'utente IAM.

Per visualizzare un elenco di chiavi di condizione QLDB, consulta Condition keys for Amazon QLDB nel Service Authorization Reference. Per sapere con quali azioni e risorse puoi utilizzare una chiave di condizione, consulta Azioni definite da Amazon QLDB.

Le `PartiQLDropTable` azioni `PartiQLDropIndex` and supportano la chiave di `qldb:Purge` condizione. Questa chiave condizionale filtra l'accesso in base al valore specificato in un'istruzione PartiQL`DROP`. `purge` Tuttavia, QLDB attualmente supporta `purge = true` solo le istruzioni e `DROP INDEX` le istruzioni. `purge = false` `DROP TABLE` Altre azioni QLDB supportano alcune chiavi di condizione globali.

Per visualizzare esempi di policy basate sull'identità QLDB, vedere. Esempi di policy basate sull'identità per Amazon QLDB

Elenchi di controllo degli accessi (ACL) in QLDB

Supporta le ACL	No

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni ad accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino il formato del documento di policy JSON.

Controllo degli accessi basato sugli attributi (ABAC) con QLDB

Supporta ABAC (tag nelle policy)	Sì

Il controllo dell'accesso basato su attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base agli attributi. In, questi attributi sono chiamati AWS tag. Puoi allegare tag a entità IAM (utenti o ruoli) e a molte AWS risorse. L'assegnazione di tag alle entità e alle risorse è il primo passaggio di ABAC. In seguito, vengono progettate policy ABAC per consentire operazioni quando il tag dell'entità principale corrisponde al tag sulla risorsa a cui si sta provando ad accedere.

La strategia ABAC è utile in ambienti soggetti a una rapida crescita e aiuta in situazioni in cui la gestione delle policy diventa impegnativa.

Per controllare l'accesso basato su tag, fornisci informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Yes (Sì). Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per ulteriori informazioni su ABAC, consulta Che cos'è ABAC? nella Guida per l'utente di IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consulta Utilizzo del controllo degli accessi basato su attributi (ABAC) nella Guida per l'utente di IAM.

Per ulteriori informazioni sull'etichettatura delle risorse QLDB, vedere. Assegnazione di tag alle risorse Amazon QLDB

Per visualizzare una policy basata sulle identità di esempio per limitare l'accesso a una risorsa basata su tag su tale risorsa, consulta Aggiornamento dei registri QLDB in base ai tag.

Utilizzo di credenziali temporanee con QLDB

Supporta le credenziali temporanee	Sì

Alcune Servizi AWS non funzionano quando accedi utilizzando credenziali temporanee. Per ulteriori informazioni, incluse quelle che Servizi AWS funzionano con credenziali temporanee, consulta la sezione relativa alla Servizi AWS compatibilità con IAM nella IAM User Guide.

Stai utilizzando credenziali temporanee se accedi AWS Management Console utilizzando qualsiasi metodo tranne nome utente e password. Ad esempio, quando accedete AWS utilizzando il link Single Sign-On (SSO) della vostra azienda, tale processo crea automaticamente credenziali temporanee. Le credenziali temporanee vengono create in automatico anche quando accedi alla console come utente e poi cambi ruolo. Per ulteriori informazioni sullo scambio dei ruoli, consulta Cambio di un ruolo (console) nella Guida per l'utente di IAM.

È possibile creare manualmente credenziali temporanee utilizzando l'API or. AWS CLI AWS È quindi possibile utilizzare tali credenziali temporanee per accedere. AWS AWS consiglia di generare dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta Credenziali di sicurezza provvisorie in IAM.

Autorizzazioni principali multiservizio per QLDB

Supports forward access sessions (FAS)	No

Quando utilizzi un utente o un ruolo IAM per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FAS utilizza le autorizzazioni del principale che chiama an Servizio AWS, in combinazione con la richiesta Servizio AWS per effettuare richieste ai servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Forward access sessions.

Ruoli di servizio per QLDB

Supporta i ruoli di servizio	Sì

Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.

> **Warning**
>
> La modifica delle autorizzazioni per un ruolo di servizio potrebbe interrompere la funzionalità QLDB. Modifica i ruoli di servizio solo quando QLDB fornisce indicazioni in tal senso.

QLDB supporta i ruoli di servizio per `ExportJournalToS3` le operazioni `StreamJournalToKinesis` e le API, come descritto nella sezione seguente.

Scelta di un ruolo IAM in QLDB

Quando esportate o trasmettete in streaming blocchi di journal in QLDB, dovete scegliere un ruolo per consentire a QLDB di scrivere oggetti nella destinazione specificata per vostro conto. Se in precedenza hai creato un ruolo di servizio, QLDB ti fornisce un elenco di ruoli tra cui scegliere. È importante scegliere un ruolo che consenta l'accesso alla scrittura nel bucket Amazon S3 specificato per un'esportazione o alla risorsa Amazon Kinesis Data Streams specificata per uno stream. Per ulteriori informazioni, consulta Autorizzazioni di esportazione del diario in QLDB o Autorizzazioni di streaming in QLDB.

> **Note**
>
> Per passare un ruolo a QLDB quando si richiede un'esportazione o uno stream del journal, è necessario disporre delle autorizzazioni per eseguire `iam:PassRole` l'azione sulla risorsa del ruolo IAM. Ciò si aggiunge alle autorizzazioni da eseguire `qldb:ExportJournalToS3` sulla risorsa registro QLDB o `qldb:StreamJournalToKinesis` sulla sottorisorsa del flusso QLDB.

Ruoli collegati ai servizi per QLDB

Supporta i ruoli collegati ai servizi	No

Un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'operazione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

Per informazioni dettagliate sulla creazione o la gestione di ruoli collegati ai servizi, consulta Servizi AWS That work with IAM. Trova un servizio nella tabella che include un `Yes` nella colonna Service-linked role (Ruolo collegato ai servizi). Scegli il collegamento Sì per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Guida introduttiva alla modalità di autorizzazione standard in Amazon QLDB

Usa questa sezione per iniziare a usare la modalità di autorizzazione standard in Amazon QLDB. Questa sezione fornisce una tabella di riferimento per aiutarti a scrivere una policy basata sull'identità in AWS Identity and Access Management (IAM) per azioni PartiQL e risorse di tabella in QLDB. Include anche un step-by-step tutorial per la creazione di politiche di autorizzazione in IAM e istruzioni per trovare un ARN di tabella e creare tag di tabella in QLDB.

Argomenti

- La modalità di autorizzazione STANDARD
- Riferimento alle autorizzazioni PartiQL
- Ricerca di un ID di tabella e di un ARN
- Tabelle di etichettatura
- Tutorial di avvio rapido: Creazione di politiche di autorizzazione

La modalità di autorizzazione **`STANDARD`**

QLDB ora supporta `STANDARD` una modalità di autorizzazione per le risorse contabili. La modalità di autorizzazione standard consente il controllo degli accessi con una maggiore granularità per registri, tabelle e comandi PartiQL. Per impostazione predefinita, questa modalità nega tutte le richieste di esecuzione di comandi PartiQL su qualsiasi tabella in un registro.

> **Note**
>
> In precedenza, l'unica modalità di autorizzazione disponibile per un registro era. `ALLOW_ALL` La `ALLOW_ALL` modalità consente il controllo degli accessi con granularità a livello di API per i registri e continua a essere supportata, ma non è consigliata, per i registri QLDB. Questa modalità consente agli utenti che dispongono dell'autorizzazione `SendCommand` API di eseguire tutti i comandi PartiQL su qualsiasi tabella del registro specificato dalla politica di autorizzazione (quindi, «consenti tutti» i comandi PartiQL).
> È possibile modificare la modalità di autorizzazione dei registri esistenti da a. `ALLOW_ALL` `STANDARD` Per informazioni, consulta Migrazione alla modalità di autorizzazione standard.

Per consentire i comandi in modalità standard, devi creare una politica di autorizzazioni in IAM per risorse di tabella specifiche e azioni PartiQL. Ciò si aggiunge all'autorizzazione `SendCommand` API per il registro. Per facilitare le politiche in questa modalità, QLDB ha introdotto una serie di azioni IAM per i comandi PartiQL e Amazon Resource Names (ARN) per le tabelle QLDB. Per ulteriori informazioni sul modello di oggetti dati QLDB, vedere. Concetti e terminologia fondamentali in Amazon QLDB

Riferimento alle autorizzazioni PartiQL

La tabella seguente elenca ogni comando QLDB PartiQL, le azioni IAM corrispondenti per le quali è necessario concedere le autorizzazioni per eseguire il comando AWS e le risorse per le quali è possibile concedere le autorizzazioni. Puoi specificare le azioni nel campo `Action` della policy e il valore della risorsa nel campo `Resource`.

> **Important**
>
> - Le politiche IAM che concedono le autorizzazioni a questi comandi PartiQL si applicano al registro solo se `STANDARD` la modalità di autorizzazione è assegnata al registro. Tali politiche non sono applicabili ai registri in modalità autorizzazioni. `ALLOW_ALL`
>
> Per informazioni su come specificare la modalità di autorizzazione durante la creazione o l'aggiornamento di un registroOperazioni di base per i libri mastri Amazon QLDB, consulta o Fase 1: creazione di un nuovo libro mastro in Guida introduttiva alla console.
> - Per eseguire qualsiasi comando PartiQL su un registro, è inoltre necessario concedere l'autorizzazione all'azione `SendCommand` API per la risorsa registro. Ciò si aggiunge alle

azioni PartiQL e alle risorse della tabella elencate nella tabella seguente. Per ulteriori informazioni, consulta Esecuzione di transazioni di dati.

Comandi Amazon QLDB PartiQL e autorizzazioni richieste

Comando	Autorizzazioni richieste (azioni IAM)	Risorse	Operazioni dipendenti
CREATE TABLE	qldb:PartiQLCreateTable	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*	qldb:TagResource (per l'aggiunta di tag alla creazione)
DROP TABLE	qldb:PartiQLDropTable	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
TAVOLO UNDROP	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	qldb:PartiQLDelete	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	qldb:PartiQLSelect

Comando	Autorizzazioni richieste (azioni IAM)	Risorse	Operazioni dipendenti
(per interi documenti)			
INSERT	qldb:PartiQLInsert	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
UPDATE DA (INSERISCI, RIMUOVI o IMPOSTA)	qldb:PartiQLUpdate	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	qldb:PartiQLSelect
REVISIONE_REDAZIONE(procedura memorizzata)	qldb:PartiQLRedact	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
SELEZIONA DA table_name	qldb:PartiQLSelect	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	

Comando	Autorizzazioni richieste (azioni IAM)	Risorse	Operazioni dipendenti
SELEZIONA information_schema.user_tables	qldb:PartiQLSelect	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /information_schema/user_tables	
SELEZIONA DALLA cronologia (table_name)	qldb:PartiQLHistoryFunction	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	

Per esempi di documenti relativi alle policy IAM che concedono le autorizzazioni a questi comandi PartiQL, consulta o consultaTutorial di avvio rapido: Creazione di politiche di autorizzazione. Esempi di policy basate sull'identità per Amazon QLDB

Ricerca di un ID di tabella e di un ARN

È possibile trovare un ID di tabella utilizzando AWS Management Console o interrogando la tabella information_schema.user_tables. Per visualizzare i dettagli della tabella sulla console o per interrogare questa tabella del catalogo di sistema, è necessario disporre dell'autorizzazione per la risorsa del catalogo di sistema. `SELECT` Ad esempio, per trovare l'ID della `Vehicle` tabella, è possibile eseguire la seguente istruzione.

```
SELECT * FROM information_schema.user_tables
WHERE name = 'Vehicle'
```

Questa query restituisce i risultati in un formato simile all'esempio seguente.

```
{
```

```
    tableId: "Au1EiThbt8s0z9wM26REZN",
    name: "Vehicle",
    indexes: [
        { indexId: "Djg2nt0yIs2GY0T29Kud1z", expr: "[VIN]", status: "ONLINE" },
        { indexId: "4tPW3fUhaVhDinRgKRLhGU", expr: "[LicensePlateNumber]", status:
 "BUILDING" }
    ],
    status: "ACTIVE"
}
```

Per concedere le autorizzazioni per eseguire istruzioni PartiQL su una tabella, si specifica una risorsa di tabella nel seguente formato ARN.

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

Di seguito è riportato un esempio di tabella ARN per l'ID della tabella. `Au1EiThbt8s0z9wM26REZN`

```
arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/Au1EiThbt8s0z9wM26REZN
```

Utilizzo della console

Puoi anche usare la console QLDB per trovare un ARN di tabella.

Per trovare l'ARN di una tabella (console)

1. Accedi a e apri AWS Management Console la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel pannello di navigazione, scegli Ledgers.
3. Nell'elenco dei libri contabili, scegliete il nome del libro contabile di cui desiderate trovare l'ARN della tabella.
4. Nella pagina dei dettagli del registro, nella scheda Tabelle, individua il nome della tabella di cui desideri trovare l'ARN. Per copiare l'ARN, scegliete l'icona di copia () accanto ad esso.

Tabelle di etichettatura

Puoi etichettare le risorse della tua tabella. Per gestire i tag per le tabelle esistenti, utilizza AWS Management Console o le operazioni `TagResource` API `eListTagsForResource`. `UntagResource` Per ulteriori informazioni, consulta Assegnazione di tag alle risorse Amazon QLDB.

> **Note**
>
> Le risorse delle tabelle non ereditano i tag della loro risorsa root ledger.
> L'etichettatura delle tabelle al momento della creazione è attualmente supportata per i libri contabili solo in modalità autorizzazioni. `STANDARD`

È inoltre possibile definire i tag della tabella durante la creazione della tabella utilizzando la console QLDB o specificandoli in un'istruzione PartiQL. `CREATE TABLE` L'esempio seguente crea una tabella denominata `Vehicle` con il tag. `environment=production`

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

L'aggiunta di tag alle tabelle al momento della creazione richiede l'accesso `qldb:PartiQLCreateTable` sia alle `qldb:TagResource` azioni che.

L'aggiunta di tag alle risorse in fase di creazione consente di evitare di eseguire script di tagging personalizzati dopo la creazione delle risorse. Dopo aver aggiunto i tag a una tabella, puoi controllare l'accesso alla tabella in base a tali tag. Ad esempio, puoi concedere l'accesso completo solo alle tabelle che hanno un tag specifico. Per un esempio di policy JSON, vediAccesso completo a tutte le azioni basate sui tag della tabella.

Utilizzo della console

Puoi anche utilizzare la console QLDB per definire i tag della tabella durante la creazione della tabella.

Per etichettare una tabella al momento della creazione (console)

1. Accedi a e apri AWS Management Console la console Amazon QLDB all'indirizzo https://console.aws.amazon.com/qldb.
2. Nel pannello di navigazione, scegli Ledgers.

3. Nell'elenco dei libri contabili, scegli il nome del libro contabile in cui desideri creare la tabella.
4. Nella pagina dei dettagli del registro, nella scheda Tabelle, scegli Crea tabella.
5. Nella pagina Crea tabella, procedi come segue:

   - Nome tabella: immettete un nome per la tabella.
   - Tag: aggiungi metadati alla tabella allegando i tag come coppie chiave-valore. Puoi aggiungere tag alla tabella per organizzarli e identificarli.

     Scegli Aggiungi tag, quindi inserisci le coppie chiave-valore appropriate.
6. Dopo aver selezionato le impostazioni desiderate, scegli Create table (Crea tabella).

Tutorial di avvio rapido: Creazione di politiche di autorizzazione

Questo tutorial ti guida attraverso i passaggi per creare politiche di autorizzazione in IAM per un registro Amazon QLDB in modalità autorizzazioni. `STANDARD` Puoi quindi assegnare le autorizzazioni ai tuoi utenti, gruppi o ruoli.

Per altri esempi di documenti relativi alle policy IAM che concedono autorizzazioni ai comandi PartiQL e alle risorse delle tabelle, consulta. Esempi di policy basate sull'identità per Amazon QLDB

Argomenti

- Prerequisiti
- Creazione di una politica di sola lettura
- Creazione di una politica di accesso completa
- Creazione di una politica di sola lettura per una tabella specifica
- Assegnare le autorizzazioni

Prerequisiti

Prima di iniziare, assicurati di fare quanto segue:

1. Segui le istruzioni di AWS configurazione riportate inAccesso ad Amazon QLDB, se non l'hai già fatto. Questi passaggi includono la registrazione AWS e la creazione di un utente amministrativo.
2. Crea un nuovo registro e scegli la modalità di `STANDARD` autorizzazione per il registro. Per ulteriori informazioni, consulta la sezione Fase 1: creazione di un nuovo libro mastro Guida introduttiva alla console oppure. Operazioni di base per i libri mastri Amazon QLDB

Creazione di una politica di sola lettura

Per utilizzare l'editor di policy JSON per creare una policy di sola lettura per tutte le tabelle di un registro nella modalità di autorizzazione standard, procedi come segue:

1. Accedi AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione a sinistra, seleziona Policies (Policy).

   Se è la prima volta che selezioni Policy, verrà visualizzata la pagina Benvenuto nelle policy gestite. Seleziona Inizia.
3. Nella parte superiore della pagina, scegli Crea policy.
4. Scegli la scheda JSON.
5. Copia e incolla il seguente documento di policy JSON. Questo criterio di esempio concede l'accesso in sola lettura a tutte le tabelle di un libro mastro.

   Per utilizzare questa politica, sostituisci *us-east-1,* 123456789012 *myExampleLedger*e nell'esempio con le tue informazioni.

```
{
  "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. Scegli Verifica policy.

> **Note**
>
> È possibile passare tra le schede Visual editor (Editor visivo) e JSON in qualsiasi momento. Se tuttavia si apportano modifiche o se si seleziona Review policy (Rivedi policy) nella scheda Visual editor (Editor visivo), IAM potrebbe modificare la policy per ottimizzarla per l'editor visivo. Per ulteriori informazioni, consulta Modifica della struttura delle policy nella Guida per l'utente di IAM.

7. Nella pagina Review policy (Rivedi policy), inserisci i valori per Name (Nome) e Description (Descrizione) (facoltativa) per la policy che stai creando. Consulta il Summary (Riepilogo) della policy per visualizzare le autorizzazioni concesse dalla policy. Seleziona Create policy (Crea policy) per salvare il proprio lavoro.

Creazione di una politica di accesso completa

Per creare una politica di accesso completo per tutte le tabelle in un registro QLDB nella modalità di autorizzazione standard, procedi come segue:

- Ripetere i passaggi precedenti utilizzando il seguente documento di policy. Questo criterio di esempio consente l'accesso a tutti i comandi PartiQL per tutte le tabelle in un registro, utilizzando caratteri jolly (*) per coprire tutte le azioni PartiQL e tutte le risorse in un registro.

> **Warning**
>
> Questo è un esempio di utilizzo di un carattere jolly (*) per consentire tutte le azioni PartiQL, incluse le operazioni amministrative e di lettura/scrittura su tutte le tabelle in un registro QLDB. È invece consigliabile specificare esplicitamente ogni azione da concedere e solo ciò di cui l'utente, il ruolo o il gruppo ha bisogno.

Per utilizzare questa politica, sostituisci *us-east-1,* 123456789012 *myExampleLedger*e nell'esempio con le tue informazioni.

```
{
```

```
  "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"
      ]
    }
  ]
}
```

Creazione di una politica di sola lettura per una tabella specifica

Per creare una politica di accesso di sola lettura per una tabella specifica in un registro QLDB nella modalità di autorizzazione standard, procedi come segue:

1. Trova l'ARN per la tabella utilizzando AWS Management Console o interrogando la tabella del catalogo di sistema. `information_schema.user_tables` Per istruzioni, consulta Ricerca di un ID di tabella e di un ARN.
2. Utilizza la tabella ARN per creare una policy che consenta l'accesso in sola lettura alla tabella. Per fare ciò, ripeti i passaggi precedenti utilizzando il seguente documento di policy.

   Questo criterio di esempio concede l'accesso in sola lettura solo alla tabella specificata. In questo esempio, l'ID della tabella è. `Au1EiThbt8s0z9wM26REZN` Per utilizzare questa politica, sostituisci *`us-east-1,`* 123456789012 *`e EiThbt Au1 myExampleLedger8S0z9WM26Rezn nell'esempio con le tue informazioni`*.

   ```
   {
     "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"
      ]
    }
  ]
}
```

Assegnare le autorizzazioni

Dopo aver creato una politica di autorizzazioni QLDB, assegna le autorizzazioni come segue.

Per fornire l'accesso, aggiungi autorizzazioni ai tuoi utenti, gruppi o ruoli:

- Utenti e gruppi in: AWS IAM Identity Center

  Crea un set di autorizzazioni. Segui le istruzioni riportate nella pagina Create a permission set (Creazione di un set di autorizzazioni) nella Guida per l'utente di AWS IAM Identity Center .

- Utenti gestiti in IAM tramite un provider di identità:

  Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate nella pagina Creating a role for a third-party identity provider (federation) (Creazione di un ruolo per un provider di identità di terze parti [federazione]) nella Guida per l'utente di IAM.

- Utenti IAM:
  - Crea un ruolo che l'utente possa assumere. Per istruzioni, consulta la pagina Creating a role for an IAM user (Creazione di un ruolo per un utente IAM) nella Guida per l'utente di IAM.

- (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate nella pagina Aggiunta di autorizzazioni a un utente (console) nella Guida per l'utente di IAM.

Esempi di policy basate sull'identità per Amazon QLDB

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse QLDB. Inoltre, non possono eseguire attività utilizzando AWS Management Console, AWS Command Line Interface (AWS CLI) o AWS l'API. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. L'amministratore può quindi aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Per dettagli sulle azioni e sui tipi di risorse definiti da QLDB, incluso il formato degli ARN per ciascun tipo di risorsa, consulta Azioni, risorse e chiavi di condizione per Amazon QLDB nel Service Authorization Reference.

Indice

- Esportazione di un journal in un bucket Amazon S3
- Trasmissione di un diario su Kinesis Data Streams
- Aggiornamento dei registri QLDB in base ai tag

Best practice per le policy

Le politiche basate sull'identità determinano se qualcuno può creare, accedere o eliminare risorse QLDB nel tuo account. Queste azioni possono comportare costi aggiuntivi per l' Account AWS. Quando crei o modifichi policy basate su identità, segui queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le policy gestite che concedono le autorizzazioni per molti casi d'uso comuni.AWS Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per ulteriori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente IAM.
- Applica le autorizzazioni con privilegi minimi: quando imposti le autorizzazioni con le policy IAM, concedi solo le autorizzazioni richieste per eseguire un'attività. Puoi farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegi minimi. Per ulteriori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso: per limitare l'accesso ad azioni e risorse puoi aggiungere una condizione alle tue policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio AWS CloudFormation. Per ulteriori informazioni, consulta la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo di IAM Access Analyzer per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali: IAM Access Analyzer convalida le policy nuove ed esistenti in modo che aderiscano alla sintassi della policy IAM (JSON) e alle best practice di IAM. IAM Access Analyzer offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per ulteriori informazioni, consulta Convalida delle policy per IAM Access Analyzer nella Guida per l'utente di IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungi le condizioni MFA alle policy. Per ulteriori

informazioni, consulta Configurazione dell'accesso alle API protetto con MFA nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Utilizzo della console QLDB

Per accedere alla console Amazon QLDB, devi disporre di un set minimo di autorizzazioni. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle risorse QLDB presenti nel tuo. Account AWS Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Non è necessario consentire autorizzazioni minime per la console per gli utenti che effettuano chiamate solo verso o l' AWS CLI API. AWS Al contrario, concedi l'accesso solo alle operazioni che corrispondono all'operazione API che stanno cercando di eseguire.

Per garantire che utenti e ruoli abbiano pieno accesso alla console QLDB e a tutte le sue funzionalità, allega la AWS seguente policy gestita alle entità. Per ulteriori informazioniAWS politiche gestite per Amazon QLDB, consulta Aggiungere autorizzazioni a un utente nella Guida per l'utente IAM.

```
AmazonQLDBConsoleFullAccess
```

Autorizzazioni per la cronologia delle query

Oltre alle autorizzazioni QLDB, alcune funzionalità della console richiedono le autorizzazioni per il Database Query Metadata Service (prefisso del servizio:). `dbqms` Si tratta di un servizio solo interno che gestisce le query recenti e salvate nell'editor di query della console per QLDB e altri. Servizi AWSPer un elenco completo delle azioni dell'API DBQMS, consulta Database Query Metadata Service nel Service Authorization Reference.

Per consentire le autorizzazioni relative alla cronologia delle query, puoi utilizzare la policy gestita AmazonQLDB AWS . ConsoleFullAccess Questa policy utilizza un wildcard (`dbqms:*`) per consentire tutte le azioni DBQMS per tutte le risorse.

In alternativa, puoi creare una policy IAM personalizzata e includere le seguenti azioni DBQMS. L'editor di query PartiQL sulla console QLDB richiede le autorizzazioni per utilizzare queste azioni per le funzionalità della cronologia delle query.

```
dbqms:CreateFavoriteQuery
dbqms:CreateQueryHistory
dbqms:DeleteFavoriteQueries
dbqms:DeleteQueryHistory
dbqms:DescribeFavoriteQueries
dbqms:DescribeQueryHistory
dbqms:UpdateFavoriteQuery
```

Autorizzazioni di accesso completo alla console senza cronologia delle query

Per consentire l'accesso completo alla console QLDB senza alcuna autorizzazione alla cronologia delle query, puoi creare una policy IAM personalizzata che escluda tutte le azioni DBQMS. Ad esempio, il seguente documento di policy consente le stesse autorizzazioni concesse dalla policy AWS gestita AmazonQLDB ConsoleFullAccess, ad eccezione delle azioni che iniziano con il prefisso del servizio. 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"
        }
      }
    }
  ]
}
```

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono cpllegate alla relativa identità utente. Questa

politica include le autorizzazioni per completare questa azione sulla console o utilizzando programmaticamente l'API o. AWS CLI 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": "*"
        }
    ]
}
```

Esecuzione di transazioni di dati

Per interagire con l'API dei dati transazionali QLDB (sessione QLDB) eseguendo istruzioni PartiQL su un registro, è necessario concedere l'autorizzazione all'azione API. `SendCommand` Il seguente documento JSON è un esempio di policy che concede l'autorizzazione solo all'azione API sul registro. `SendCommand` `myExampleLedger`

Per utilizzare questa politica, sostituisci *us-east-1,* 123456789012 *myExampleLedger*e nell'esempio con le tue informazioni.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        }
    ]
}
```

Se `myExampleLedger` utilizza la modalità `ALLOW_ALL` autorizzazioni, questa politica concede le autorizzazioni per eseguire tutti i comandi PartiQL su qualsiasi tabella del registro.

Puoi anche utilizzare una policy AWS gestita per concedere l'accesso completo a tutte le risorse QLDB. Per ulteriori informazioni, consulta AWS politiche gestite per Amazon QLDB.

Autorizzazioni standard per le azioni PartiQL e le risorse delle tabelle

Per i registri in modalità `STANDARD` autorizzazioni, puoi fare riferimento ai seguenti documenti sulle policy IAM come esempi di concessione delle autorizzazioni PartiQL appropriate. Per un elenco delle autorizzazioni richieste per ogni comando PartiQL, vedere. Riferimento alle autorizzazioni PartiQL

Argomenti

- Accesso completo a tutte le azioni
- Accesso completo a tutte le azioni basate sui tag della tabella
- Accesso in lettura/scrittura
- Accesso in sola lettura
- Accesso in sola lettura a una tabella specifica
- Consenti l'accesso per creare tabelle
- Consenti l'accesso per creare tabelle basate sui tag di richiesta

Accesso completo a tutte le azioni

Il seguente documento di policy JSON garantisce l'accesso completo all'utilizzo di tutti i comandi PartiQL su tutte le tabelle di. `myExampleLedger` Questa politica produce lo stesso effetto dell'utilizzo della modalità di `ALLOW_ALL` autorizzazione per il registro.

Per utilizzare questa politica, sostituisci *us-east-1,* 123456789012 *myExampleLedger*e nell'esempio con le tue informazioni.

```
{
    "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"
            ]
        }
    ]
}
```

Accesso completo a tutte le azioni basate sui tag della tabella

Il seguente documento di policy JSON utilizza una condizione basata sui tag delle risorse della tabella per concedere l'accesso completo all'utilizzo di tutti i comandi PartiQL su tutte le tabelle in. `myExampleLedger` Le autorizzazioni vengono concesse solo se il tag della tabella `environment` ha il valore. `development`

> **Warning**
>
> Questo è un esempio di utilizzo di un carattere jolly (*) per consentire tutte le azioni PartiQL, incluse le operazioni amministrative e di lettura/scrittura su tutte le tabelle in un registro QLDB. È invece consigliabile specificare esplicitamente ogni azione da concedere e solo ciò di cui l'utente, il ruolo o il gruppo ha bisogno.

Per utilizzare questa politica, sostituisci *us-east-1,* 123456789012 *myExampleLedger*e nell'esempio con le tue informazioni.

```
{
    "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" }
            }
```

```
        }
    ]
}
```

Accesso in lettura/scrittura

Il seguente documento sulla politica JSON concede le autorizzazioni per selezionare, inserire, aggiornare ed eliminare i dati su tutte le tabelle di. `myExampleLedger` Questa politica non concede le autorizzazioni per oscurare i dati o modificare lo schema, ad esempio per creare e eliminare tabelle e indici.

> **Note**
>
> Un'`UPDATE`istruzione richiede le autorizzazioni sia per le azioni che per le azioni sulla tabella che viene `qldb:PartiQLUpdate` modificata`qldb:PartiQLSelect`. Quando si esegue un'`UPDATE`istruzione, questa esegue un'operazione di lettura oltre all'operazione di aggiornamento. La richiesta di entrambe le azioni garantisce che solo gli utenti autorizzati a leggere il contenuto di una tabella ricevano `UPDATE` le autorizzazioni.
> Analogamente, un'`DELETE`istruzione richiede le autorizzazioni sia per le azioni che per `qldb:PartiQLDelete` le `qldb:PartiQLSelect` azioni.

Per utilizzare questa politica, sostituisci *`us-east-1,`* 123456789012 *`myExampleLedger`*e nell'esempio con le tue informazioni.

```
{
    "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"
            ]
        }
    ]
}
```

Accesso in sola lettura

Il seguente documento sulla politica JSON concede autorizzazioni di sola lettura su tutte le tabelle in. `myExampleLedger` Per utilizzare questa politica, sostituisci *us-east-1,* 123456789012 *myExampleLedger*e nell'esempio con le tue informazioni.

```
{
    "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"
            ]
        }
    ]
}
```

Accesso in sola lettura a una tabella specifica

Il seguente documento di policy JSON concede autorizzazioni di sola lettura su una tabella specifica in. `myExampleLedger` In questo esempio, l'ID della tabella è. `Au1EiThbt8s0z9wM26REZN`

Per utilizzare questa politica, sostituisci *us-east-1,* 123456789012 *e EiThbt Au1 myExampleLedger8S0z9WM26Rezn nell'esempio con le tue informazioni*.

```
{
    "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"
            ]
        }
    ]
}
```

Consenti l'accesso per creare tabelle

Il seguente documento di policy JSON concede il permesso di creare tabelle in. `myExampleLedger` L'`qldb:PartiQLCreateTable`azione richiede le autorizzazioni per il tipo di risorsa della tabella. Tuttavia, l'ID di tabella di una nuova tabella non è noto al momento dell'esecuzione di un'`CREATE TABLE`istruzione. Pertanto, una politica che concede l'`qldb:PartiQLCreateTable`autorizzazione deve utilizzare un carattere jolly (*) nella tabella ARN per specificare la risorsa.

Per utilizzare questa politica, sostituisci *us-east-1,* 123456789012 *myExampleLedger*e nell'esempio con le tue informazioni.

```
{
    "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/*"
            ]
        }
    ]
}
```

Consenti l'accesso per creare tabelle basate sui tag di richiesta

Il seguente documento di policy JSON utilizza una condizione basata sulla chiave di `aws:RequestTag` contesto per concedere l'autorizzazione alla creazione di tabella`myExampleLedger`. Le autorizzazioni vengono concesse solo se il tag di richiesta `environment` ha il valore. `development` L'aggiunta di tag alle tabelle al momento della creazione richiede l'accesso `qldb:PartiQLCreateTable` sia `qldb:TagResource` alle azioni che. Per informazioni su come etichettare le tabelle al momento della creazione, consultaTabelle di etichettatura.

Per utilizzare questa politica, sostituisci *`us-east-1,`* 123456789012 *`myExampleLedger`*e nell'esempio con le tue informazioni.

```
{
    "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" }
            }
        }
    ]
}
```

Esportazione di un journal in un bucket Amazon S3

Fase 1: Autorizzazioni di esportazione del journal QLDB

Nell'esempio seguente, concedi a un utente Account AWS le autorizzazioni per eseguire l'`qldb:ExportJournalToS3`azione su una risorsa di registro QLDB. Concedi anche le autorizzazioni per eseguire l'`iam:PassRole`azione sulla risorsa del ruolo IAM che desideri passare al servizio QLDB. Questo è necessario per tutte le richieste di esportazione del diario.

Per utilizzare questa politica, sostituisci *`us-east-1,`* 123456789012 *`e`* qldb-s3-export nell'*`myExampleLedger`*esempio con le tue informazioni.

```
{
    "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"
               }
            }
        }
    ]
}
```

Fase 2: autorizzazioni per i bucket Amazon S3

Nell'esempio seguente, utilizzi un ruolo IAM per concedere l'accesso a QLDB per la scrittura in uno dei tuoi bucket Amazon S3,. `DOC-EXAMPLE-BUCKET` Ciò è necessario anche per tutte le esportazioni di giornali QLDB.

Oltre a concedere l'`s3:PutObject`autorizzazione, la politica concede anche l'`s3:PutObjectAcl`autorizzazione per la possibilità di impostare le autorizzazioni dell'elenco di controllo degli accessi (ACL) per un oggetto.

Per utilizzare questa politica, sostituisci DOC-EXAMPLE-BUCKET nell'esempio con il nome del tuo bucket Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

Quindi, colleghi questa policy di autorizzazione a un ruolo IAM che QLDB può assumere per accedere al tuo bucket Amazon S3. Il seguente documento JSON è un esempio di policy di fiducia

che consente a QLDB di assumere il ruolo IAM solo per qualsiasi risorsa QLDB nell'account. `123456789012`

Per utilizzare questa politica, sostituisci *`us-east-1`* *`e`* 123456789012 nell'esempio con le tue informazioni.

```
{
    "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"
                }
            }
        }
    ]
}
```

Trasmissione di un diario su Kinesis Data Streams

Passaggio 1: autorizzazioni per lo streaming del journal QLDB

Nell'esempio seguente, concedi a un utente tra le tue Account AWS autorizzazioni per eseguire l'`qldb:StreamJournalToKinesis`azione su tutte le sottorisorse di flusso QLDB in un registro. Concedi anche le autorizzazioni per eseguire l'`iam:PassRole`azione sulla risorsa del ruolo IAM che desideri passare al servizio QLDB. Questa operazione è necessaria per tutte le richieste di flusso journal.

Per utilizzare questa politica, sostituisci *`us-east-1`*, 123456789012 *`qldb-kinesis-stream`*e, nell'esempio *`myExampleLedger`*, con le tue informazioni.

```
{
    "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"
               }
            }
        }
    ]
}
```

Fase 2: Autorizzazioni Kinesis Data Streams

Nell'esempio seguente, utilizzi un ruolo IAM per concedere l'accesso a QLDB per scrivere record di dati nel flusso di dati di Amazon Kinesis,. *stream-for-qldb* Ciò è necessario anche per tutti i flussi di journal QLDB.

Per utilizzare questa politica, sostituisci *us-east-1,* 123456789012 *stream-for-qldb*e nell'esempio con le tue informazioni.

```
{
    "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"
        }
    ]
}
```

Quindi, colleghi questa policy di autorizzazione a un ruolo IAM che QLDB può assumere per accedere al flusso di dati Kinesis. Il seguente documento JSON è un esempio di policy di fiducia che consente a QLDB di assumere un ruolo IAM per qualsiasi flusso QLDB nell'account solo per il registro. `123456789012` `myExampleLedger`

Per utilizzare questa politica, sostituisci *`us-east-1,`* 123456789012 *`myExampleLedger`*e nell'esempio con le tue informazioni.

```
{
    "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"
                }
            }
        }
    ]
}
```

Aggiornamento dei registri QLDB in base ai tag

Puoi utilizzare le condizioni nella tua policy basata sull'identità per controllare l'accesso alle risorse QLDB in base ai tag. Questo esempio mostra come è possibile creare una politica che consenta l'aggiornamento di un registro. Tuttavia, l'autorizzazione viene concessa solo se il tag ledger `Owner` ha il valore del nome utente di quell'utente. Questa policy concede anche le autorizzazioni necessarie per completare questa azione nella 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}"}
            }
        }
    ]
}
```

Puoi collegare questa policy agli utenti nel tuo account. Se un utente denominato `richard-roe` tenta di aggiornare un registro QLDB, il registro deve essere contrassegnato con o. `Owner=richard-roe` `owner=richard-roe` In caso contrario, gli viene negato l'accesso. La chiave di tag di condizione `Owner` corrisponde sia a `Owner` che a `owner` perché i nomi delle chiavi di condizione non distinguono tra maiuscole e minuscole. Per ulteriori informazioni, consulta la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.

Prevenzione del confused deputy tra servizi

Con "confused deputy" si intende un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire una certa operazione può costringere un'entità con più privilegi a eseguire tale operazione. Nel frattempo AWS, l'impersonificazione tra servizi può portare al confuso problema del vicesceriffo.

La rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante) effettua una chiamata a un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare il problema del «Confused Deputy», AWS fornisce strumenti che consentono di proteggere i dati relativi a tutti i servizi, con responsabili del servizio a cui è stato concesso l'accesso alle risorse del vostro account.

Consigliamo di utilizzare le chiavi di contesto `aws:SourceArn` e `aws:SourceAccount` global condition nelle politiche delle risorse per limitare le autorizzazioni che Amazon QLDB fornisce a un altro servizio alla risorsa. Se utilizzi entrambe le chiavi di contesto della condizione globale, il `aws:SourceAccount` valore e l'account nel `aws:SourceArn` valore devono utilizzare lo stesso ID account quando vengono utilizzati nella stessa dichiarazione politica.

La tabella seguente elenca i valori possibili delle operazioni dell'API `aws:SourceArn` for `ExportJournalToS3` e `StreamsJournalToKinesis` QLDB. Queste operazioni rientrano nell'ambito di questo problema di sicurezza perché chiamano AWS Security Token Service (AWS STS) per assumere un ruolo IAM specificato dall'utente.

Operazione API	Servizio chiamato	leggi: SourceArn
`ExportJournalToS3`	AWS STS (`AssumeRole`)	Consente a QLDB di assumere il ruolo di qualsiasi risorsa QLDB nell'account: `arn:aws:qldb: us-east-1 :123456789012 :*` Attualmente, QLDB supporta solo questo ARN wildcard per le esportazioni di riviste.

`StreamsJournalToKinesis`	AWS STS (`AssumeRole`)	Consente a QLDB di assumere il ruolo per uno specifico stream QLDB:
		`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /IiPT4brpZCqCq3f4MTHbYy`
		Nota: è possibile specificare un ID di flusso nell'ARN solo dopo la creazione della risorsa di flusso. Utilizzando questo ARN, puoi consentire l'utilizzo del ruolo solo per un singolo flusso QLDB.
		Consente a QLDB di assumere il ruolo per qualsiasi flusso QLDB di un registro:
		`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /*`

Operazione API	Servizio chiamato	leggi: SourceArn

		Consente a QLDB di assumere il ruolo di qualsiasi stream QLDB nell'account:
		`arn:aws:qldb: us-east-1 :123456789012 :stream/*`
		Consente a QLDB di assumere il ruolo di qualsiasi risorsa QLDB nell'account:
		`arn:aws:qldb: us-east-1 :123456789012 :*`

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn`con l'ARN completo della risorsa. Se non conosci l'ARN completo della risorsa o se stai specificando più risorse, usa la chiave di condizione di contesto `aws:SourceArn` globale con caratteri jolly (*) per le parti sconosciute dell'ARN, ad esempio. `arn:aws:qldb:us-east-1:123456789012:*`

Il seguente esempio di politica di fiducia per un ruolo IAM mostra come utilizzare le chiavi di contesto `aws:SourceArn` e `aws:SourceAccount` global condition per evitare il confuso problema del vice. Con questa politica di fiducia, QLDB può assumere il ruolo di qualsiasi flusso QLDB presente nell'account solo per il registro. `123456789012` `myExampleLedger`

Per utilizzare questa politica, sostituisci *us-east-1,* 123456789012 *myExampleLedger*e nell'esempio con le tue informazioni.

```
{
  "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 politiche gestite per Amazon QLDB

Una policy AWS gestita è una policy autonoma creata e amministrata da. AWS AWS le politiche gestite sono progettate per fornire autorizzazioni per molti casi d'uso comuni, in modo da poter iniziare ad assegnare autorizzazioni a utenti, gruppi e ruoli.

Tieni presente che le policy AWS gestite potrebbero non concedere le autorizzazioni con il privilegio minimo per i tuoi casi d'uso specifici, poiché sono disponibili per tutti i clienti. AWS Consigliamo pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente specifiche per i tuoi casi d'uso.

Non è possibile modificare le autorizzazioni definite nelle politiche gestite. AWS Se AWS aggiorna le autorizzazioni definite in una politica AWS gestita, l'aggiornamento ha effetto su tutte le identità principali (utenti, gruppi e ruoli) a cui è associata la politica. AWS è più probabile che aggiorni una policy AWS gestita quando ne Servizio AWS viene lanciata una nuova o quando diventano disponibili nuove operazioni API per i servizi esistenti.

Per ulteriori informazioni, consultare Policy gestite da AWSnella Guida per l'utente di IAM.

Per ulteriori informazioni sulle operazioni dell'API QLDB in AWS queste politiche gestite, consulta. Documentazione di riferimento dell'API Amazon QLDB

Argomenti

- AWS politica gestita: AmazonQLDB ReadOnly
- AWS politica gestita: AmazonQLDB FullAccess
- AWS politica gestita: AmazonQLDB ConsoleFullAccess
- Aggiornamenti QLDB alle policy gestite AWS

AWS politica gestita: AmazonQLDB ReadOnly

Utilizza la ReadOnly policy AmazonQLDB per concedere autorizzazioni di sola lettura a tutte le risorse QLDB. Puoi collegare questa policy alle tue identità IAM.

Dettagli dell'autorizzazione

Questa politica include le seguenti autorizzazioni per il `qldb` servizio.

- Consente ai responsabili di descrivere ed elencare tutte le risorse QLDB e i relativi tag. Queste risorse includono registri, processi di esportazione in Amazon S3 e flussi verso Kinesis Data Streams.
- Consente ai responsabili di ottenere un blocco, un riepilogo o una revisione dal journal in qualsiasi registro per verificare i dati crittograficamente.
- Non consente ai principali di eseguire comandi PartiQL su nessuna tabella in nessun registro.

AWS politica gestita: AmazonQLDB FullAccess

Utilizza la FullAccess policy AmazonQLDB per concedere autorizzazioni amministrative complete a tutte le risorse QLDB tramite l'API QLDB o il. AWS CLI Puoi collegare questa policy alle tue identità IAM.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `qldb`
  - Consente ai responsabili di creare, descrivere, elencare e gestire tutte le risorse QLDB e i relativi tag. Queste risorse includono registri, processi di esportazione in Amazon S3 e flussi verso Kinesis Data Streams.
  - Consente ai principali di eseguire tutti i comandi PartiQL su tutte le tabelle in qualsiasi registro utilizzando il driver QLDB o la shell QLDB.
  - Consente ai responsabili di ottenere un blocco, un riepilogo o una revisione dal journal in qualsiasi registro per verificare i dati crittograficamente.
- `iam`— Consente ai responsabili di trasferire qualsiasi risorsa del ruolo IAM presente nell'account al servizio QLDB. Questo è necessario per tutte le richieste di esportazione e streaming del journal.

AWS politica gestita: AmazonQLDB ConsoleFullAccess

Utilizza la ConsoleFullAccess policy AmazonQLDB per concedere autorizzazioni amministrative complete a tutte le risorse QLDB tramite, l'API QLDB o AWS Management Console il. AWS CLI Puoi allegare questa policy alle tue identità IAM.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `qldb`
  - Consente ai responsabili di creare, descrivere, elencare e gestire tutte le risorse QLDB e i relativi tag. Queste risorse includono registri, processi di esportazione in Amazon S3 e flussi verso Kinesis Data Streams.
  - Consente ai principali di eseguire tutti i comandi PartiQL su tutte le tabelle in qualsiasi registro utilizzando la console QLDB, il driver QLDB o la shell QLDB.
  - Consente ai responsabili di inserire dati applicativi di esempio in qualsiasi registro utilizzando la console QLDB.
  - Consente ai responsabili di ottenere un blocco, un riepilogo o una revisione dal giornale in qualsiasi registro per verificare i dati crittograficamente.
- `dbqms`— Consente ai responsabili di utilizzare tutte le azioni del Database Query Metadata Service. Si tratta di un servizio solo interno richiesto dalla console QLDB per creare, descrivere e gestire le query recenti e salvate per l'editor di query PartiQL.
- `kinesis`— Consente ai responsabili di descrivere ed elencare le risorse di Amazon Kinesis Data Streams. Queste risorse sono le destinazioni di destinazione su cui le risorse di flusso QLDB possono scrivere dati.
- `iam`— Consente ai responsabili di trasferire qualsiasi risorsa del ruolo IAM presente nell'account al servizio QLDB. Questo è necessario per tutte le richieste di esportazione e streaming del journal.

Aggiornamenti QLDB alle policy gestite AWS

Visualizza i dettagli sugli aggiornamenti delle policy AWS gestite per QLDB da quando questo servizio ha iniziato a tenere traccia di queste modifiche. Per ricevere avvisi automatici sulle modifiche a questa pagina, iscriviti al feed RSS nella pagina della cronologia delle versioni di QLDB.

Modifica	Descrizione	Data
AmazonQLDB, AmazonQLDB: aggiornamento FullAccess alle politiche esistenti ConsoleFullAccess	QLDB ha aggiunto una nuova autorizzazione per consentire ai responsabili di redigere le revisioni dei documenti in tutti i registri in modalità autorizzazioni. `STANDARD`	4 novembre 2022
AmazonQLDB, FullAccessAmazonQLDB: aggiornamento alle politiche esistenti ConsoleFullAccess	QLDB ha aggiunto nuove autorizzazioni per consentire ai responsabili di trasferire qualsiasi risorsa del ruolo IAM nel tuo account al servizio QLDB. Questo è necessario per tutte le richieste di esportazione e streaming del diario.	2 settembre 2021
AmazonQLDB ReadOnly: aggiornamento a una politica esistente	QLDB ha rimosso un'azione `qldb:GetBlock` duplicata precedentemente elencata due volte e ha riordinato `"Effect"` il campo in modo che appaia prima del campo. `"Action"`	1° luglio 2021
AmazonQLDB, AmazonQLDB: aggiornamento FullAccess alle politiche esistenti ConsoleFullAccess	QLDB ha aggiunto nuove autorizzazioni per consentire ai responsabili di aggiornare la modalità di autorizzazione in tutti i registri e di eseguire tutti i comandi PartiQL in tutti i registri nella nuova modalità di autorizzazione. `STANDARD`	27 maggio 2021

Modifica	Descrizione	Data
	La modalità `STANDARD` autorizzazioni supporta il controllo degli accessi a livello di tabella e la granularità per i comandi PartiQL. Per facilitare la nuova modalità di autorizza zione, QLDB ha introdotto una serie di azioni IAM per i tipi di comandi PartiQL e Amazon Resource Names (ARN) per le risorse delle tabelle QLDB. Queste due politiche vengono aggiornate per includere le nuove azioni PartiQL per garantire l'accesso completo ai `STANDARD` registri.	
QLDB ha iniziato a tracciare le modifiche	QLDB ha iniziato a tracciare le modifiche per AWS le sue policy gestite.	1 marzo 2021

Risoluzione dei problemi relativi all'identità e all'accesso ad Amazon QLDB

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con QLDB e IAM.

Argomenti

- Non sono autorizzato a eseguire un'azione in QLDB
- Non sono autorizzato a eseguire iam: PassRole
- Voglio consentire a persone esterne a me di accedere Account AWS alle mie risorse QLDB

Non sono autorizzato a eseguire un'azione in QLDB

Se ti AWS Management Console dice che non sei autorizzato a eseguire un'azione, devi contattare l'amministratore per ricevere assistenza. L'amministratore è colui che ti ha fornito le credenziali di accesso.

Il seguente esempio di errore si verifica quando l'utente `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`myExampleLedger`* fittizia, ma non dispone di autorizzazioni `qldb:`*`DescribeLedger`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 qldb:DescribeLedger on resource: myExampleLedger
```

In questo caso, Mateo richiede al suo amministratore di aggiornare le policy per poter accedere alla risorsa *`myExampleLedger`* utilizzando l'operazione `qldb:`*`DescribeLedger`*.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un messaggio di errore indicante che non sei autorizzato a eseguire l'`iam:PassRole`azione, le tue policy devono essere aggiornate per consentirti di trasferire un ruolo a QLDB.

Alcuni Servizi AWS consentono di trasferire un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

Il seguente errore di esempio si verifica quando un utente IAM denominato `marymajor` tenta di utilizzare la console per eseguire un'azione in QLDB. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per passare il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Per una guida alla risoluzione di questo errore specifico relativo alle operazioni di esportazione o di streaming del diario, vedereRisoluzione dei problemi di Amazon QLDB.

Voglio consentire a persone esterne a me di accedere Account AWS alle mie risorse QLDB

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per servizi che supportano policy basate su risorse o liste di controllo accessi (ACL), utilizza tali policy per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per sapere se QLDB supporta queste funzionalità, vedere. Come funziona Amazon QLDB con IAM
- Per scoprire come fornire l'accesso alle tue risorse su tutto Account AWS ciò che possiedi, consulta Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a soggetti Account AWS di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (Federazione delle identità) nella Guida per l'utente di IAM.
- Per informazioni sulle differenze tra l'utilizzo di ruoli e policy basate su risorse per l'accesso multi-account, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente IAM.

Registrazione e monitoraggio in Amazon QLDB

Il monitoraggio è una parte importante per mantenere l'affidabilità, la disponibilità e le prestazioni di Amazon QLDB e delle tue soluzioni. AWS È necessario raccogliere i dati di monitoraggio da tutte le parti della AWS soluzione in modo da poter eseguire più facilmente il debug di un errore multipunto, se si verifica. Tuttavia, prima di iniziare a monitorare QLDB, è necessario creare un piano di monitoraggio che includa le risposte alle seguenti domande:

- Quali sono gli obiettivi del monitoraggio?
- Di quali risorse si intende eseguire il monitoraggio?
- Con quale frequenza sarà eseguito il monitoraggio di queste risorse?

- Quali strumenti di monitoraggio verranno utilizzati?
- Chi eseguirà i processi di monitoraggio?
- Chi deve ricevere una notifica quando si verifica un problema?

Il passaggio successivo consiste nello stabilire una linea di base per le normali prestazioni QLDB nell'ambiente, misurando le prestazioni in diversi momenti e in diverse condizioni di carico. Durante il monitoraggio di QLDB, archivia i dati di monitoraggio storici in modo da poterli confrontare con i dati sulle prestazioni correnti, identificare i modelli di prestazioni normali e le anomalie delle prestazioni e ideare metodi per risolvere i problemi.

Per stabilire una baseline, è necessario monitorare almeno gli elementi seguenti:

- Leggi e scrivi I/O e archiviazione, in modo da poter tenere traccia dei modelli di consumo del registro ai fini della fatturazione.
- Latenza dei comandi, in modo da poter monitorare le prestazioni del registro durante l'esecuzione delle operazioni sui dati.
- Eccezioni, in modo da poter determinare se alcune richieste hanno generato un errore.

Argomenti

Strumenti di monitoraggio

AWS fornisce diversi strumenti che puoi utilizzare per monitorare Amazon QLDB. Alcuni di questi strumenti possono essere configurati in modo che eseguano automaticamente il monitoraggio, mentre altri richiedono l'intervento manuale. Si consiglia di automatizzare il più possibile i processi di monitoraggio.

Argomenti

Strumenti di monitoraggio automatici

Puoi utilizzare i seguenti strumenti di monitoraggio automatizzato per guardare QLDB e segnalare quando qualcosa non va:

- Amazon CloudWatch Alarms: monitora una singola metrica in un periodo di tempo specificato ed esegui una o più azioni in base al valore della metrica rispetto a una determinata soglia in diversi periodi di tempo. L'azione è una notifica inviata a un argomento di Amazon Simple Notification Service (Amazon SNS) o a una policy di Amazon EC2 Auto Scaling. CloudWatch gli allarmi non richiamano azioni semplicemente perché si trovano in uno stato particolare; lo stato deve essere cambiato e mantenuto per un determinato numero di periodi. Per ulteriori informazioni, consulta Monitoraggio con Amazon CloudWatch.
- Amazon CloudWatch Logs: monitora, archivia e accedi ai tuoi file di registro da AWS CloudTrail o altre fonti. Per ulteriori informazioni, consulta Monitoring log files nella Amazon CloudWatch User Guide.
- Amazon CloudWatch Events: abbina gli eventi e li indirizza a una o più funzioni o stream di destinazione per apportare modifiche, acquisire informazioni sullo stato e intraprendere azioni correttive. Per ulteriori informazioni, consulta What is Amazon CloudWatch Events nella Amazon CloudWatch User Guide.
- AWS CloudTrail Monitoraggio dei log: condividi i file di CloudTrail registro tra account, monitora i file di registro in tempo reale inviandoli a CloudWatch Logs, scrivi applicazioni di elaborazione dei log in Java e verifica che i file di registro non siano cambiati dopo la consegna da parte di. CloudTrail Per ulteriori informazioni, consulta Lavorare con i file di CloudTrail registro nella Guida per l'AWS CloudTrail utente.

Strumenti di monitoraggio manuali

Un'altra parte importante del monitoraggio di QLDB consiste nel monitorare manualmente gli elementi che gli allarmi non CloudWatch coprono. QLDB CloudWatch e AWS Management Console altri dashboard forniscono at-a-glance una visione dello stato dell'ambiente. Trusted Advisor AWS Ti consigliamo di controllare anche i file di registro su Amazon QLDB.

- Il pannello di controllo QLDB mostra quanto segue:
  - I/O di lettura e scrittura
  - Archiviazione a diario e indicizzata
  - Latenza dei comandi

  - Eccezioni
- La CloudWatch home page mostra quanto segue:
  - Stato e allarmi attuali
  - Grafici degli allarmi e delle risorse
  - Stato di integrità dei servizi

  Inoltre, è possibile utilizzare CloudWatch per effettuare le seguenti operazioni:
  - Crea pannelli di controllo personalizzati per monitorare i servizi di interesse.
  - Crea grafici dei dati dei parametri per la risoluzione di problemi e il rilevamento di tendenze.
  - Cerca e sfoglia tutte le metriche AWS delle tue risorse
  - Crea e modifica gli allarmi per ricevere le notifiche dei problemi.

Monitoraggio con Amazon CloudWatch

Puoi monitorare Amazon QLDB CloudWatch utilizzando, che raccoglie ed elabora dati grezzi da Amazon QLDB in parametri leggibili. near-real-time Registra queste statistiche per due settimane in modo da poter accedere alle informazioni storiche e avere una prospettiva migliore sulle prestazioni della tua applicazione o del tuo servizio web. Per impostazione predefinita, i dati delle metriche QLDB vengono inviati automaticamente in periodi di 1 o CloudWatch 15 minuti. Per ulteriori informazioni, consulta Cosa sono Amazon CloudWatch, Amazon CloudWatch Events e Amazon CloudWatch Logs? nella Amazon CloudWatch User Guide.

Argomenti

- Come posso usare le metriche QLDB?
- Metriche e dimensioni di Amazon QLDB
- Creazione di CloudWatch allarmi per monitorare Amazon QLDB

Come posso usare le metriche QLDB?

Le metriche riportate da QLDB forniscono informazioni che è possibile analizzare in diversi modi. L'elenco seguente mostra alcuni usi comuni dei parametri. Questi suggerimenti sono solo introduttivi e non costituiscono un elenco completo.

- È possibile monitorare `JournalStorage` e, `IndexedStorage` per un periodo di tempo specificato, tenere traccia della quantità di spazio su disco consumata dal registro.

- È possibile monitorare `ReadIOs` e`WriteIOs`, per un periodo di tempo specificato, tenere traccia del numero di richieste elaborate dal registro.
- È possibile monitorare `CommandLatency` per tenere traccia delle prestazioni del registro per quanto riguarda le operazioni sui dati e analizzare i tipi di comandi che generano la maggiore latenza.

Metriche e dimensioni di Amazon QLDB

Quando interagisci con Amazon QLDB, invia le seguenti metriche e dimensioni a. CloudWatch I parametri di storage vengono segnalati ogni 15 minuti e tutti gli altri parametri vengono aggregati e riportati ogni minuto. È possibile utilizzare le seguenti procedure per visualizzare le metriche per QLDB.

Per visualizzare le metriche utilizzando la console CloudWatch

I parametri vengono raggruppati prima in base allo spazio dei nomi del servizio e successivamente in base alle diverse combinazioni di dimensioni all'interno di ogni spazio dei nomi.

1. Apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Se necessario, modificare la regione . Nella barra di navigazione, scegli la regione in cui risiedono AWS le tue risorse. Per ulteriori informazioni, consulta Regioni ed endpoint.
3. Nel pannello di navigazione, seleziona Parametri.
4. Nella scheda Tutte le metriche, scegli QLDB.

Per visualizzare le metriche utilizzando il AWS CLI

- Al prompt dei comandi utilizza il comando seguente.

```
aws cloudwatch list-metrics --namespace "AWS/QLDB"
```

CloudWatch visualizza le seguenti metriche per QLDB.

Dimensioni e parametri di Amazon QLDB

Le metriche e le dimensioni che Amazon QLDB invia ad CloudWatch Amazon sono elencate qui.

Metriche QLDB

Parametro	Descrizione
`JournalStorage`	La quantità totale di spazio su disco utilizzata dal diario del libro mastro, riportata a intervalli di 15 minuti. Il diario contiene la cronologia completa, immutabile e verificabile di tutte le modifiche ai dati. Unità: `Bytes` Dimensioni: `LedgerName`
`IndexedStorage`	La quantità totale di spazio su disco utilizzata dalle tabelle, dagli indici e dalla cronologia indicizzata del registro, riportata a intervalli di 15 minuti. Lo storage indicizzato è costituito da dati di registro ottimizzati per query ad alte prestazioni. Unità: `Bytes` Dimensioni: `LedgerName`
`ReadIOs`	Il numero di richieste di I/O di lettura, riportate a intervalli di un minuto. Questo consente di acquisire tutti i tipi di operazioni di lettura, incluse transazioni di dati, richieste di verifica, esportazioni di journal e stream di journal. Unità: `Count` Dimensioni: `LedgerName`
`WriteIOs`	Il numero di richieste I/O di scrittura, riportate a intervalli di un minuto. Unità: `Count` Dimensioni: `LedgerName`

Parametro	Descrizione
`CommandLatency`	La quantità di tempo impiegata per le operazioni sui dati, riportata a intervalli di un minuto. Unità: `Milliseconds` Dimensioni: `CommandType, LedgerName`
`IsImpaired`	Il flag che indica se lo streaming di un journal verso Kinesis Data Streams è compromesso, riportato a intervalli di un minuto. Il valore di `1` indica che lo stream è in stato interrotto e indica il contrario. `0` Unità: `Boolean` (0 o 1) Dimensioni: `LedgerName, StreamId`
`OccConflictExceptions`	Il numero di richieste a QLDB che generano un. `OccConflictException` Per informazioni sul controllo ottimistico della concorrenza (OCC), vedere. Modello di concorrenza Amazon QLDB Unità: `Count`
`Session4xxExceptions`	Il numero di richieste a QLDB che generano un errore HTTP 4xx. Unità: `Count`
`Session5xxExceptions`	Il numero di richieste a QLDB che generano un errore HTTP 5xx. Unità: `Count`
`SessionRateExceededExceptions`	Il numero di richieste a QLDB che generano un. `SessionRateExceededException` Unità: `Count`

Dimensioni per le metriche QLDB

Le metriche per QLDB sono qualificate in base ai valori dell'account, del nome del registro, dell'ID dello stream o del tipo di comando. È possibile utilizzare la CloudWatch console per recuperare i dati QLDB lungo una qualsiasi delle dimensioni nella tabella seguente.

Dimensione	Descrizione
`LedgerName`	Questa dimensione limita i dati a un registro specifico. Questo valore può essere qualsiasi nome di libro contabile nel registro corrente Regione AWS e in quello corrente. Account AWS
`StreamId`	Questa dimensione limita i dati a un flusso di diario specifico. Questo valore può essere qualsiasi ID di stream per un registro corrente Regione AWS e corrente Account AWS.
`CommandType`	Questa dimensione limita i dati a uno dei seguenti comandi API dati QLDB: • `AbortTransaction` • `CommitTransaction` • `EndSession` • `ExecuteStatement` • `FetchPage` • `StartSession` • `StartTransaction` Per informazioni su come QLDB utilizza questi comandi per gestire le operazioni sui dati, vedere. Gestione delle sessioni con il conducente

Creazione di CloudWatch allarmi per monitorare Amazon QLDB

Puoi creare un CloudWatch allarme Amazon che invia un messaggio Amazon Simple Notification Service (Amazon SNS) quando l'allarme cambia stato. Un allarme monitora un singolo parametro per un periodo di tempo specificato. L'allarme esegue una o più operazioni basate sul valore del

parametro relativo a una soglia prestabilita per un certo numero di periodi. L'operazione corrisponde all'invio di una notifica a un argomento di Amazon SNS o a una policy di Auto Scaling.

Gli allarmi richiamano azioni solo per cambiamenti di stato sostenuti. CloudWatch gli allarmi non richiamano azioni semplicemente perché si trovano in uno stato particolare. Lo stato deve essere cambiato e restare costante per un numero specificato di periodi.

Per ulteriori informazioni sulla creazione di CloudWatch allarmi, consulta Using Amazon CloudWatch alarms nella Amazon CloudWatch User Guide.

Automazione di Amazon QLDB con eventi CloudWatch

Amazon CloudWatch Events ti consente di automatizzare Servizi AWS e rispondere automaticamente a eventi di sistema come problemi di disponibilità delle applicazioni o modifiche delle risorse. Gli eventi di Servizi AWS vengono trasmessi a CloudWatch Events quasi in tempo reale. Puoi compilare regole semplici che indichino quali eventi sono considerati di interesse per te e quali azioni automatizzate intraprendere quando un evento corrisponde a una regola. Le azioni che possono essere attivate automaticamente includono le seguenti:

- Invocare una funzione AWS Lambda
- Richiamo del comando di esecuzione di Amazon EC2
- Inoltro dell'evento a Amazon Kinesis Data Streams
- Attivazione di una macchina a stati AWS Step Functions
- Notifica di un argomento Amazon SNS o di una coda Amazon SQS

Amazon QLDB segnala un evento CloudWatch a Events ogni volta che lo stato di una risorsa di registro nelle tue pagine cambia. Account AWS Gli eventi vengono attualmente emessi su at-least-once base garantita, solo per le risorse del registro QLDB.

Di seguito è riportato un esempio di evento riportato da QLDB, in cui lo stato di un registro è cambiato in. `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"
    }
}
```

Alcuni esempi di utilizzo di CloudWatch Events with QLDB possono includere, a titolo esemplificativo ma non esaustivo, quanto segue:

- Attivazione di una funzione Lambda ogni volta che un nuovo registro viene inizialmente creato `CREATING` in uno stato e alla fine lo diventa. `ACTIVE`
- Notifica di un argomento di Amazon SNS quando lo stato del registro cambia `DELETING` e poi lo stesso. `DELETED`

Per ulteriori informazioni, consulta la Amazon CloudWatch Events User Guide.

Registrazione delle chiamate API Amazon QLDB con AWS CloudTrail

Amazon QLDB è integrato AWS CloudTrail con un servizio che fornisce un registro delle azioni intraprese da un utente, un ruolo o un QLDB interno. Servizio AWS CloudTrail acquisisce tutte le chiamate API di gestione delle risorse per QLDB come eventi. Le chiamate acquisite includono chiamate dalla console QLDB e chiamate di codice alle operazioni dell'API QLDB. Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon Simple Storage Service (Amazon S3), inclusi gli eventi per QLDB. Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti sulla CloudTrail console nella cronologia degli eventi. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare la richiesta che è stata effettuata a QLDB, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e dettagli aggiuntivi.

Per saperne di più CloudTrail, incluso come configurarlo e abilitarlo, consulta la Guida per l'AWS CloudTrail utente.

Informazioni QLDB in CloudTrail

CloudTrail è abilitato sul tuo account al Account AWS momento della creazione dell'account. Quando si verifica un'attività supportata in QLDB, tale attività viene registrata in CloudTrail un evento insieme ad Servizio AWS altri eventi nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare

eventi recenti in. Account AWS Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi nel tuo Account AWS, compresi gli eventi per QLDB, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando si crea un percorso nella console, questo sarà valido in tutte le Regioni AWS. Il trail registra gli eventi di tutte le regioni della AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Inoltre, puoi configurarne altri Servizi AWS per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail

Per ulteriori informazioni, consulta gli argomenti seguenti nella Guida per l'utente di AWS CloudTrail :

- Panoramica della creazione di un percorso
- CloudTrail servizi e integrazioni supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni
- Ricezione di file di CloudTrail registro da più account

Tutte le azioni API per la gestione delle risorse QLDB e i dati non transazionali vengono registrate e CloudTrail sono documentate nel riferimento dell'API Amazon QLDB. Ad esempio, le chiamate a, e le azioni generano voci nei `CreateLedger` file di `DescribeLedger` registro. `DeleteLedger` CloudTrail

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali utente o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro Servizio AWS

Per ulteriori informazioni, vedete l'elemento CloudTrail userIdentity.

Informazioni sulle voci dei file di registro QLDB

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro.

Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'azione richiesta, la data e l'ora dell'azione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra una voce di CloudTrail registro che illustra queste azioni:

- `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"
    }
  ]
}
```


Convalida della conformità per Amazon QLDB

I revisori di terze parti valutano la sicurezza e la conformità di Amazon QLDB nell'ambito di AWS diversi programmi di conformità, tra cui, a titolo esemplificativo ma non esaustivo, i seguenti:

- System and Organization Controls (SOC)
- Payment Card Industry (PCI)
- International Organization for Standardization (ISO)
- Programma di gestione e valutazione della sicurezza dei sistemi informativi (ISMAP)
- Health Insurance Portability and Accountability Act (HIPAA)

> **Note**
>
> Questo non è un elenco completo delle certificazioni Amazon QLDB.

Per sapere se una Servizio AWS rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione Scope by Compliance Program Servizi AWS e scegli il programma di conformità che ti interessa. Per informazioni generali, consulta Programmi di AWS conformità Programmi di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La vostra responsabilità di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. AWS fornisce le seguenti risorse per contribuire alla conformità:

- Guide introduttive su sicurezza e conformità: queste guide all'implementazione illustrano considerazioni sull'architettura e forniscono passaggi per implementare ambienti di base incentrati sulla AWS sicurezza e la conformità.
- Progettazione per la sicurezza e la conformità HIPAA su Amazon Web Services: questo white paper descrive in che modo le aziende possono utilizzare AWS per creare applicazioni idonee all'HIPAA.

  > **Note**
  >
  > Non Servizi AWS tutte sono idonee all'HIPAA. Per ulteriori informazioni, consulta la sezione Riferimenti sui servizi conformi ai requisiti HIPAA.

- AWS Risorse per la per la conformità: questa raccolta di cartelle di lavoro e guide potrebbe essere valida per il tuo settore e la tua località.
- AWS Guide alla conformità dei clienti: comprendi il modello di responsabilità condivisa attraverso la lente della conformità. Le guide riassumono le migliori pratiche per la protezione Servizi AWS e mappano le linee guida per i controlli di sicurezza su più framework (tra cui il National Institute of Standards and Technology (NIST), il Payment Card Industry Security Standards Council (PCI) e l'International Organization for Standardization (ISO)).

- Valutazione delle risorse con regole nella Guida per gli AWS Config sviluppatori: il AWS Config servizio valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida e alle normative del settore.
- AWS Security Hub— Ciò Servizio AWS fornisce una visione completa dello stato di sicurezza interno. AWS La Centrale di sicurezza utilizza i controlli di sicurezza per valutare le risorse AWS e verificare la conformità agli standard e alle best practice del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta la pagina Documentazione di riferimento sui controlli della Centrale di sicurezza.
- Amazon GuardDuty: Servizio AWS rileva potenziali minacce ai tuoi carichi di lavoro Account AWS, ai contenitori e ai dati monitorando l'ambiente alla ricerca di attività sospette e dannose. GuardDuty può aiutarti a soddisfare vari requisiti di conformità, come lo standard PCI DSS, soddisfacendo i requisiti di rilevamento delle intrusioni imposti da determinati framework di conformità.
- AWS Audit Manager— Ciò Servizio AWS consente di verificare continuamente l' AWS utilizzo per semplificare la gestione del rischio e la conformità alle normative e agli standard di settore.

Resilienza in Amazon QLDB

L'infrastruttura AWS globale è costruita attorno a zone di disponibilità. Regioni AWS Regioni AWS forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti. Con le zone di disponibilità, è possibile progettare e gestire applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture tradizionali a data center singolo o multiplo.

Per ulteriori informazioni sulle zone di disponibilità, vedere Global Regioni AWS Infrastructure.AWS

Durabilità dello storage

L'archiviazione del journal QLDB offre la replica sincrona su più zone di disponibilità sui commit delle transazioni. Ciò garantisce che anche un guasto completo della zona di disponibilità dello storage del journal non comprometterebbe l'integrità dei dati o la capacità di mantenere un servizio attivo. Inoltre, il journal QLDB offre archivi asincroni su storage con tolleranza ai guasti. Questa funzionalità supporta il disaster recovery nell'eventualità altamente improbabile di un errore di archiviazione simultaneo per più zone di disponibilità.

Lo storage indicizzato QLDB è supportato dalla replica su più zone di disponibilità. Ciò garantisce che anche un guasto completo della zona di disponibilità dello storage indicizzato non comprometterebbe l'integrità dei dati o la capacità di mantenere un servizio attivo.

Funzionalità di durabilità dei dati

Oltre all'infrastruttura AWS globale, QLDB offre le seguenti funzionalità per supportare le esigenze di resilienza e backup dei dati.

Funzionalità del servizio QLDB

Esportazione di diari su richiesta

QLDB offre una funzionalità di esportazione del diario su richiesta. Accedi ai contenuti del tuo diario esportando blocchi di journal dal tuo registro in un bucket Amazon S3. Puoi utilizzare questi dati per vari scopi, come la conservazione dei dati, l'analisi e il controllo. Per ulteriori informazioni, consulta Esportazione dei dati del diario da Amazon QLDB.

Backup e ripristino

Il ripristino automatico delle esportazioni non è attualmente supportato. L'esportazione offre una funzionalità di base per creare una ridondanza aggiuntiva dei dati alla frequenza definita. Tuttavia, uno scenario di ripristino dipende dall'applicazione, in cui i record esportati vengono presumibilmente riscritti in un nuovo registro utilizzando lo stesso metodo di inserimento o un metodo di inserimento simile.

Al momento QLDB non fornisce una funzionalità di backup dedicata e relativa funzionalità di ripristino.

Stream del diario

QLDB offre anche una funzionalità di flusso continuo del journal. Puoi integrare i flussi di journal QLDB con la piattaforma di streaming Amazon Kinesis per elaborare i dati del journal in tempo reale. Per ulteriori informazioni, consulta Streaming dei dati del diario da Amazon QLDB.

Funzionalità di progettazione QLDB

QLDB è progettato per resistere alla corruzione logica. Il journal QLDB è immutabile e garantisce che tutte le transazioni impegnate vengano mantenute nel journal. Inoltre, ogni modifica al documento confermata viene registrata, in quanto ciò consente la point-in-timevisibilità di eventuali modifiche non intenzionali ai dati del registro.

Al momento, QLDB non fornisce una funzionalità di ripristino automatico per scenari di danneggiamento logico.

Sicurezza dell'infrastruttura in Amazon QLDB

In quanto servizio gestito, Amazon QLDB è protetto dalle AWS procedure di sicurezza di rete globali descritte nel white paper Amazon Web Services: Overview of Security Processes.

Utilizzi chiamate API AWS pubblicate per accedere a QLDB attraverso la rete. I client devono supportare Transport Layer Security (TLS) 1.0 o versioni successive. È consigliabile TLS 1.2 o versioni successive. I client devono, inoltre, supportare le suite di cifratura con PFS (Perfect Forward Secrecy), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando credenziali programmatiche associate a un principale IAM. O puoi utilizzare AWS Security Token Service (AWS STS) per generare credenziali di sicurezza temporanee per sottoscrivere le richieste.

Puoi anche utilizzare un endpoint VPC (Virtual Private Cloud) per QLDB. Gli endpoint VPC di interfaccia consentono alle risorse Amazon VPC di utilizzare i loro indirizzi IP privati per accedere a QLDB senza esposizione alla rete Internet pubblica. Per ulteriori informazioni, consulta Accedi ad Amazon QLDB utilizzando un endpoint di interfaccia ()AWS PrivateLink.

Accedi ad Amazon QLDB utilizzando un endpoint di interfaccia ()AWS PrivateLink

Puoi usarlo AWS PrivateLink per creare una connessione privata tra il tuo VPC e Amazon QLDB. Puoi accedere a QLDB come se fosse nel tuo VPC, senza l'uso di un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione. AWS Direct Connect Le istanze nel tuo VPC non necessitano di indirizzi IP pubblici per accedere a QLDB.

Stabilisci questa connessione privata creando un endpoint di interfaccia attivato da AWS PrivateLink. In ciascuna sottorete viene creata un'interfaccia di rete endpoint da abilitare per l'endpoint di interfaccia. Si tratta di interfacce di rete gestite dai richiedenti che fungono da punto di ingresso per il traffico destinato a QLDB.

Per ulteriori informazioni, consulta la sezione Accesso a Servizi AWS tramite AWS PrivateLink nella Guida di AWS PrivateLink .

Argomenti

Considerazioni per QLDB

Prima di configurare un endpoint di interfaccia per QLDB, consulta le considerazioni nella Guida.AWS PrivateLink

> **Note**
>
> QLDB supporta solo le chiamate all'API dei dati transazionali della sessione QLDB tramite l'endpoint dell'interfaccia. Questa API include solo l'operazione. SendCommand Nella modalità `STANDARD` autorizzazioni di un registro, puoi controllare le autorizzazioni per azioni PartiQL specifiche in questa API.

Creare un endpoint di interfaccia per QLDB

Puoi creare un endpoint di interfaccia per QLDB utilizzando la console Amazon VPC o (). AWS Command Line Interface AWS CLI Per ulteriori informazioni, consulta la sezione Creazione di un endpoint di interfaccia nella Guida per l'utente di AWS PrivateLink .

Crea un endpoint di interfaccia per QLDB utilizzando il seguente nome di servizio:

```
com.amazonaws.region.qldb.session
```

Se abiliti il DNS privato per l'endpoint dell'interfaccia, puoi effettuare richieste API a QLDB utilizzando il nome DNS regionale predefinito. Ad esempio, `session.qldb.us-east-1.amazonaws.com`.

Creazione di una policy dell' endpoint per l'endpoint dell'interfaccia

Una policy dell'endpoint è una risorsa IAM che è possibile allegare all'endpoint dell'interfaccia. La policy endpoint predefinita consente l'accesso completo a QLDB tramite l'endpoint dell'interfaccia. Per controllare l'accesso consentito a QLDB dal tuo VPC, collega una policy personalizzata per gli endpoint all'endpoint dell'interfaccia.

Una policy di endpoint specifica le informazioni riportate di seguito:

- I principali che possono eseguire azioni (Account AWS utenti e ruoli).
- Le azioni che possono essere eseguite.
- Le risorse in cui è possibile eseguire le operazioni.

Per ulteriori informazioni, consulta la sezione Controllo dell'accesso ai servizi con policy di endpoint nella Guida di AWS PrivateLink .

È inoltre possibile utilizzare il `Condition` campo in una policy associata a un utente, gruppo o ruolo per consentire l'accesso solo da un endpoint di interfaccia specificato. Se utilizzate insieme, le policy degli endpoint e le policy IAM possono limitare l'accesso a specifiche azioni QLDB su registri specifici a un endpoint di interfaccia specificato.

Esempio di policy per gli endpoint: limitare l'accesso a un registro QLDB specifico

Di seguito è riportato un esempio di policy endpoint personalizzata per QLDB. Quando colleghi questa policy all'endpoint dell'interfaccia, concede l'accesso all'`SendCommand`azione e alle azioni di sola lettura PartiQL per tutti i principali sulla risorsa di registro specificata. In questo esempio, il registro deve essere in modalità autorizzazioni. `STANDARD`

Per utilizzare questa politica, sostituisci *`us-east-1,`* 123456789012 *`myExampleLedger`*e nell'esempio con le tue informazioni.

```
{
   "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"
         ]
      }
   ]
}
```

Esempio di policy IAM: limita l'accesso a un registro QLDB solo da un endpoint di interfaccia specifico

Di seguito è riportato un esempio di policy basata sull'identità IAM per QLDB. Quando si collega questa politica a un utente, ruolo o gruppo, consente l'`SendCommand`accesso a una risorsa di registro solo dall'endpoint di interfaccia specificato.

Per utilizzare questa politica, sostituisci *`us-east-1,`* 123456789012 *`e`* vpce-1a2b3c4d nell'*`myExampleLedger`*esempio con le tue informazioni.

```
{
   "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"
            }
         }
      }
   ]
}
```

Disponibilità di endpoint di interfaccia per QLDB

Amazon QLDB supporta endpoint di interfaccia con policy in tutti i paesi in Regioni AWS cui è disponibile QLDB. Per un elenco completo delle regioni disponibili, consulta gli endpoint e le quote di Amazon QLDB nel. Riferimenti generali di AWS

Risoluzione dei problemi di Amazon QLDB

Le seguenti sezioni forniscono un elenco aggregato degli errori comuni che potresti riscontrare quando utilizzi Amazon QLDB e una guida su come risolverli.

Per una guida alla risoluzione dei problemi specifica per l'accesso IAM, vedereRisoluzione dei problemi relativi all'identità e all'accesso ad Amazon QLDB.

Per le best practice per l'ottimizzazione delle istruzioni PartiQL, vedereOttimizzazione delle prestazioni delle query.

Argomenti

Esecuzione di transazioni utilizzando il driver QLDB

Questa sezione elenca le eccezioni comuni che il driver Amazon QLDB può restituire quando lo si utilizza per eseguire transazioni PartiQL su un libro mastro. Per ulteriori informazioni sull'utilizzo di questa caratteristica, consulta Nozioni base sul driver. Per le best practice per la configurazione e l'utilizzo del driver, vedereConsigli per i.

Ogni eccezione include il messaggio di errore specifico, seguito da una breve descrizione e suggerimenti per possibili soluzioni.

CapacityExceededException

Messaggio: capacità superata

Amazon QLDB ha rifiutato la richiesta perché superava la capacità di elaborazione del libro mastro. QLDB impone un limite di scalabilità interno per registro per mantenere l'integrità e le prestazioni del servizio. Questo limite varia in base alle dimensioni del carico di lavoro di ogni singola richiesta. Ad esempio, una richiesta può comportare un aumento del carico di lavoro se esegue transazioni di dati inefficienti, ad esempio scansioni di tabelle risultanti da una query qualificata non indicizzata.

Ti consigliamo di attendere prima di riprovare la richiesta. Se la tua candidatura riscontra costantemente questa eccezione, ottimizza i rendiconti e riduci la frequenza e il volume delle richieste che invii al libro mastro. Esempi di ottimizzazione dei rendiconti includono l'esecuzione di un minor numero di rendiconti per transazione e l'ottimizzazione degli indici delle tabelle. Per informazioni su come ottimizzare le dichiarazioni ed evitare le scansioni delle tabelle, vedereOttimizzazione delle prestazioni delle query.

Si consiglia inoltre di utilizzare l'ultima versione del driver QLDB. Il driver ha una politica di riprova predefinita che utilizza Exponential Backoff e Jitter per riprovare automaticamente eccezioni come questa. Il concetto di backoff esponenziale consiste nell'utilizzare tempi di attesa progressivamente più lunghi tra nuovi tentativi per risposte di errore consecutive.

InvalidSessionException

Messaggio: *`TransactionID della`* transazione è scaduto

Una transazione ha superato la durata massima. Una transazione può durare fino a 30 secondi prima di essere confermata. Dopo questo limite di timeout, qualsiasi lavoro svolto sulla transazione viene rifiutato e QLDB annulla la sessione. Questo limite protegge il cliente dalla divulgazione delle sessioni avviando transazioni e non eseguendole o annullandole.

Se si tratta di un'eccezione comune nella tua applicazione, è probabile che le transazioni richiedano semplicemente troppo tempo per essere eseguite. Se la durata della transazione richiede più di 30 secondi, ottimizza i rendiconti per velocizzare le transazioni. Esempi di ottimizzazione dei rendiconti includono l'esecuzione di un minor numero di rendiconti per transazione e l'ottimizzazione degli indici delle tabelle. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.

InvalidSessionException

Messaggio: l'*`sessionId`* è scaduto

QLDB ha scartato la sessione perché ha superato la durata massima totale. QLDB scarta le sessioni dopo 13-17 minuti, indipendentemente da una transazione attiva. Le sessioni possono essere perse o compromesse per diversi motivi, ad esempio guasti hardware, errori di rete o riavvii delle applicazioni. Pertanto, QLDB impone una durata massima delle sessioni per garantire che il software client sia resistente agli errori di sessione.

Se riscontri questa eccezione, ti consigliamo di acquisire una nuova sessione e riprovare la transazione. Si consiglia inoltre di utilizzare l'ultima versione del driver QLDB, che gestisce il pool di sessioni e il relativo stato di salute per conto dell'applicazione.

InvalidSessionException

Messaggio: sessione inesistente

Il cliente ha provato a effettuare transazioni con QLDB utilizzando una sessione che non esiste. Supponendo che il client stia utilizzando una sessione che esisteva in precedenza, la sessione potrebbe non esistere più a causa di uno dei seguenti motivi:

- Se una sessione è coinvolta in un errore interno del server (ovvero un errore con il codice di risposta HTTP 500), QLDB potrebbe scegliere di annullare completamente la sessione, anziché consentire al cliente di effettuare transazioni con una sessione in stato incerto. Quindi, qualsiasi tentativo di nuovo su quella sessione fallisce con questo errore.
- Le sessioni scadute vengono infine dimenticate da QLDB. Quindi, qualsiasi tentativo di continuare a utilizzare la sessione genera questo errore, anziché l'iniziale`InvalidSessionException`.

Se riscontri questa eccezione, ti consigliamo di acquisire una nuova sessione e riprovare la transazione. Si consiglia inoltre di utilizzare l'ultima versione del driver QLDB, che gestisce il pool di sessioni e il relativo stato di salute per conto dell'applicazione.

RateExceededException

Messaggio: la percentuale è stata superata

QLDB ha limitato un client in base all'identità del chiamante. QLDB impone la limitazione per regione e per account utilizzando un algoritmo di limitazione del bucket di token. QLDB lo fa per aiutare le prestazioni del servizio e per garantire un utilizzo equo a tutti i clienti QLDB. Ad esempio, il tentativo di acquisire un gran numero di sessioni simultanee utilizzando l'`StartSessionRequest`operazione potrebbe portare a una limitazione.

Per mantenere l'integrità dell'applicazione e mitigare ulteriori limitazioni, è possibile riprovare questa eccezione utilizzando Exponential Backoff e Jitter. Il concetto di backoff esponenziale consiste nell'utilizzare tempi di attesa progressivamente più lunghi tra nuovi tentativi per risposte di errore consecutive. Si consiglia di utilizzare l'ultima versione del driver QLDB. Il driver ha una politica di riprova predefinita che utilizza il backoff e il jitter esponenziali per riprovare automaticamente eccezioni come questa.

L'ultima versione del driver QLDB può essere utile anche se l'applicazione viene costantemente limitata da QLDB per`StartSessionRequest` le chiamate. Il driver gestisce un pool di sessioni che vengono riutilizzate tra le transazioni, il che può aiutare a ridurre il numero

di`StartSessionRequest` chiamate effettuate dall'applicazione. Per richiedere un aumento dei limiti di throttling delle API, contatta il AWS SupportCentro.

LimitExceededException

Messaggio: è stato superato il limite di sessione

Un libro mastro ha superato la sua quota (nota anche come limite) sul numero di sessioni attive. Questa quota è definita inQuote e limiti in Amazon QLDB. Il conteggio delle sessioni attive di un libro mastro alla fine è costante e i registri che si avvicinano costantemente alla quota potrebbero periodicamente vedere questa eccezione.

Per mantenere l'integrità dell'applicazione, ti consigliamo di riprovare con questa eccezione. Per evitare questa eccezione, assicurati di non aver configurato più di 1.500 sessioni simultanee da utilizzare per un singolo libro mastro per tutti i client. Ad esempio, puoi utilizzare il maxConcurrentTransactionsmetodo del driver Amazon QLDB per Java per configurare il numero massimo di sessioni disponibili in un'istanza di driver.

QldbClientException

Messaggio: un risultato trasmesso in streaming è valido solo quando la transazione principale è aperta

La transazione è chiusa e non può essere utilizzata per recuperare i risultati da QLDB. Una transazione si chiude quando viene confermata o annullata.

Questa eccezione si verifica quando il cliente lavora direttamente con l'`Transaction`oggetto e sta cercando di recuperare i risultati da QLDB dopo aver commesso o annullato una transazione. Per mitigare questo problema, il cliente deve leggere i dati prima di chiudere la transazione.

Esportazione dei dati del giornale

Questa sezione elenca le eccezioni comuni che QLDB può restituire quando si esportano i dati delle scritture contabili da un registro in un bucket Amazon S3. Per ulteriori informazioni sull'utilizzo di questa caratteristica, consulta Esportazione dei dati del diario da Amazon QLDB.

Ogni eccezione include il messaggio di errore specifico, seguito da una breve descrizione e suggerimenti per possibili soluzioni.

AccessDeniedException

Messaggio: Utente: *UserArn* non è autorizzato a eseguire: iam:PassRole nessuna risorsa: *roleARN*

non disponi delle autorizzazioni a trasferire un ruolo IAM al servizio QLDB. QLDB richiede un ruolo per tutte le richieste di esportazione dei diari ed è necessario disporre delle autorizzazioni per passare questo ruolo a QLDB. Il ruolo fornisce a QLDB le autorizzazioni di scrittura nel bucket Amazon S3 specificato.

Verifica di definire una politica IAM che conceda l'autorizzazione a eseguire l'operazione`PassRole` API sulla risorsa di ruolo IAM specificata per il servizio QLDB (`qldb.amazonaws.com`). Per un esempio di policy, consulta Esempi di policy basate sull'identità per Amazon QLDB.

IllegalArgumentException

Messaggio: QLDB ha riscontrato un errore durante la convalida della configurazione S3: *ErrorCode ErrorMessage*

Una possibile causa di questo errore è che il bucket Amazon S3 fornito non esiste in Amazon S3. In alternativa, QLDB non dispone di autorizzazioni sufficienti per scrivere oggetti nel bucket Amazon S3 specificato.

Verifica che il nome del bucket S3 fornito nella richiesta di lavoro di esportazione sia corretto. Per ulteriori informazioni sulla denominazione dei bucket, consulta Restrizioni e limitazioni dei bucket nella Guida per l'utente di Amazon Simple Storage Service.

Inoltre, verifica di definire una politica per il bucket specificato che concede`PutObject` e`PutObjectAcl` autorizza il servizio QLDB (`qldb.amazonaws.com`). Per ulteriori informazioni, consulta Autorizzazioni di esportazione.

IllegalArgumentException

Messaggio: risposta inaspettata da parte di Amazon S3 durante la convalida della configurazione S3. Risposta da S3: *ErrorCode ErrorMessage*

Il tentativo di scrivere i dati di esportazione del diario nel bucket S3 fornito è fallito con la risposta di errore fornita da Amazon S3. Per ulteriori informazioni sulle possibili cause, consulta Risoluzione dei problemi di Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

IllegalArgumentException

Messaggio: il prefisso del bucket Amazon S3 non deve superare i 128 caratteri

Il prefisso fornito nella richiesta di esportazione del giornale contiene più di 128 caratteri.

IllegalArgumentException

Messaggio: la data di inizio non deve essere successiva alla data di fine

Entrambi`InclusiveStartTime`e`ExclusiveEndTime` deve essere in formato data e ora ISO 8601 e in Coordinated Universal Time (UTC).

IllegalArgumentException

Messaggio: la data di fine non può essere in future

Entrambi`InclusiveStartTime`e`ExclusiveEndTime` deve essere in formato`ISO 8601` data e ora e in UTC.

IllegalArgumentException

Messaggio: l'impostazione di crittografia degli oggetti fornita (S3EncryptionConfiguration) non è compatibile con una chiaveAWS Key Management Service (AWS KMS)

Hai fornito un`KMSKeyArn` con uno`ObjectEncryptionType` dei due`NO_ENCRYPTION` o`SSE_S3`. Puoi fornire a un cliente gestitoAWS KMS key solo un tipo di crittografia a oggetti di`SSE_KMS`. Per ulteriori informazioni sulle opzioni di crittografia lato server in Amazon S3, consulta Protezione dei dati con la crittografia lato server nella Guida per gli sviluppatori di Amazon S3.

LimitExceededException

Messaggio: è stato superato il limite di 2 processi di esportazione del diario in esecuzione simultanea

QLDB impone un limite predefinito di due lavori simultanei di esportazione di diari.

Streaming dei dati del diario

Questa sezione elenca le eccezioni comuni che QLDB può restituire quando si trasmettono i dati del diario da un libro mastro ad Amazon Kinesis Data Streams. Per ulteriori informazioni sull'utilizzo di questa caratteristica, consulta Streaming dei dati del diario da Amazon QLDB.

Ogni eccezione include il messaggio di errore specifico, seguito da una breve descrizione e suggerimenti per possibili soluzioni.

AccessDeniedException

Messaggio: Utente: *UserArn* non è autorizzato a eseguire: iam:PassRole nessuna risorsa: *roleARN*

non disponi delle autorizzazioni a trasferire un ruolo IAM al servizio QLDB. QLDB richiede un ruolo per tutte le richieste di stream di journal ed è necessario disporre delle autorizzazioni per passare questo ruolo a QLDB. Il ruolo fornisce a QLDB le autorizzazioni di scrittura nella risorsa Amazon Kinesis Data Streams specificata.

Verifica di definire una politica IAM che conceda l'autorizzazione a eseguire l'operazione`PassRole` API sulla risorsa di ruolo IAM specificata per il servizio QLDB (`qldb.amazonaws.com`). Per un esempio di policy, consulta Esempi di policy basate sull'identità per Amazon QLDB.

IllegalArgumentException

Messaggio: QLDB ha riscontrato un errore durante la convalida di Kinesis Data Streams: Risposta da Kinesis: *ErrorCode ErrorMessage*

Una possibile causa di questo errore è che la risorsa Kinesis Data Streams fornita non esiste. In alternativa, QLDB non dispone di autorizzazioni sufficienti per scrivere record di dati nel flusso di dati Kinesis specificato.

Verifica che il flusso di dati Kinesis fornito nella richiesta di streaming sia corretto. Per ulteriori informazioni, consulta Creazione e aggiornamento dei flussi di dati nella Guida per gli sviluppatori di Amazon Data Streams.

Inoltre, verifica di definire una politica per il flusso di dati Kinesis specificato che conceda al servizio QLDB (`qldb.amazonaws.com`) le autorizzazioni per le seguenti azioni. Per ulteriori informazioni, consulta Autorizzazioni di streaming.

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

Messaggio: risposta inaspettata da parte di Kinesis Data Streams durante la convalida della configurazione Kinesis. Risposta di Kinesis: *`ErrorCode ErrorMessage`*

Il tentativo di scrivere record di dati nel flusso di dati Kinesis fornito è fallito con la risposta di errore Kinesis fornita. Per ulteriori informazioni sulle possibili cause, consulta Risoluzione dei problemi dei produttori di Amazon Amazon Kinesis Data Streams nella Guida per gli sviluppatori di Amazon Data Streams.

IllegalArgumentException

Messaggio: la data di inizio non deve essere successiva alla data di fine.

Entrambi`InclusiveStartTime``ExclusiveEndTime` deve essere in formato data e ora ISO 8601 e in Coordinated Universal Time (UTC).

IllegalArgumentException

Messaggio: la data di inizio non può essere in future a.

Entrambi`InclusiveStartTime``ExclusiveEndTime` deve essere in formato`ISO 8601` data e ora e in UTC.

LimitExceededException

Messaggio: è stato superato il limite di 5 flussi di Journal in esecuzione simultanea su Kinesis Data Streams

QLDB impone un limite predefinito di cinque flussi di diari simultanei.

Verifica dei dati del diario

Questa sezione elenca le eccezioni comuni che QLDB può restituire quando si verificano i dati contabili in un libro contabile. Per ulteriori informazioni sull'utilizzo di questa caratteristica, consulta Verifica dei dati in Amazon QLDB.

Ogni eccezione include il messaggio di errore specifico, seguito dalle operazioni API che possono generarlo, una breve descrizione e suggerimenti per possibili soluzioni.

IllegalArgumentException

Messaggio: il valore Ion fornito non è valido e non può essere analizzato.

Operazioni API:`GetDigest, GetBlock, GetRevision`

Assicurati di fornire un valore Amazon Ion valido prima di riprovare la richiesta.

IllegalArgumentException

Messaggio: l'indirizzo di blocco fornito non è valido.

Operazioni API:`GetDigest, GetBlock, GetRevision`

Assicurati di fornire un indirizzo di blocco valido prima di riprovare la richiesta. Un indirizzo di blocco è una struttura Amazon Ion con due campi:`strandId` e`sequenceNo`.

Ad esempio: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`

IllegalArgumentException

Messaggio: il numero di sequenza dell'indirizzo del digest fornito supera l'ultimo record registrato nel filone.

Operazioni API:`GetDigest, GetBlock, GetRevision`

L'indirizzo del riepilogo fornito deve avere un numero progressivo inferiore o uguale al numero progressivo dell'ultimo record registrato del filone di giornale. Prima di riprovare la richiesta, assicurati di fornire un indirizzo digest tip con un numero di sequenza valido.

IllegalArgumentException

Messaggio: L'ID Strand dell'indirizzo di blocco fornito non è valido.

Operazioni API:`GetDigest, GetBlock, GetRevision`

L'indirizzo del blocco fornito deve avere un ID del filamento che corrisponda all'ID del filamento del diario. Prima di riprovare la richiesta, assicurati di fornire un indirizzo di blocco con un ID di filamento valido.

IllegalArgumentException

Messaggio: il numero di sequenza dell'indirizzo di blocco fornito supera l'ultimo record di commit del filamento.

Operazioni API:`GetBlock, GetRevision`

L'indirizzo di blocco fornito deve avere un numero di sequenza minore o uguale al numero di sequenza dell'ultimo record di commit del filamento. Prima di riprovare la richiesta, assicurati di fornire un indirizzo di blocco con un numero di sequenza valido.

IllegalArgumentException

Messaggio: l'ID Strand dell'indirizzo di blocco fornito deve corrispondere all'ID Strand dell'indirizzo di suggerimento fornito dal digest.

Operazioni API:`GetBlock, GetRevision`

Puoi verificare la revisione o il blocco di un documento solo se esiste nella stessa sezione del diario del riassunto fornito.

IllegalArgumentException

Messaggio: il numero di sequenza dell'indirizzo di blocco fornito non deve essere maggiore del numero di sequenza dell'indirizzo digest tip fornito.

Operazioni API:`GetBlock, GetRevision`

Puoi verificare la revisione o il blocco di un documento solo se è coperto dal digest fornito. Ciò significa che è stato inserito nel diario prima dell'indirizzo del digest tip.

IllegalArgumentException

Messaggio: l'ID del documento fornito non è stato trovato nel blocco all'indirizzo di blocco specificato.

Funzionamento dell'API:`GetRevision`

L'ID del documento fornito deve essere presente nell'indirizzo di blocco fornito. Prima di riprovare la richiesta, assicurati che questi due parametri siano coerenti.

Documentazione di riferimento Amazon QLDB PartiQL

Amazon QLDB supporta un sottoinsieme del linguaggio di interrogazione PartiQL. I seguenti argomenti descrivono l'implementazione QLDB di PartiQL.

> **Note**
>
> - QLDB non supporta tutte le operazioni PartiQL.
> - Tutte le istruzioni PartiQL in QLDB sono soggette a limiti di transazione, come definito inQuote e limiti in Amazon QLDB.
> - Questo riferimento fornisce esempi di sintassi di base e di utilizzo delle istruzioni PartiQL eseguite manualmente nella console QLDB o nella shell QLDB. Per esempi di codice che mostrano come eseguire istruzioni simili a livello di programmazione utilizzando il driver QLDB, consulta i tutorial inNozioni base sul driver.

Argomenti

Che cos'è PartiQL?

PartiQL consente l'accesso alle query compatibile con SQL su più archivi dati contenenti dati strutturati, semistrutturati e nidificati. È ampiamente utilizzato in Amazon ed è ora disponibile come parte di moltiServizi AWS, tra cui QLDB.

Per la specifica PartiQL e un tutorial sul linguaggio delle query di base, consulta la Documentazione di PartiQL.

PartiQL estende SQL-92 per supportare i documenti nel formato dati Amazon Ion. Per informazioni su Amazon Ion, consultaRiferimento al formato dei dati Amazon Ion in Amazon QLDB.

PartiQL in Amazon QLDB

Per eseguire query PartiQL in QLDB, è possibile utilizzare uno dei seguenti:

- L'editor PartiQL suAWS Management Console per QLDB
- La shell QLDB da riga di comando
- Un driver QLDBAWS fornito per eseguire interrogazioni a livello di codice

Per informazioni sull'utilizzo di questi metodi per accedere a QLDB, vedereAccesso ad Amazon QLDB.

Per informazioni su come controllare l'accesso per eseguire ogni comando PartiQL su tabelle specifiche, vedereGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

Suggerimenti rapidi PartiQL in QLDB

Di seguito è riportato un breve riepilogo dei suggerimenti e delle migliori pratiche per lavorare con PartiQL in QLDB:

- Comprendi i limiti di concorrenza e transazione: tutti i rendiconti, comprese le`SELECT` interrogazioni, sono soggetti a conflitti e limiti di transazione (OCC) ottimistici, incluso un timeout delle transazioni di 30 secondi.
- Usa gli indici: utilizza indici ad alta cardinalità ed esegui interrogazioni mirate per ottimizzare i rendiconti ed evitare scansioni complete delle tabelle. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.

- Usa i predicati di uguaglianza: le ricerche indicizzate richiedono un operatore di uguaglianza (`=` o `IN`). Gli operatori di disuguaglianza (`<>`, `LIKE`, , `BETWEEN`) non sono idonei per le ricerche indicizzate e generano scansioni complete delle tabelle.
- Usa solo join interni: QLDB supporta solo i join interni. Come buona prassi, unisciti ai campi indicizzati per ogni tabella a cui ti stai unendo. Scegli indici ad alta cardinalità sia per i criteri di unione che per i predicati di uguaglianza.

Convenzioni di riferimento Amazon QLDB PartiQL

Questa sezione spiega le convenzioni utilizzate per scrivere la sintassi per i comandi, le funzioni e le espressioni PartiQL descritti nel riferimento PartiQL di Amazon QLDB. Queste convenzioni non devono essere confuse con la sintassi e la semantica del linguaggio di interrogazione PartiQL stesso.

Carattere	Descrizione
CAPS	Le parole in lettere maiuscole sono parole chiave.
[ ]	Le parentesi indicano argomenti o clausole opzionali. Più argomenti tra parentesi indicano che è possibile scegliere qualsiasi numero degli argomenti . Inoltre, gli argomenti tra parentesi su righe separate indicano che il parser QLDB prevede che gli argomenti siano nell'ordine in cui sono elencati nella sintassi.
|	Le pipe indicano che è possibile scegliere tra gli argomenti.
corsivo rosso	Le parole in corsivo rosso indicano dei segnaposto. Inserisci il valore appropriato al posto della parola in corsivo.
. . .	I puntini di sospensione indicano che è possibile ripetere l'elemento precedente.
'	I valori tra virgolette singole indicano che è necessario inserire le virgolette singole. In PartiQL, le virgolette singole indicano valori di stringa o nomi di campo nelle strutture Amazon Ion.
"	I valori tra virgolette doppie indicano che è necessario inserire le virgolette doppie. In PartiQL, le virgolette doppie indicano gli identificatori tra virgolette.

Carattere	Descrizione
`	I valori nei backtick indicano che è necessario inserire i backtick. In PartiQL, i contrassegni indicano i valori letterali di Ion.

Tipi di dati in Amazon QLDB

Amazon QLDB archivia i documenti in formato Amazon Ion. Amazon Ion è un formato di serializzazione dei dati (sia in formato testo che in formato binario) che è un superset di JSON. Nella tabella seguente sono elencati i tipi di dati Ion che è possibile usare nei documenti QLDB.

Tipo di dati	Descrizione
`null`	Un valore nullo generico
`bool`	Valori booleani
`int`	Interi firmati di dimensioni arbitrarie
`decimal`	Numeri reali con codifica decimale di precisione arbitraria
`float`	Numeri in virgola mobile con codifica binaria (IEEE a 64 bit)
`timestamp`	Momenti di data/ora/fuso orario di precisione arbitraria
`string`	Letteralità di testo Unicode
`symbol`	Atomi simbolici Unicode (identificatori)
`blob`	Dati binari di codifica definita dall'utente
`clob`	Dati di testo della codifica definita dall'utente
`struct`	Raccolte non ordinate di coppie nome-valore
`list`	Raccolte di valori eterogenee ordinate

Consulta il documento sulle specifiche Ion sul GitHub sito Amazon per un elenco completo dei tipi di dati Ion core con descrizioni complete e dettagli di formattazione dei valori.

Documentazione Amazon QLDB

Amazon QLDB archivia i record di dati come documenti, che sono solo`struct` oggetti Amazon Ion inseriti in una tabella. Per le specifiche degli ioni, consulta il GitHub sito Amazon Ion.

Argomenti

- Struttura del documento Ion
- Mappatura del tipo a ioni parziali
- ID del documento

Struttura del documento Ion

Come JSON, i documenti QLDB sono composti da coppie nome-valore nella seguente struttura.

```
{
  name1: value1,
  name2: value2,
  name3: value3,
  ...
  nameN: valueN
}
```

I nomi sono simboli e i valori sono illimitati. Ogni coppia nome-valore è chiamata campo. Il valore di un campo può essere uno qualsiasi degli IonTipi di dati, inclusi i tipi di contenitore: strutture annidate, elenchi ed elenchi di strutture.

Inoltre, come JSON, `astruct` è indicato da parentesi ricce (`{...}`) e `alist` è indicato da parentesi quadre (`[...]`). L'esempio seguente è un documento tratto dai dati di esempio inNozioni di base sulla console Amazon QLDB cui sono contenuti valori di vario tipo.

```
{
    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**
>
> In Ion, le virgolette doppie indicano i valori delle stringhe e i simboli senza virgolette rappresentano i nomi dei campi. Ma in PartiQL, le virgolette singole indicano sia le stringhe che i nomi dei campi.
> Questa differenza di sintassi consente al linguaggio di query PartiQL di mantenere la compatibilità SQL e al formato dati Amazon Ion di mantenere la compatibilità JSON. Per dettagli sulla sintassi e sulla semantica di PartiQL in QLDB, vedereInterrogare Ion con PartiQL.

Mappatura del tipo a ioni parziali

In QLDB, PartiQL estende il sistema di tipi SQL per coprire il modello di dati Ion. La mappatura è descritta come segue:

- I tipi scalari SQL sono coperti dalle loro controparti Ion. Ad esempio:
  - `CHAR`e`VARCHAR` sono sequenze Unicode che corrispondono al`string` tipo Ion.
  - `NUMBER`corrisponde al`decimal` tipo Ion.
- Il`struct` tipo di Ion è equivalente a una tupla SQL, che tradizionalmente rappresenta una riga della tabella.
  - Tuttavia, con contenuto aperto e senza schema, le query che si basano sulla natura ordinata di una tupla SQL non sono supportate (come l'ordine di output di`SELECT *`).
- Inoltre`NULL`, PartiQL ha un`MISSING` tipo. Questa è una specializzazione`NULL` e indica la mancanza di un campo. Questo tipo è necessario perché`struct` i campi ionici potrebbero essere sparsi.

ID del documento

QLDB assegna un ID documento a ogni documento inserito in una tabella. Tutti gli ID assegnati dal sistema sono identificatori univoci universali (UUID), ciascuno rappresentato in una stringa codificata in Base62 (ad esempio`3Qv67yjXEwB9SjmvkuG6Cp`). Per ulteriori informazioni, consulta ID univoci in Amazon QLDB.

Ogni revisione del documento è identificata in modo univoco da una combinazione dell'ID del documento e di un numero di versione a base zero.

I campi dell'ID e della versione del documento sono inclusi nei metadati del documento, che è possibile interrogare nella vista commit (la vista definita dal sistema di una tabella). Per ulteriori informazioni sulle visualizzazioni in QLDB, consultareConcetti principali. Per ulteriori informazioni sui metadati, consultaInterrogazione dei metadati dei documenti.

Interrogazione di Ion con PartiQL in Amazon QLDB

Quando esegui una query sui dati in Amazon QLDB, scrivi le istruzioni in formato PartiQL, ma QLDB restituisce i risultati in formato Amazon Ion. PartiQL è pensato per essere compatibile con SQL, mentre Ion è un'estensione di JSON. Ciò comporta differenze sintattiche tra il modo in cui si annotano i dati nelle istruzioni di interrogazione e il modo in cui vengono visualizzati i risultati della query.

Questa sezione descrive la sintassi e la semantica di base per eseguire manualmente le istruzioni PartiQL utilizzando la console QLDB o la shell QLDB.

> **Tip**
>
> Quando si eseguono interrogazioni PartiQL in modo programmatico, la procedura migliore consiste nell'utilizzare istruzioni parametrizzate. È possibile utilizzare un punto interrogativo (?) come segnaposto della variabile di associazione nelle dichiarazioni per evitare queste regole di sintassi. Questo è anche più sicuro ed efficiente.
> Per ulteriori informazioni, consulta i seguenti tutorial in Guida introduttiva al driver:
>
> - Giava:Guida di avvio rapido |Documentazione di riferimento del libro
> - .NET:Guida introduttiva rapida al tutorial |Documentazione di riferimento del libro
> - Vai:Fase 4 di Quick Start |Riferimento del libro di cucina
> - Node.js:Guida di avvio rapido |Riferimento del libro di cucina

- Pitone:Guida di avvio rapido |Riferimento del libro di cucina

Argomenti

Sintassi e semantica

Quando si utilizza la console QLDB o la shell QLDB per interrogare i dati Ion, le seguenti sono la sintassi e la semantica fondamentali di PartiQL:

Distinzione tra lettere maiuscole e minuscole

Tutti i nomi degli oggetti del sistema QLDB, inclusi i nomi dei campi, i nomi delle tabelle e i nomi dei registri, fanno distinzione tra maiuscole e minuscole.

Valori di stringa

In Ion, le virgolette doppie (`"..."`) indicano una stringa.

In PartiQL, le virgolette singole (`'...'`) indicano una stringa.

Simboli e identificatori

In Ion, le virgolette singole (`'...'`) indicano un simbolo. Un sottoinsieme di simboli in Ion chiamati identificatori è rappresentato da testo senza virgolette.

In PartiQL, le virgolette doppie (`"..."`) indicano un identificatore PartiQL tra virgolette, ad esempio una parola riservata utilizzata come nome di tabella. Il testo senza virgolette rappresenta un normale identificatore PartiQL, ad esempio un nome di tabella che non è una parola riservata.

Letterali Ion

Qualsiasi valore letterale Ion può essere indicato con backticks (`` `...` ``) in un'istruzione PartiQL.

Nomi di campo

I nomi dei campi Ion fanno distinzione tra maiuscole e minuscole. PartiQL consente di indicare i nomi dei campi con virgolette singole in un'istruzione DML. Questa è un'alternativa abbreviata all'utilizzo della`cast` funzione di PartiQL per definire un simbolo. È anche più intuitivo rispetto all'uso delle barrette inverse per indicare un simbolo ionico letterale.

Valori letterali

I valori letterali del linguaggio di interrogazione PartiQL corrispondono ai tipi di dati Ion, come segue:

Scalari

Segui la sintassi SQL quando applicabile, come descritto nellaMappatura del tipo a ioni parziali sezione. Ad esempio:

- `5`
- `'foo'`
- `null`

Strutture

Conosciuti anche come tuple o oggetti in molti formati e altri modelli di dati.

Denotato da parentesi ricce (`{...}`) con`construct` elementi separati da virgole.

- `{ 'id' : 3, 'arr': [1, 2] }`

Elenchi

Conosciuti anche come array.

Indicato da parentesi quadre (`[...]`) con elementi dell'elenco separati da virgole.

- `[ 1, 'foo' ]`

Borse

Raccolte non ordinate in PartiQL.

Indicato da parentesi a doppio angolo (`<<...>>`) con elementi del sacchetto separati da virgole. In QLDB, un tavolo può essere paragonato a una borsa. Tuttavia, una borsa non può essere annidata all'interno dei documenti di una tabella.

- `<< 1, 'foo' >>`

Esempio

Di seguito viene mostrato un esempio della sintassi di un'`INSERT`istruzione con vari tipi di ioni.

```
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`
}
```

Notazione con contrassegno

PartiQL copre completamente tutti i tipi di dati Ion, in modo da poter scrivere qualsiasi dichiarazione senza utilizzare contrassegni. Ma ci sono casi in cui questa sintassi letterale di Ion può rendere le tue affermazioni più chiare e concise.

Ad esempio, per inserire un documento con timestamp e valori simbolici di Ion, è possibile scrivere la seguente dichiarazione utilizzando solo una sintassi puramente PartiQL.

```
INSERT INTO myTable VALUE
{
    'myTimestamp': to_timestamp('2019-09-04T'),
    'mySymbol': cast('foo' as symbol)
}
```

Questo è abbastanza dettagliato, quindi puoi invece usare i contrassegni per semplificare la tua dichiarazione.

```
INSERT INTO myTable VALUE
{
    'myTimestamp': `2019-09-04T`,
    'mySymbol': `foo`
}
```

Puoi anche racchiudere l'intera struttura in barre inverse per risparmiare qualche altra sequenza di tasti.

```
INSERT INTO myTable VALUE
`{
    myTimestamp: 2019-09-04T,
    mySymbol: foo
}`
```

Important

Le stringhe e i simboli sono classi diverse in PartiQL. Ciò significa che anche se hanno lo stesso testo, non sono uguali. Ad esempio, le seguenti espressioni PartiQL restituiscono valori Ion diversi.

```
'foo'
```

```
`foo`
```

Navigazione del percorso

Quando si scrivono istruzioni DML (Data Manipulation Language) o interrogazioni, è possibile accedere ai campi all'interno di strutture annidate utilizzando i passaggi del percorso. PartiQL supporta la notazione a punti per accedere ai nomi dei campi di una struttura principale. L'esempio seguente accede al`Model` campo di un genitore`Vehicle`.

```
Vehicle.Model
```

Per accedere a un elemento specifico di un elenco, è possibile utilizzare l'operatore tra parentesi quadre per indicare un numero ordinale a base zero. L'esempio seguente accede all'elemento

di`SecondaryOwners` con un numero ordinale di2. In altre parole, questo è il terzo elemento dell'elenco.

```
SecondaryOwners[2]
```

Alias

QLDB supporta contenuti e schemi aperti. Quindi, quando accedi a determinati campi di un'istruzione, il modo migliore per assicurarti di ottenere i risultati che ti aspetti è usare degli alias. Ad esempio, se non specifichi un alias esplicito, il sistema ne genera uno implicito per le tue`FROM` fonti.

```
SELECT VIN FROM Vehicle
--is rewritten to
SELECT Vehicle.VIN FROM Vehicle AS Vehicle
```

Tuttavia, i risultati sono imprevedibili in caso di conflitti di nomi di campo. Se`VIN` esiste un altro campo denominato in una struttura annidata all'interno dei documenti, i`VIN` valori restituiti da questa interrogazione potrebbero sorprenderti. Come buona pratica, scrivi invece la seguente dichiarazione. Questa query vienev dichiarata come alias che si estende sulla`Vehicle` tabella. La`AS` parola chiave è facoltativa.

```
SELECT v.VIN FROM Vehicle [ AS ] v
```

L'aliasing è particolarmente utile quando si inserisce il percorso in raccolte annidate all'interno di un documento. Ad esempio, la seguente dichiarazione dichiarao come alias che si estende all'interno della raccolta`VehicleRegistration.Owners`.

```
SELECT o.SecondaryOwners
FROM VehicleRegistration AS r, @r.Owners AS o
```

Il@ personaggio è tecnicamente opzionale qui. Ma indica esplicitamente che si desidera la`Owners` struttura all'interno`VehicleRegistration`, non una raccolta con un nome diverso`Owners` (se ne esiste una).

Specificazione PartiQL

Per ulteriori informazioni sul linguaggio di query PartiQL, vedere la specifica PartiQL.

Comandi PartiQL in Amazon QLDB

PartiQL estende SQL-92 per supportare i documenti nel formato dati Amazon Ion. Amazon QLDB supporta i seguenti comandi PartiQL.

Per informazioni su come controllare l'accesso per eseguire ogni comando PartiQL su tabelle specifiche, vedereGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

> **Note**
>
> - QLDB non supporta tutti i comandi PartiQL.
> - Tutte le istruzioni PartiQL in QLDB sono soggette a limiti di transazione, come definito inQuote e limiti in Amazon QLDB.
> - Questo riferimento fornisce esempi di sintassi di base e di utilizzo delle istruzioni PartiQL eseguite manualmente nella console QLDB o nella shell QLDB. Per esempi di codice che mostrano come eseguire istruzioni simili utilizzando un linguaggio di programmazione supportato, consulta i tutorial inNozioni base sul driver.

Dichiarazioni DDL (linguaggio di definizione dei dati)

Il linguaggio di definizione dei dati (DDL, Data Language), l'insieme di istruzioni PartiQL utilizzate per gestire gli oggetti del database, ad esempio tabelle e indici. Si utilizza DDL per creare e rilasciare questi oggetti.

- CREATE INDEX
- CREATE TABLE
- DROP INDEX
- DROP TABLE
- TAVOLO UNDROP

Istruzioni DML (linguaggio di manipolazione dei dati)

Il linguaggio di manipolazione dei dati (DML, Data Manulation Language), l'insieme di istruzioni PartiQL utilizzate per gestire i dati nelle tabelle QLDB. Le istruzioni DML vengono utilizzate per aggiungere, modificare o eliminare i dati in una tabella.

Sono supportate le seguenti istruzioni DML e del linguaggio di query:

- DELETE
- DA (INSERISCI, RIMUOVI o IMPOSTA)
- INSERT
- SELECT
- UPDATE

Comando CREATE INDEX in Amazon QLDB

In Amazon QLDB, usa il`CREATE INDEX` comando per creare un indice per un campo del documento su una tabella.

Per informazioni su come controllare l'accesso per eseguire questo comando PartiQL su tabelle specifiche, vedereGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

> **Important**
>
> QLDB richiede un indice per cercare in modo efficiente un documento. Senza un indice, QLDB deve eseguire una scansione completa della tabella durante la lettura dei documenti. Ciò può causare problemi di prestazioni su tabelle di grandi dimensioni, inclusi conflitti di concorrenza e timeout delle transazioni.
> Per evitare la scansione delle tabelle, è necessario eseguire istruzioni con una clausola`WHERE` predicativa utilizzando un operatore di uguaglianza (`=`o`IN`) su un campo indicizzato o un ID di documento. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.

Nota i seguenti vincoli durante la creazione degli indici:

- Un indice può essere creato solo su un singolo campo di primo livello. Gli indici compositi, annidati, unici e basati su funzioni non sono supportati.
- È possibile creare un indice su qualsiasi tipo di dati Ion, inclusi`list` `struct`. Tuttavia, è possibile eseguire la ricerca indicizzata solo in base all'uguaglianza dell'intero valore Ion indipendentemente dal tipo di Ion. Ad esempio, quando si utilizza un`list` tipo come indice, non è possibile eseguire una ricerca indicizzata per un elemento all'interno dell'elenco.

- Le prestazioni delle query migliorano solo quando si utilizza un predicato di uguaglianza, ad esempio`WHERE indexedField = 123` o`WHERE indexedField IN (456, 789)`.

  QLDB non rispetta le disuguaglianze nei predicati delle query. Di conseguenza, le scansioni filtrate per intervallo non vengono implementate.
- I nomi dei campi indicizzati fanno distinzione tra maiuscole e minuscole e minuscole e possono contenere fino a 128 caratteri.
- La creazione di indici in QLDB è asincrona. La quantità di tempo necessaria per completare la creazione di un indice su una tabella non vuota varia a seconda delle dimensioni della tabella. Per ulteriori informazioni, consulta Gestione degli indici.

Argomenti

Sintassi

```
CREATE INDEX ON table_name (field)
```

Parametri

table_name

Il nome della tabella in cui si desidera creare l'indice. La tabella deve già essere presente.

Per il nome del file è prevista la distinzione tra maiuscole e minuscole.

campo

Il nome del campo del documento per cui creare l'indice. Il campo deve essere un attributo di primo livello.

I nomi dei campi indicizzati fanno distinzione tra maiuscole e minuscole e minuscole e possono contenere fino a 128 caratteri.

Puoi creare un indice su qualsiasi tipo di dati Amazon Ion, inclusi`list` `estruct`. Tuttavia, è possibile eseguire la ricerca indicizzata solo in base all'uguaglianza dell'intero valore Ion indipendentemente dal tipo di Ion. Ad esempio, quando si utilizza un`list` tipo come indice, non è possibile eseguire una ricerca indicizzata per un elemento all'interno dell'elenco.

Valore restituito

`tableId`— L'ID univoco della tabella su cui hai creato l'indice.

Esempi

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

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

Esecuzione a livello di codice utilizzando il driver

Per imparare a eseguire questa istruzione a livello di codice utilizzando il driver QLDB, consulta i seguenti tutorial in Guida introduttiva al driver:

- Giava:Guida di avvio rapido |Documentazione di riferimento del libro
- .NET:Guida introduttiva rapida al tutorial |Documentazione di riferimento del libro
- Vai:Fase 4 di Quick Start |Riferimento del libro di cucina
- Node.js:Guida di avvio rapido |Riferimento del libro di cucina
- Pitone:Guida di avvio rapido |Riferimento del libro di cucina

Comando CREATE TABLE in Amazon QLDB

In Amazon QLDB, usa il`CREATE TABLE` comando per creare una nuova tabella.

Le tabelle hanno nomi semplici senza namespace. QLDB supporta contenuti aperti e non applica schemi, quindi non si definiscono attributi o tipi di dati durante la creazione di tabelle.

> **Note**
>
> Per informazioni su come controllare l'accesso per eseguire questo comando PartiQL in un libro mastro, vedereGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

Argomenti

Sintassi

```
CREATE TABLE table_name [ WITH (aws_tags = `{'key': 'value'}`) ]
```

Parametri

table_name

Il nome univoco della tabella da creare. Una tabella attiva con lo stesso nome non deve già esistere. Di seguito sono riportati i vincoli di denominazione:

- Deve contenere solo da 1 a 128 caratteri alfanumerici o caratteri di sottolineatura.
- Devi contenere una lettera o un trattino basso per il primo carattere.
- Può avere qualsiasi combinazione di caratteri alfanumerici e caratteri di sottolineatura per i caratteri rimanenti.
- distingue tra maiuscole e minuscole
- Non deve essere una parola riservata QLDB PartiQL.

'***chiave***': '***valore***'

(Facoltativo) I tag da applicare alla risorsa della tabella durante la creazione. Ogni tag è definito come una coppia chiave-valore, in cui la chiave e il valore sono indicati ciascuno da virgolette

singole. Ogni coppia chiave-valore è definita all'interno di una struttura Amazon Ion indicata da contrassegni.

L'etichettatura delle tabelle al momento della creazione è attualmente supportata per i libri contabili solo in modalità*STANDARD* autorizzazioni.

Valore restituito

`tableId`— L'ID univoco della tabella che hai creato.

Tag di tabelle durante la creazione

> **Note**
>
> L'etichettatura delle tabelle al momento della creazione è attualmente supportata per i libri contabili solo in modalità`STANDARD` autorizzazioni.

Facoltativamente, puoi etichettare le risorse della tabella specificando i tag in una`CREATE TABLE` dichiarazione. Per ulteriori informazioni sui tag, consulta Assegnazione di tag alle risorse Amazon QLDB. L'esempio seguente crea una tabella denominata`Vehicle` con il tag`environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

L'assegnazione di tag alle tabelle al momento della creazione richiede l'accesso sia alle azioni che`qldb:TagResource` alle azioni.`qldb:PartiQLCreateTable` Per ulteriori informazioni sui permessi per le risorse QLDB, vedereCome funziona Amazon QLDB con IAM.

L'aggiunta di tag alle risorse in fase di creazione consente di evitare di eseguire script di tagging personalizzati dopo la creazione delle risorse. Dopo aver applicato un tag a una tabella, puoi controllare l'accesso alla tabella in base a tali tag. Ad esempio, puoi concedere l'accesso completo solo alle tabelle con un tag specifico. Per un esempio di policy JSON, vedereAccesso completo a tutte le azioni basate sui tag della tabella.

Esempi

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'development'}`)
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'key1': 'value1', 'key2': 'value2'}`)
```

Esecuzione a livello di codice utilizzando il driver

Per imparare a eseguire questa istruzione a livello di codice utilizzando il driver QLDB, consulta i seguenti tutorial in Guida introduttiva al driver:

- Giava:Guida di avvio rapido |Documentazione di riferimento del libro
- .NET:Guida introduttiva rapida al tutorial |Documentazione di riferimento del libro
- Vai:Fase 4 di Quick Start |Riferimento del libro di cucina
- Node.js:Guida di avvio rapido |Riferimento del libro di cucina
- Pitone:Guida di avvio rapido |Riferimento del libro di cucina

Comando DELETE in Amazon QLDB

In Amazon QLDB, utilizza il`DELETE` comando per contrassegnare un documento attivo come eliminato in una tabella creando una nuova ma definitiva revisione del documento. Questa revisione finale indica che il documento è stato eliminato. Questa operazione termina il ciclo di vita di un documento, il che significa che non è possibile creare ulteriori revisioni del documento con lo stesso ID del documento.

Questa operazione è irreversibile. È comunque possibile interrogare la cronologia delle revisioni di un documento eliminato utilizzando ilFunzione di cronologia.

> **Note**
>
> Per informazioni su come controllare l'accesso per eseguire questo comando PartiQL su tabelle specifiche, vedereGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

Argomenti

- Sintassi

Sintassi

```
DELETE FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
```

Parametri

table_name

Il nome della tabella utente contenente i dati da eliminare. Le istruzioni DML sono supportate solo nella visualizzazione utente predefinita. Ogni istruzione può essere eseguita solo su una singola tabella.

Tabella_alias AS

(Facoltativo) Un alias definito dall'utente che si estende su una tabella da cui eliminare. La`AS` parola chiave è facoltativa.

DI ***id_alias***

(Facoltativo) Un alias definito dall'utente che si lega al campo dei`id` metadati di ogni documento nel set di risultati. L'alias deve essere dichiarato nella`FROM` clausola utilizzando la`BY` parola chiave. Ciò è utile quando si desidera filtrare in base all'ID del documento mentre si esegue una query sulla visualizzazione utente predefinita. Per ulteriori informazioni, consulta Utilizzo della clausola BY per interrogare l'ID del documento.

WHERE ***condition***

I criteri di selezione per i documenti da eliminare.

> **Note**
>
> Se si`WHERE` omette la clausola, saranno eliminati tutti i documenti della tabella.

Valore restituito

`documentId`— L'ID univoco di ogni documento eliminato.

Esempi

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

Esecuzione a livello di codice utilizzando il driver

Per imparare a eseguire questa istruzione a livello di codice utilizzando il driver QLDB, consulta i seguenti tutorial in Guida introduttiva al driver:

- Giava:Guida di avvio rapido |Documentazione di riferimento del libro
- .NET:Guida introduttiva rapida al tutorial |Documentazione di riferimento del libro
- Vai:Fase 4 di Quick Start |Riferimento del libro di cucina
- Node.js:Guida di avvio rapido |Riferimento del libro di cucina
- Pitone:Guida di avvio rapido |Riferimento del libro di cucina

Comando DROP INDEX in Amazon QLDB

In Amazon QLDB, usa il`DROP INDEX` comando per eliminare un indice da una tabella.

> **Note**
>
> Per informazioni su come controllare l'accesso per eseguire questo comando PartiQL su tabelle specifiche, vedereGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

Argomenti

Sintassi

```
DROP INDEX "indexId" ON table_name WITH (purge = true)
```

> **Note**
>
> La clausola`WITH (purge = true)` è obbligatoria per tutte le`DROP INDEX` istruzioni ed`true` è attualmente l'unico valore supportato.
> La parola chiave`purge` fa distinzione tra maiuscole e minuscole e minuscole e minuscole.

Parametri

«*ID indice*»

L'ID univoco dell'indice da eliminare, indicato tra virgolette doppie.

ON *nome_tabella*

Il nome della tabella di cui vuoi eliminare l'indice.

Valore restituito

`tableId`— L'ID univoco della tabella di cui hai eliminato l'indice.

Esempi

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

Comando DROP TABLE in Amazon QLDB

In Amazon QLDB, usa il`DROP TABLE` comando per disattivare una tabella esistente. È possibile utilizzare l'TAVOLO UNDROPistruzione per riattivarla. La disattivazione o la riattivazione di una tabella non ha alcun effetto sui relativi documenti o indici.

> **Note**
>
> Per informazioni su come controllare l'accesso per eseguire questo comando PartiQL su tabelle specifiche, vedereGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

Argomenti

Sintassi

```
DROP TABLE table_name
```

Parametri

`table_name`

Il nome della tabella da disattivare. La tabella deve essere già esistente e con statoACTIVE.

Valore restituito

`tableId`— L'ID univoco della tabella che hai disattivato.

Esempi

```
DROP TABLE VehicleRegistration
```

Comando FROM (INSERT, REMOVE o SET) in Amazon QLDB

In Amazon QLDB, un'istruzione che inizia conFROM è un'estensione PartiQL che consente di inserire e rimuovere elementi specifici all'interno di un documento. È inoltre possibile utilizzare questa istruzione per aggiornare gli elementi esistenti in un documento, in modo simile alUPDATE comando.

> **Note**
>
> Per informazioni su come controllare l'accesso per eseguire questo comando PartiQL su tabelle specifiche, vedereGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

Argomenti

Sintassi

DA INSERIMENTO

Inserisci un nuovo elemento in un documento esistente. Per inserire un nuovo documento di primo livello in una tabella, è necessario utilizzareINSERT.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
INSERT INTO element VALUE data [ AT key_name ]
```

DA-RIMUOVI

Rimuovi un elemento esistente all'interno di un documento o rimuovi un intero documento di primo livello. Quest'ultima è semanticamente uguale allaDELETE sintassi tradizionale.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
REMOVE element
```

DAL SET

Aggiorna uno o più elementi all'interno di un documento. Se un elemento non esiste, viene inserito. Semanticamente è la stessa dellaUPDATE sintassi tradizionale.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
SET element = data [, element = data, ... ]
```

Parametri

table_name

:il nome della tabella utente contenente i dati da modificare. Le istruzioni DML sono supportate solo nella visualizzazione utente predefinita. Ogni istruzione può essere eseguita solo su una singola tabella.

In questa clausola, puoi anche includere una o più raccolte annidate all'interno della tabella specificata. Per ulteriori dettagli, consulta Collezioni nidificate.

Tabella_alias AS

(Facoltativo) Un alias definito dall'utente che si estende su una tabella da modificare. Tutti gli alias di tabella utilizzati nella`WHERE` clausola`SETREMOVEINSERT INTO`,, or devono essere dichiarati nella`FROM` clausola. La`AS` parola chiave è facoltativa.

DI *id_alias*

(Facoltativo) Un alias definito dall'utente che si lega al campo dei`id` metadati di ogni documento nel set di risultati. L'alias deve essere dichiarato nella`FROM` clausola utilizzando la`BY` parola chiave. Ciò è utile quando si desidera filtrare in base all'ID del documento mentre si esegue una query sulla visualizzazione utente predefinita. Per ulteriori informazioni, consulta Utilizzo della clausola BY per interrogare l'ID del documento.

WHERE *condition*

I criteri di selezione per i documenti da modificare.

> **Note**
>
> Se si`WHERE` omette la clausola, saranno modificati tutti i documenti della tabella.

elemento

Un elemento di documento da creare o modificare.

dati

Un nuovo valore per l'elemento.

Nome_chiave AT

Un nome chiave da aggiungere nei documenti da modificare. È necessario specificare il corrispondenteVALUE insieme al nome della chiave. Ciò è necessario per inserire un nuovo valore inAT una posizione specifica all'interno di un documento.

Collezioni nidificate

Sebbene sia possibile eseguire un'istruzione DML solo su una singola tabella, è possibile specificare le raccolte annidate all'interno dei documenti di quella tabella come fonti aggiuntive. Ogni alias dichiarato per una raccolta annidata può essere utilizzato nellaWHERE clausola e nellaREMOVE clausolaSETINSERT INTO, or.

Ad esempio, leFROM fonti della seguente dichiarazione includono sia laVehicleRegistration tabella che laOwners.SecondaryOwners struttura annidata.

```
FROM VehicleRegistration r, @r.Owners.SecondaryOwners o
WHERE r.VIN = '1N4AL11D75C109151' AND o.PersonId = 'abc123'
SET o.PersonId = 'def456'
```

Questo esempio aggiorna l'elemento specifico dell'SecondaryOwnerselenco che ha un o'abc123' all'internoPersonId delVehicleRegistration documento che ha unVIN di'1N4AL11D75C109151'. Questa espressione consente di specificare un elemento di un elenco in base al suo valore anziché al suo indice.

Valore restituito

documentId— L'ID univoco di ogni documento aggiornato o eliminato.

Esempi

Modifica un elemento all'interno di un documento. Se l'elemento non esiste, viene inserito.

```
FROM Vehicle AS v
WHERE v.VIN = '1N4AL11D75C109151' AND v.Color = 'Silver'
SET v.Color = 'Shiny Gray'
```

Modifica o inserisci un elemento e filtra nel campo dei`id` metadati del documento assegnato dal sistema.

```
FROM Vehicle AS v BY v_id
WHERE v_id = 'documentId'
SET v.Color = 'Shiny Gray'
```

Modifica il`PersonId` campo del primo elemento dell'`Owners.SecondaryOwners`elenco all'interno di un documento.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
```

Rimuove un elemento esistente all'interno di un documento.

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

Rimuove un intero documento da una tabella.

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

Rimuove il primo elemento dell'`Owners.SecondaryOwners`elenco all'interno di un documento della`VehicleRegistration` tabella.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
REMOVE r.Owners.SecondaryOwners[0]
```

Inserisci`{'Mileage':26500}` come coppia nome-valore di primo livello all'interno di un documento nella`Vehicle` tabella.

```
FROM Vehicle AS v
WHERE v.VIN = '1N4AL11D75C109151'
INSERT INTO v VALUE 26500 AT 'Mileage'
```

Aggiungi`{'PersonId':'abc123'}` come coppia nome-valore nel`0wners.SecondaryOwners` campo di un documento nella`VehicleRegistration` tabella. Nota che`0wners.SecondaryOwners` deve già esistere e deve essere un tipo di dati di elenco affinché questa dichiarazione sia valida. Altrimenti, la parola chiave`AT` è obbligatoria nella`INSERT INTO` clausola.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
```

Inserisci`{'PersonId':'abc123'}` come primo elemento nell'`0wners.SecondaryOwner`elenco esistente all'interno di un documento.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
```

Aggiungi più coppie nome-valore all'`0wners.SecondaryOwner`elenco esistente all'interno di un documento.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
```

Esecuzione a livello di codice utilizzando il driver

Per imparare a eseguire questa istruzione a livello di codice utilizzando il driver QLDB, consulta i seguenti tutorial in Guida introduttiva al driver:

- Giava:Guida di avvio rapido |Documentazione di riferimento del libro
- .NET:Guida introduttiva rapida al tutorial |Documentazione di riferimento del libro
- Vai:Fase 4 di Quick Start |Riferimento del libro di cucina
- Node.js:Guida di avvio rapido |Riferimento del libro di cucina

- Pitone:Guida di avvio rapido |Riferimento del libro di cucina

Comando INSERT in Amazon QLDB

In Amazon QLDB, usa il`INSERT` comando per aggiungere uno o più documenti Amazon Ion a una tabella.

> **Note**
>
> Per informazioni su come controllare l'accesso per eseguire questo comando PartiQL su tabelle specifiche, vedereGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

Argomenti

- Sintassi
- Parametri
- Valore restituito
- Esempi
- Esecuzione a livello di codice utilizzando il driver

Sintassi

Inserire un singolo documento.

```
INSERT INTO table_name VALUE document
```

Inserisci più documenti.

```
INSERT INTO table_name << document, document, ... >>
```

Parametri

table_name

Il nome della tabella utente in cui si desidera inserire i dati. La tabella deve già essere presente. Le istruzioni DML sono supportate solo nella visualizzazione utente predefinita.

documento

Un documento QLDB valido. Devi specificare almeno un documento. Devi separare più documenti da virgole.

Il documento deve essere contrassegnato da parentesi ricce (`{...}`).

Ogni nome di campo nel documento è un simbolo Ion con distinzione tra maiuscole e minuscole che può essere indicato da virgolette singole (`'...'`) in PartiQL.

Anche i valori stringa sono riportati con virgolette singole (`'...'`) in PartiQL.

Qualsiasi valore letterale Ion può essere indicato con contrassegni (`` `...` ``).

> **Note**
>
> Le parentesi a doppio angolo (`<<...>>`) indicano una raccolta non ordinata (nota come borsa in PartiQL) e sono necessarie solo se si desidera inserire più documenti.

Valore restituito

`documentId`— L'ID univoco di ogni documento inserito.

Esempi

Inserire un singolo documento.

```
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`
}
```

Questa istruzione restituisce l'ID univoco del documento che hai inserito, come segue.

```
{
    documentId: "2kKuOPNB07D2iTPBrUTWGl"
}
```

Inserisci più documenti.

```
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'
}
>>
```

Questa istruzione restituisce l'ID univoco di ogni documento inserito, come segue.

```
{
    documentId: "6WXzLscsJ3bDWW97Dy8nyp"
},
{
    documentId: "35e0ToZyTGJ7LGvcwrkX65"
},
{
    documentId: "BVHPcH612o7JROQ4yP8jiH"
}
```

Esecuzione a livello di codice utilizzando il driver

Per imparare a eseguire questa istruzione a livello di codice utilizzando il driver QLDB, consulta i seguenti tutorial in Guida introduttiva al driver:

- Giava:Guida di avvio rapido |Documentazione di riferimento del libro
- .NET:Guida introduttiva rapida al tutorial |Documentazione di riferimento del libro
- Vai:Fase 4 di Quick Start |Riferimento del libro di cucina
- Node.js:Guida di avvio rapido |Riferimento del libro di cucina
- Pitone:Guida di avvio rapido |Riferimento del libro di cucina

Comando SELECT in Amazon QLDB

In Amazon QLDB, usa il`SELECT` comando per recuperare dati da una o più tabelle. Ogni`SELECT` query in QLDB viene elaborata in una transazione ed è soggetta a un limite di timeout della transazione.

L'ordine dei risultati non è specifico e può variare per ogni`SELECT` interrogazione. Non dovresti fare affidamento sull'ordine dei risultati per nessuna query in QLDB.

Per informazioni su come controllare l'accesso per eseguire questo comando PartiQL su tabelle specifiche, vedereGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

> **Warning**
>
> Quando si esegue una query in QLDB senza una ricerca indicizzata, viene richiamata una scansione completa della tabella. PartiQL supporta tali interrogazioni perché è compatibile

con SQL. Tuttavia, non eseguite scansioni di tabelle per casi d'uso di produzione in QLDB. Le scansioni delle tabelle possono causare problemi di prestazioni su tabelle di grandi dimensioni, inclusi conflitti di concorrenza e timeout delle transazioni.
Per evitare la scansione delle tabelle, è necessario eseguire istruzioni con una clausolaWHERE predicativa utilizzando un operatore di uguaglianza su un campo indicizzato o un ID di documento; ad esempio,`WHERE indexedField = 123` o`WHERE indexedField IN (456, 789)`. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.

Argomenti

- Sintassi
- Parametri
- Join
- Limitazioni delle query nidificate
- Esempi
- Esecuzione a livello di codice utilizzando il driver

Sintassi

```
SELECT [ VALUE ] expression [ AS field_alias ] [, expression, ... ]
FROM source [ AS source_alias ] [ AT idx_alias ] [ BY id_alias ] [, source, ... ]
[ WHERE condition ]
```

Parametri

VALUE

Un qualificatore per l'espressione che fa sì che la query restituisca il valore del tipo di dati non elaborati, anziché il valore racchiuso in una struttura a tupla.

espressione

Una proiezione formata dal carattere* jolly o un elenco di proiezione di uno o più campi del documento dal set di risultati. Un'espressione può essere costituita da chiamate a Funzioni PartiQL o da campi modificati da Operatori QL PartiQL.

Alias di campo AS

(Facoltativo) Un alias temporaneo definito dall'utente per il campo utilizzato nel set di risultati finali. LaAS parola chiave è facoltativa.

Se non si specifica un alias per un'espressione che non è un semplice nome di campo, il set di risultati applica un nome predefinito a quel campo.

DALLA ***fonte***

Una fonte da interrogare. Le uniche fonti attualmente supportate sono i nomi delle tabelle, i join interni tra tabelle, leSELECT interrogazioni annidate (soggette aLimitazioni delle query nidificate) e le chiamate alle funzioni di cronologia per una tabella.

Devi specificare almeno una fonte. Le fonti multiple devono essere separate da virgole.

Alias sorgente AS

(Facoltativo) Un alias definito dall'utente che si estende su una fonte da interrogare. Tutti gli alias di origine utilizzati nellaWHERE clausolaSELECT OR devono essere dichiarati nellaFROM clausola. LaAS parola chiave è facoltativa.

AT ***idx_alias***

(Facoltativo) Un alias definito dall'utente che si lega al numero di indice (ordinale) di ogni elemento all'interno di un elenco dalla fonte. L'alias deve essere dichiarato nellaFROM clausola utilizzando laAT parola chiave.

DI ***id_alias***

(Facoltativo) Un alias definito dall'utente che si lega al campo deiid metadati di ogni documento nel set di risultati. L'alias deve essere dichiarato nellaFROM clausola utilizzando laBY parola chiave. Ciò è utile quando si desidera proiettare o filtrare l'ID del documento mentre si esegue una query sulla visualizzazione utente predefinita. Per ulteriori informazioni, consulta Utilizzo della clausola BY per interrogare l'ID del documento.

WHERE ***condition***

I criteri di selezione e i criteri di partecipazione (se applicabili) per l'interrogazione.

> **Note**
>
> Se siWHERE omette la clausola, saranno recuperati tutti i documenti della tabella.

Join

Sono attualmente supportati solo i join interni. È possibile scrivere interrogazioni interne utilizzando la`INNER JOIN` clausola esplicita, come segue. In questa sintassi,`JOIN` deve essere associato a e`ON` la`INNER` parola chiave è facoltativa.

```
SELECT expression
FROM table1 AS t1 [ INNER ] JOIN table2 AS t2
ON t1.element = t2.element
```

In alternativa, puoi scrivere join interni usando la sintassi implicita, come segue.

```
SELECT expression
FROM table1 AS t1, table2 AS t2
WHERE t1.element = t2.element
```

Limitazioni delle query nidificate

È possibile scrivere interrogazioni annidate (sottoquery) all'interno di`SELECT` espressioni e all'interno delle`FROM` fonti. La restrizione principale è che solo la query più esterna può accedere all'ambiente di database globale. Ad esempio, supponiamo di avere un libro mastro con tabelle`VehicleRegistration` e`Person`. La seguente query annidata non è valida perché l'utente interno`SELECT` tenta di accedere`Person`.

```
SELECT r.VIN,
    (SELECT p.PersonId FROM Person AS p WHERE p.PersonId =
 r.Owners.PrimaryOwner.PersonId) AS PrimaryOwner
FROM VehicleRegistration AS r
```

Considerando che la seguente query annidata è valida.

```
SELECT r.VIN, (SELECT o.PrimaryOwner.PersonId FROM @r.Owners AS o) AS PrimaryOwner
FROM VehicleRegistration AS r
```

Esempi

La seguente query mostra un carattere jolly di base`SELECT` con una clausola`WHERE` predicativa standard che utilizza l'`IN`operatore.

```
SELECT * FROM Vehicle
WHERE VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Quanto segue mostra leSELECT proiezioni con un filtro a stringa.

```
SELECT FirstName, LastName, Address
FROM Person
WHERE Address LIKE '%Seattle%'
AND GovId = 'LEWISR261LL'
```

Quanto segue mostra una subquery correlata che appiattisce i dati annidati. Nota che il@ personaggio è tecnicamente opzionale qui. Ma indica esplicitamente che desideri la`Owners` struttura che è annidata all'interno`VehicleRegistration`, non una raccolta con un nome diverso`Owners` (se ne esiste una). Per maggiori informazioni, consulta ilDati nidificati capitolo Utilizzo dei dati e della cronologia.

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

Quanto segue mostra una subquery nell'SELECTelenco che proietta dati annidati e un join interno implicito.

```
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')
```

Quanto segue mostra un inner join esplicito.

```
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')
```

Quanto segue mostra una proiezione del campo dei`id` metadati del documento, utilizzando la`BY` clausola.

```
SELECT
    r_id,
    r.VIN
FROM
    VehicleRegistration AS r BY r_id
WHERE
    r_id = 'documentId'
```

Di seguito viene utilizzata la`BY` clausola per unire le`Person` tabelle`DriversLicense` e le tabelle rispettivamente nei rispettivi`id` campi`PersonId` e nel campo del documento.

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
ON d.PersonId = pid
WHERE pid = 'documentId'
```

Di seguito viene utilizzato[Visione impegnata](#) per unire le`Person` tabelle`DriversLicense` e nei rispettivi`id` campi`PersonId` e nel campo del documento.

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS cp
ON d.PersonId = cp.metadata.id
WHERE cp.metadata.id = 'documentId'
```

Quanto segue restituisce il numero ordinale`PersonId` e l'indice di ogni persona nell'`Owners.SecondaryOwners`elenco di un documento nella tabella`VehicleRegistration`.

```
SELECT s.PersonId, owner_idx
FROM VehicleRegistration AS r, @r.Owners.SecondaryOwners AS s AT owner_idx
```

```
WHERE r.VIN = 'KM8SRDHF6EU074761'
```

Esecuzione a livello di codice utilizzando il driver

Per imparare a eseguire questa istruzione a livello di codice utilizzando il driver QLDB, consulta i seguenti tutorial in Guida introduttiva al driver:

- Giava:Guida di avvio rapido |Documentazione di riferimento del libro
- .NET:Guida introduttiva rapida al tutorial |Documentazione di riferimento del libro
- Vai:Fase 4 di Quick Start |Riferimento del libro di cucina
- Node.js:Guida di avvio rapido |Riferimento del libro di cucina
- Pitone:Guida di avvio rapido |Riferimento del libro di cucina

Comando UPDATE in Amazon QLDB

In Amazon QLDB, utilizzate ilUPDATE comando per modificare il valore di uno o più elementi all'interno di un documento. Se un elemento non esiste, viene inserito.

È inoltre possibile utilizzare questo comando per inserire e rimuovere in modo esplicito elementi specifici all'interno di un documento, in modo simile alleDA (INSERISCI, RIMUOVI o IMPOSTA) istruzioni.

> **Note**
>
> Per informazioni su come controllare l'accesso per eseguire questo comando PartiQL su tabelle specifiche, vedereGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

Argomenti

Sintassi

SET DI AGGIORNAMENTO

Aggiorna uno o più elementi all'interno di un documento. Se un elemento non esiste, viene inserito. Semanticamente è la stessa dell'istruzione FROM-SET.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
SET element = data [, element = data, ... ]
[ WHERE condition ]
```

AGGIORNA-INSERIMENTO

Inserisci un nuovo elemento in un documento esistente. Per inserire un nuovo documento di primo livello in una tabella, è necessario utilizzareINSERT.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
INSERT INTO element VALUE data [ AT key_name ]
[ WHERE condition ]
```

AGGIORNA-RIMUOVI

Rimuovi un elemento esistente all'interno di un documento o rimuovi un intero documento di primo livello. Quest'ultima è semanticamente uguale allaDELETE sintassi tradizionale.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
REMOVE element
[ WHERE condition ]
```

Parametri

table_name

:il nome della tabella utente contenente i dati da modificare. Le istruzioni DML sono supportate solo nella visualizzazione utente predefinita. Ogni istruzione può essere eseguita solo su una singola tabella.

Tabella_alias AS

(Facoltativo) Un alias definito dall'utente che si estende su una tabella da aggiornare. La`AS` parola chiave è facoltativa.

DI ***id_alias***

(Facoltativo) Un alias definito dall'utente che si lega al campo dei`id` metadati di ogni documento nel set di risultati. L'alias deve essere dichiarato nella`UPDATE` clausola utilizzando la`BY` parola chiave. Ciò è utile quando si desidera filtrare in base all'ID del documento mentre si esegue una query sulla visualizzazione utente predefinita. Per ulteriori informazioni, consulta Utilizzo della clausola BY per interrogare l'ID del documento.

elemento

Un elemento di documento da creare o modificare.

dati

Un nuovo valore per l'elemento.

Nome_chiave AT

Un nome chiave da aggiungere nei documenti da modificare. È necessario specificare il corrispondente`VALUE` insieme al nome della chiave. Ciò è necessario per inserire un nuovo valore in`AT` una posizione specifica all'interno di un documento.

WHERE ***condition***

I criteri di selezione per i documenti da modificare.

> **Note**
>
> Se si`WHERE` omette la clausola, saranno modificati tutti i documenti della tabella.

Valore restituito

`documentId`— L'ID univoco di ogni documento che hai aggiornato.

Esempi

Aggiorna un campo in un documento. Se il campo non esiste, viene inserito.

```
UPDATE Person AS p
SET p.LicenseNumber = 'HOLLOR123ZZ'
```

```
WHERE p.GovId = '111-22-3333'
```

Filtrare in base al campo dei`id` metadati del documento assegnato dal sistema.

```
UPDATE Person AS p BY pid
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE pid = 'documentId'
```

Sovrascrivi un intero documento.

```
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'
```

Modifica il`PersonId` campo del primo elemento dell'`Owners.SecondaryOwners`elenco all'interno di un documento.

```
UPDATE VehicleRegistration AS r
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
WHERE r.VIN = '1N4AL11D75C109151'
```

Inserisci`{'Mileage':26500}` come coppia nome-valore di primo livello all'interno di un documento nella`Vehicle` tabella.

```
UPDATE Vehicle AS v
INSERT INTO v VALUE 26500 AT 'Mileage'
WHERE v.VIN = '1N4AL11D75C109151'
```

Aggiungi`{'PersonId':'abc123'}` come coppia nome-valore nel`Owners.SecondaryOwners` campo di un documento nella`VehicleRegistration` tabella. Nota che`Owners.SecondaryOwners` deve già esistere e deve essere un tipo di dati di elenco affinché questa dichiarazione sia valida. Altrimenti, la parola chiave`AT` è obbligatoria nella`INSERT INTO` clausola.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
WHERE r.VIN = '1N4AL11D75C109151'
```

Inserisci`{'PersonId':'abc123'}` come primo elemento nell'`Owners.SecondaryOwners`elenco esistente all'interno di un documento.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
WHERE r.VIN = '1N4AL11D75C109151'
```

Aggiungi più coppie nome-valore all'`Owners.SecondaryOwners`elenco esistente all'interno di un documento.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
WHERE r.VIN = '1N4AL11D75C109151'
```

Rimuove un elemento esistente all'interno di un documento.

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

Rimuove un intero documento da una tabella.

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

Rimuove il primo elemento dell'`Owners.SecondaryOwners`elenco all'interno di un documento della`VehicleRegistration` tabella.

```
UPDATE VehicleRegistration AS r
REMOVE r.Owners.SecondaryOwners[0]
WHERE r.VIN = '1N4AL11D75C109151'
```

Esecuzione a livello di codice utilizzando il driver

Per imparare a eseguire questa istruzione a livello di codice utilizzando il driver QLDB, consulta i seguenti tutorial in Guida introduttiva al driver:

- Giava:Guida di avvio rapido |Documentazione di riferimento del libro
- .NET:Guida introduttiva rapida al tutorial |Documentazione di riferimento del libro
- Vai:Fase 4 di Quick Start |Riferimento del libro di cucina
- Node.js:Guida di avvio rapido |Riferimento del libro di cucina
- Pitone:Guida di avvio rapido |Riferimento del libro di cucina

Comando UNDROP TABLE in Amazon QLDB

In Amazon QLDB, usa il`UNDROP TABLE` comando per riattivare una tabella precedentemente eliminata (ovvero disattivata). La disattivazione o la riattivazione di una tabella non ha alcun effetto sui relativi documenti o indici.

> **Note**
>
> Per informazioni su come controllare l'accesso per eseguire questo comando PartiQL su tabelle specifiche, vedereGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

Argomenti

- Sintassi
- Parametri
- Valore restituito
- Esempi

Sintassi

```
UNDROP TABLE "tableId"
```

Parametri

«*ID della tabella*»

L'ID univoco della tabella da riattivare, indicato tra virgolette doppie.

La tabella deve essere stata eliminata in precedenza, il che significa che esiste nella tabella del catalogo di sistema`information_schema.user_tables` e ha lo stato di`INACTIVE`. Inoltre, non deve esserci alcuna tabella esistente attiva con lo stesso nome.

Valore restituito

`tableId`—:il nome univoco della tabella che hai riattivato.

Esempi

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

Funzioni PartiQL in Amazon QLDB

PartiQL in Amazon QLDB supporta le seguenti varianti incorporate di funzioni standard SQL.

> **Note**
>
> Tutte le funzioni SQL non incluse in questo elenco non sono attualmente supportate in QLDB. Questo riferimento alla funzione si basa sulla documentazione PartiQL Built-in Functions.

Propagazione di tipo sconosciuto (nulla e mancante)

Salvo diversa indicazione, tutte queste funzioni propagano valori di argomenti nulli e mancanti. La propagazione di`NULL` o`MISSING` è definita come restituzione`NULL` se un argomento della funzione è`NULL` o`MISSING`. Di seguito sono riportati alcuni esempi di questa propagazione.

```
CHAR_LENGTH(null)    -- null
CHAR_LENGTH(missing) -- null (also returns null)
```

Funzioni di aggregazione

- AVG
- COUNT
- MAX
- MIN
- SIZE
- SUM

Funzioni condizionali

- COALESCE
- EXISTS
- NULLIF

Funzioni di data e ora

- DATE_ADD
- DATE_DIFF
- EXTRACT
- UTCNOW

Funzioni scalari

- TXID

Funzioni stringa

- CHAR_LENGTH
- CHARACTER_LENGTH
- LOWER
- SUBSTRING

- TRIM
- UPPER

Funzioni di formattazione del tipo di dati

- CAST
- TO_STRING
- TO_TIMESTAMP

Funzione AVG in Amazon QLDB

In Amazon QLDB, utilizza la`AVG` funzione per restituire la media (media aritmetica) dei valori delle espressioni di input. Questa funzione funziona con valori numerici e ignora i valori nulli o mancanti.

Sintassi

```
AVG ( expression )
```

Argomenti

espressione

Il nome o l'espressione del campo o l'espressione di un tipo di dati numerico su cui viene eseguita la funzione.

Tipi di dati

Tipi di argomenti supportati:

- `int`
- `decimal`
- `float`

Tipo di reso:`decimal`

Esempi

```
SELECT AVG(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 147.19
SELECT AVG(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 2.
```

Funzioni correlati correlati correlati

- COUNT
- MAX
- MIN
- SIZE
- SUM

Funzione CAST in Amazon QLDB

In Amazon QLDB, utilizza la`CAST` funzione per valutare una determinata espressione in un valore e convertirlo in un tipo di dati di destinazione specificato. Se la conversione non può essere eseguita, la funzione restituisce un errore.

Sintassi

```
CAST ( expression AS type )
```

Argomenti

espressione

Il nome o l'espressione del campo che restituisce un valore convertito dalla funzione. La conversione di valori null restituisce null. Questo parametro può essere uno qualsiasi dei supportatiTipi di dati.

tipo

Il nome del tipo di dati di destinazione per la conversione. Questo parametro può essere uno dei supportatiTipi di dati.

Tipo restituito

Il tipo di dati specificato dall'argomento *type*.

Esempi

Gli esempi seguenti mostrano la propagazione di tipi sconosciuti (`NULL`o`MISSING`).

```
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)
```

Qualsiasi valore che non sia di tipo sconosciuto non può essere convertito in`NULL` o`MISSING`.

```
CAST(true AS null)    -- error
CAST(true AS missing) -- error
CAST(1    AS null)    -- error
CAST(1    AS missing) -- error
```

Gli esempi seguenti mostrano il 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'`)
```

Gli esempi seguenti mostrano il 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
```

Gli esempi seguenti mostrano il 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
```

Gli esempi seguenti mostrano il casting`AS 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
```

Gli esempi seguenti mostrano il casting`AS 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
```

Gli esempi seguenti mostrano il 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'
```

Gli esempi seguenti mostrano il 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"
```

Gli esempi seguenti mostrano il castingAS `struct`.

```
CAST(`{ a: 1 }` AS struct) -- {a:1}
CAST(true       AS struct) -- err
```

Gli esempi seguenti mostrano il 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
```

Gli esempi seguenti sono istruzioni eseguibili che includono alcuni degli esempi precedenti.

```
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"
```

Funzioni correlati correlati correlati

- TO_STRING
- TO_TIMESTAMP

Funzione CHAR_LENGTH in Amazon QLDB

In Amazon QLDB, utilizza laCHAR_LENGTH funzione per restituire il numero di caratteri nella stringa specificata, dove il carattere è definito come un singolo punto di codice unicode.

Sintassi

```
CHAR_LENGTH ( string )
```

CHAR_LENGTHè sinonimo diFunzione CHARACTER_LENGTH in Amazon QLDB.

Argomenti

Stringa

Il nome del campo o l'espressione del tipo di datistring che la funzione valuta.

Tipo restituito

`int`

Esempi

```
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)
```

Funzioni correlati correlati correlati

- LOWER
- SUBSTRING

- TRIM
- UPPER

Funzione CHARACTER_LENGTH in Amazon QLDB

Sinonimo della funzione `CHAR_LENGTH`.

Consultare Funzione CHAR_LENGTH in Amazon QLDB.

Funzione COALESCE in Amazon QLDB

In Amazon QLDB, dato un elenco di uno o più argomenti, utilizza la`COALESCE` funzione per valutare gli argomenti in ordine da sinistra a destra e restituire il primo valore che non sia di tipo (`NULL`o`MISSING`) sconosciuto. Se tutti i tipi di argomento sono sconosciuti, il risultato è`NULL`.

La`COALESCE` funzione non si propaga`NULL` e`MISSING`.

Sintassi

```
COALESCE ( expression [, expression, ... ] )
```

Argomenti

`espressione`

L'elenco di uno o più nomi o espressioni di campo valutati dalla funzione. Ogni argomento può essere uno qualsiasi dei supportatiTipi di dati.

Tipo restituito

Qualsiasi tipo di dati supportato. Il tipo restituito è ugualeNULL o uguale al tipo della prima espressione che restituisce un valore non nullo e non mancante.

Esempi

```
SELECT COALESCE(1, null) FROM << 0 >>          -- 1
SELECT COALESCE(null, null, 1) FROM << 0 >>    -- 1
SELECT COALESCE(null, 'string') FROM << 0 >>  -- "string"
```

Funzioni correlati correlati correlati

- EXISTS
- NULLIF

Funzione COUNT in Amazon QLDB

In Amazon QLDB, utilizza la`COUNT` funzione per restituire il numero di documenti definiti dall'espressione specificata. Questa funzione è disponibile in due varianti:

- `COUNT(*)`— Conta tutti i documenti nella tabella di destinazione, indipendentemente dal fatto che includano o meno valori nulli o mancanti.
- `COUNT(expression)`— Calcola il numero di documenti con valori non nulli in un campo o un'espressione specifica esistente.

> **Warning**
>
> La`COUNT` funzione non è ottimizzata, quindi non è consigliabile utilizzarla senza una ricerca indicizzata. Quando si esegue una query in QLDB senza una ricerca indicizzata, viene richiamata una scansione completa della tabella. Ciò può causare problemi di prestazioni su tabelle di grandi dimensioni, inclusi conflitti di concorrenza e timeout delle transazioni.
> Per evitare la scansione delle tabelle, è necessario eseguire istruzioni con una clausola`WHERE` predicativa utilizzando un operatore di uguaglianza (`=` o`IN`) su un campo indicizzato o un ID di documento. Per ulteriori informazioni, consulta Ottimizzazione delle prestazioni delle query.

Sintassi

```
COUNT ( * | expression )
```

Argomenti

espressione

Il nome o l'espressione del campo su cui viene eseguita la funzione. Questo parametro può essere uno qualsiasi dei supportatiTipi di dati.

Tipo restituito

`int`

Esempi

```
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
```

Funzioni correlati correlati correlati

- AVG
- MAX
- MIN
- SIZE
- SUM

Funzione DATE_ADD in Amazon QLDB

In Amazon QLDB, utilizza la`DATE_ADD` funzione per incrementare un determinato valore di timestamp in base a un intervallo specificato.

Sintassi

```
DATE_ADD( datetimepart, interval, timestamp )
```

Argomenti

`parte data/ora`

Parte relativa a data o ora su cui viene eseguita la funzione. Questo parametro può essere uno dei seguenti:

- `year`
- `month`
- `day`
- `hour`

- `minute`
- `second`

`intervallo`

Il numero intero che specifica l'intervallo da aggiungere al *`timestamp`* specificato. Un intero negativo sottrae l'intervallo.

`timestamp`

Il nome del campo o l'espressione del tipo di dati`timestamp` che la funzione incrementa.

Un valore letterale del timestamp di Ion può essere indicato con backticks (`` `...` ``). Per dettagli sulla formattazione ed esempi di valori di timestamp, consulta Timestamp nel documento con le specifiche di Amazon Ion.

Tipo restituito

`timestamp`

Esempi

```
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
```

Funzioni correlati correlati correlati

- DATE_DIFF
- EXTRACT
- TO_STRING

- TO_TIMESTAMP
- UTCNOW

Funzione DATE_DIFF in Amazon QLDB

In Amazon QLDB, utilizza la`DATE_DIFF` funzione per restituire la differenza tra le parti di data specificate di due determinati timestamp.

Sintassi

```
DATE_DIFF( datetimepart, timestamp1, timestamp2 )
```

Argomenti

`parte data/ora`

Parte relativa a data o ora su cui viene eseguita la funzione. Questo parametro può essere uno dei seguenti:

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`

`timestamp1`, *`timestamp2`*

I due nomi di campo o espressioni del tipo di dati`timestamp` che la funzione confronta. Se *`timestamp2`* è successivo a *`timestamp1`*, il risultato è positivo. Se *`timestamp2`* è precedente a *`timestamp1`*, il risultato è negativo.

Un valore letterale del timestamp di Ion può essere indicato con backticks (`` `...` ``). Per dettagli sulla formattazione ed esempi di valori di timestamp, consulta Timestamp nel documento con le specifiche di Amazon Ion.

Tipo restituito

`int`

Esempi

```
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
```

Funzioni correlati correlati correlati

- DATE_ADD
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

Funzione EXISTS in Amazon QLDB

In Amazon QLDB, dato un valore PartiQL, viene eseguita la`EXISTS` funzione per restituire`TRUE` se il valore è una raccolta non vuota. Altrimenti, questa funzione restituisce`FALSE`. Se l'input to non`EXISTS` è un contenitore, il risultato è`FALSE`.

La`EXISTS` funzione non si propaga`NULL` e`MISSING`.

Sintassi

```
EXISTS ( value )
```

Argomenti

value

Il nome o l'espressione del campo o l'espressione valutata dalla funzione. Questo parametro può essere uno qualsiasi dei supportatiTipi di dati.

Tipo restituito

`bool`

Esempi

```
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
```

Funzioni correlati correlati correlati

- COALESCE
- NULLIF

Funzione EXTRACT in Amazon QLDB

In Amazon QLDB, utilizza laEXTRACT funzione per restituire il valore intero di una parte di data o ora specificata da un determinato timestamp.

Sintassi

```
EXTRACT ( datetimepart FROM timestamp )
```

Argomenti

parte data/ora

La parte di data o ora che la funzione estrae. Questo parametro può essere uno dei seguenti:

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`
- `timezone_hour`
- `timezone_minute`

timestamp

Il nome del campo o l'espressione del tipo di dati `datimestamp` cui la funzione estrae. Se questo parametro è di tipo (`NULL`o`MISSING`) sconosciuto, la funzione restituisce`NULL`.

Un valore letterale del timestamp di Ion può essere indicato con backticks (`` `...` ``). Per dettagli sulla formattazione ed esempi di valori di timestamp, consulta Timestamp nel documento con le specifiche di Amazon Ion.

Tipo restituito

`int`

Esempi

```
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
```

Funzioni correlati correlati correlati

- DATE_ADD
- DATE_DIFF
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

Funzione LOWER in Amazon QLDB

In Amazon QLDB, usa la`LOWER` funzione per convertire tutti i caratteri maiuscoli in caratteri minuscoli in una determinata stringa.

Sintassi

```
LOWER ( string )
```

Argomenti

`Stringa`

Il nome del campo o l'espressione del tipo di dati`string` che la funzione converte.

Tipo restituito

`string`

Esempi

```
SELECT LOWER('AbCdEfG!@#$') FROM << 0 >> -- 'abcdefg!@#$'
```

Funzioni correlati correlati correlati

- CHAR_LENGTH
- SUBSTRING
- TRIM
- UPPER

Funzione MAX in Amazon QLDB

In Amazon QLDB, utilizza la`MAX` funzione per restituire il valore massimo in un set di valori numerici.

Sintassi

```
MAX ( expression )
```

Argomenti

`espressione`

Il nome o l'espressione del campo o l'espressione di un tipo di dati numerico su cui viene eseguita la funzione.

Tipi di dati

Tipi di argomenti supportati:

- `int`
- `decimal`
- `float`

Tipi di reso supportati:

- `int`
- `decimal`
- `float`

Esempi

```
SELECT MAX(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 442.30
SELECT MAX(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 3
```

Funzioni correlati correlati correlati

- AVG
- COUNT
- MIN
- SIZE
- SUM

Funzione MIN in Amazon QLDB

In Amazon QLDB, utilizza la`MIN` funzione per restituire il valore minimo in un set di valori numerici.

Sintassi

```
MIN ( expression )
```

Argomenti

espressione

Il nome o l'espressione del campo o l'espressione di un tipo di dati numerico su cui viene eseguita la funzione.

Tipi di dati

Tipi di argomenti supportati:

- `int`
- `decimal`
- `float`

Tipi di reso supportati:

- `int`
- `decimal`
- `float`

Esempi

```
SELECT MIN(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 30.45
SELECT MIN(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 1
```

Funzioni correlati correlati correlati

- AVG
- COUNT
- MAX
- SIZE
- SUM

Funzione NULLIF in Amazon QLDB

In Amazon QLDB, date due espressioni, viene eseguita la`NULLIF` funzione per restituire`NULL` se le due espressioni restituisce lo stesso valore. In caso contrario, questa funzione restituisce il risultato della valutazione della prima espressione.

La`NULLIF` funzione non si propaga`NULL` e`MISSING`.

Sintassi

```
NULLIF ( expression1, expression2 )
```

Argomenti

`expression1`, *`expression2`*

I due nomi o espressioni di campo che la funzione confronta. Questi parametri possono essere uno qualsiasi dei supportatiTipi di dati.

Tipo restituito

Qualsiasi tipo di dati supportato. Il tipo restituito è ugualeNULL o uguale al tipo della prima espressione.

Esempi

```
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
```

Funzioni correlati correlati correlati

- COALESCE
- EXISTS

Funzione SIZE in Amazon QLDB

In Amazon QLDB, utilizza laSIZE funzione per restituire il numero di elementi in un determinato tipo di dati del contenitore (elenco, struttura o borsa).

Sintassi

```
SIZE ( container )
```

Argomenti

container

Il nome o l'espressione del campo contenitore su cui viene eseguita la funzione.

Tipi di dati

Tipi di argomenti supportati:

- list
- struttura
- borsa

Tipo di reso:`int`

Se l'input to non`SIZE` è un contenitore, la funzione genera un errore.

Esempi

```
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
```

Funzioni correlati correlati correlati

- AVG
- COUNT
- MAX
- MIN
- SUM

Funzione SUBSTRING in Amazon QLDB

In Amazon QLDB, usa la`SUBSTRING` funzione per restituire una sottostringa da una determinata stringa. La sottostringa inizia dall'indice iniziale specificato e termina con l'ultimo carattere della stringa o con la lunghezza specificata.

Sintassi

```
SUBSTRING ( string, start-index [, length ] )
```

Argomenti

`Stringa`

Il nome del campo o l'espressione del tipo di dati`string` da cui estrarre una sottostringa.

`indice di partenza`

La posizione iniziale all'interno della *`stringa`* da cui iniziare l'estrazione. Questo numero può essere negativo.

Il primo carattere della *`stringa`* ha un indice pari a 1.

`length`

(Facoltativo) Il numero di caratteri (punti di codice) da estrarre dalla *`stringa`*, a partire da *`start-index`* e terminando con (*`start-index`* + *`lunghezza`*) - 1. In altre parole, la lunghezza della sottostringa. Questo numero non può essere negativo.

Se questo parametro non viene fornito, la funzione procede fino alla fine della *`stringa`*.

Tipo restituito

`string`

Esempi

```
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"
```

Funzioni correlati correlati correlati

- CHAR_LENGTH
- LOWER
- TRIM
- UPPER

Funzione SUM in Amazon QLDB

In Amazon QLDB, utilizza la`SUM` funzione per restituire la somma dei valori del campo di input o dell'espressione. Questa funzione funziona con valori numerici e ignora i valori nulli o mancanti.

Sintassi

```
SUM ( expression )
```

Argomenti

`espressione`

Il nome o l'espressione del campo o l'espressione di un tipo di dati numerico su cui viene eseguita la funzione.

Tipi di dati

Tipi di argomenti supportati:

- `int`

- `decimal`
- `float`

Tipi di reso supportati:

- `int`— Per argomenti interi
- `decimal`— Per argomenti decimali o in virgola mobile

Esempi

```
SELECT SUM(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 735.95
SELECT SUM(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 6
```

Funzioni correlati correlati correlati

- AVG
- COUNT
- MAX
- MIN
- SIZE

Funzione TO_STRING in Amazon QLDB

In Amazon QLDB, utilizza la`TO_STRING` funzione per restituire una rappresentazione in formato stringa di un determinato timestamp nel modello di formato specificato.

Sintassi

```
TO_STRING ( timestamp, 'format' )
```

Argomenti

`timestamp`

Il nome del campo o l'espressione del tipo di dati`timestamp` che la funzione converte in una stringa.

Un valore letterale del timestamp di Ion può essere indicato con backticks (`` `...` ``). Per dettagli sulla formattazione ed esempi di valori di timestamp, consulta Timestamp nel documento con le specifiche di Amazon Ion.

`format`

La stringa letterale che specifica il modello di formato del risultato, in termini di parti di data. Per i formati validi, consultare Stringhe in formato timestamp.

Tipo restituito

`string`

Esempi

```
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"
```

Funzioni correlati correlati correlati

- CAST
- DATE_ADD
- DATE_DIFF

- EXTRACT
- TO_TIMESTAMP
- UTCNOW

Funzione TO_TIMESTAMP in Amazon QLDB

In Amazon QLDB, data una stringa che rappresenta un timestamp, usa la`TO_TIMESTAMP` funzione per convertire la stringa in un tipo di`timestamp` dati. Questa è l'operazione inversa di`TO_STRING`.

Sintassi

```
TO_TIMESTAMP ( string [, 'format' ] )
```

Argomenti

`Stringa`

Il nome del campo o l'espressione del tipo di dati`string` che la funzione converte in un timestamp.

`format`

(Facoltativo) La stringa letterale che definisce il modello di formato della *`stringa`* di input, in termini di parti di data. Per i formati validi, consultare Stringhe in formato timestamp.

Se questo argomento viene omesso, la funzione presuppone che la *`stringa`* abbia il formato di un timestamp Ion standard. Questo è il modo consigliato per analizzare un timestamp Ion utilizzando questa funzione.

La spaziatura interna zero è facoltativa quando si utilizza un simbolo di formato a carattere singolo (ad esempio`yMdHh`,`m`,,,,,`s`) ma è necessario per le relative varianti con riempimento zero (ad esempio`yyyyMMdd`,`HH`,,`hh`,`mm`,`ss`).

Un trattamento speciale è riservato agli anni a due cifre (simbolo del formato`yy`). 1900 viene aggiunto ai valori maggiori o uguali a 70 e 2000 ai valori inferiori a 70.

I nomi dei mesi e gli indicatori AM o PM non fanno distinzione tra maiuscole e minuscole.

Tipo restituito

`timestamp`

Esempi

```
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
```

Funzioni correlati correlati correlati

- CAST
- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- UTCNOW

Funzione TRIM in Amazon QLDB

In Amazon QLDB, usa la`TRIM` funzione per tagliare una determinata stringa rimuovendo gli spazi vuoti iniziali e finali o un set di caratteri specificato.

Sintassi

```
TRIM ( [ LEADING | TRAILING | BOTH [ characters ] FROM ] string)
```

Argomenti

LEADING

(Facoltativo) Indica che gli spazi vuoti o i caratteri specificati devono essere rimossi dall'inizio della *stringa*. Se non specificato, il comportamento predefinito è`BOTH`.

TRAILING

(Facoltativo) Indica che gli spazi vuoti o i caratteri specificati devono essere rimossi dalla fine della *stringa*. Se non specificato, il comportamento predefinito è`BOTH`.

BOTH

(Facoltativo) Indica che gli spazi vuoti iniziali e finali o i caratteri specificati devono essere rimossi dall'inizio e dalla fine della *stringa*.

characters

(Facoltativo) Il set di caratteri da rimuovere, specificato come`string`.

Se questo parametro non viene fornito, gli spazi vuoti vengono rimossi.

Stringa

Il nome del campo o l'espressione del tipo di dati`string` che le funzioni tagliano.

Tipo restituito

`string`

Esempi

```
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         "
```

Funzioni correlati correlati correlati

- CHAR_LENGTH
- LOWER
- SUBSTRING
- UPPER

Funzione TXID in Amazon QLDB

In Amazon QLDB, utilizza la`TXID` funzione per restituire l'ID di transazione univoco dell'istruzione corrente che stai eseguendo. Questo è il valore assegnato al campo dei`txId` metadati di un documento quando la transazione corrente viene salvata nel giornale.

Sintassi

```
TXID()
```

Argomenti

Nessuno

Tipo restituito

`string`

Esempi

```
SELECT TXID() FROM << 0 >>  -- "L7S9iJqcn9W2M4qOEn27ay"
SELECT TXID() FROM Person WHERE GovId = 'LEWISR261LL'  -- "BKeMb48PNyvHWJGZHkaodG"
```

Funzione UPPER in Amazon QLDB

In Amazon QLDB, usa la`UPPER` funzione per convertire tutti i caratteri minuscoli in caratteri maiuscoli in una determinata stringa.

Sintassi

```
UPPER ( string )
```

Argomenti

`Stringa`

Il nome del campo o l'espressione del tipo di dati`string` che la funzione converte.

Tipo restituito

`string`

Esempi

```
SELECT UPPER('AbCdEfG!@#$') FROM << 0 >> -- 'ABCDEFG!@#$'
```

Funzioni correlati correlati correlati

- CHAR_LENGTH
- LOWER
- SUBSTRING
- TRIM

Funzione UTCNOW in Amazon QLDB

In Amazon QLDB, utilizza la`UTCNOW` funzione per restituire l'ora corrente in UTC (Coordinated Universal Time) come`timestamp`.

Sintassi

```
UTCNOW()
```

Questa funzione richiede di specificare una`FROM` clausola in un'`SELECT`interrogazione.

Argomenti

Nessuno

Tipo restituito

`timestamp`

Esempi

```
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'
```

Funzioni correlati correlati correlati

- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP

Stringhe in formato timestamp

Questa sezione fornisce informazioni di riferimento per le stringhe in formato timestamp.

Le stringhe in formato timestamp si applicano alle`TO_TIMESTAMP` funzioni`TO_STRING` and. Queste stringhe possono contenere separatori di parti di data (come`-` ",`/` '`o`':` ') e i seguenti simboli di formato.

Formato	Esempio	Descrizione
yy	70	Anno a due cifre
y	1970	Anno a quattro cifre
yyyy	1970	Anno a quattro cifre con l'aggiunta di zero
`M`	1	Intero intero mese
`MM`	01	Intero intero mese con l'aggiunta di zero

Formato	Esempio	Descrizione
`MMM`	Jan	Nome abbreviato del mese abbreviato abbreviato
`MMMM`	Gennaio	Nome completo del mese completo
`d`	2	Giorno del mese (1—31)
`dd`	02	Giorno del mese con l'aggiunta di zero (01-31)
`a`	AM o PM	Indicatore meridiano (per orologio a 12 ore)
`h`	3	Ora (formato 12 ore, 1—12)
`hh`	03	Ora con l'aggiunta di zero (formato 12 ore, 01-12)
`H`	3	Ora (formato 24 ore, 0—23)
`HH`	03	Ora con l'aggiunta di zero (formato 24 ore, 00-23)
`m`	4	Minuti (0—59)
`mm`	04	Minuti preimpostati a zero (00—59)
`s`	5	Secondi (0—59)
`ss`	05	Zero secondi riempiti (00—59)
`S`	0	Frazione di secondo (precisione: 0,1, intervallo: 0,0-0,9)
`SS`	06	Frazione di secondo (precisione: 0,01, intervallo: 0,0—0,99)

Formato	Esempio	Descrizione
SSS	060	Frazione di secondo (precisio ne: 0,001, intervallo: 0,0— 0,999)
X	+07 o Z	Offset rispetto a UTC in ore o «Z» se l'offset è 0
XX	+0700 o Z	Offset rispetto a UTC in ore e minuti o «Z» se l'offset è 0
XXX	+ 07:00 o Z	Offset rispetto a UTC in ore e minuti o «Z» se l'offset è 0
x	+7	Compensazione da UTC in ore
xx	+0700	Offset rispetto a UTC in ore e minuti
xxx	+ 07:00	Offset rispetto a UTC in ore e minuti

Stored

In Amazon QLDB, puoi utilizzare ilEXEC comando per eseguire le stored procedure PartiQL nella seguente sintassi.

```
EXEC stored_procedure_name argument [, ... ]
```

QLDB supporta solo le seguenti Procedure ure ure ure ure ure ure ure ure ure ure ure ure ure I

Argomenti

- Stored

Stored

> **Note**
>
> I registri creati prima del 22 luglio 2021 non sono attualmente idonei alla redazione. Puoi visualizzare l'ora di creazione del tuo libro contabile sulla console Amazon QLDB.

In Amazon QLDB, utilizza la proceduraREDACT_REVISION archiviata per eliminare definitivamente una singola revisione di un documento inattiva sia nell'archiviazione indicizzata che nell'archiviazione del giornale. Questa procedura memorizzata elimina tutti i dati utente nella revisione specificata. Tuttavia, lascia invariati la sequenza del diario e i metadati del documento, inclusi l'ID e l'hash del documento. Questa operazione è irreversibile.

La revisione del documento specificata deve essere una revisione inattiva nella cronologia. L'ultima revisione attiva di un documento non è idonea alla redazione.

Dopo aver inviato una richiesta di redazione eseguendo questa procedura memorizzata, QLDB elabora la redazione dei dati in modo asincrono. Al termine della redazione, i dati utente della revisione specificata (rappresentati dalladata struttura) vengono sostituiti da un nuovodataHash campo. Il valore di questo campo è l'hash Amazon Ion delladata struttura rimossa. Di conseguenza, il libro mastro mantiene l'integrità complessiva dei dati e rimane verificabile crittograficamente tramite le operazioni API di verifica esistenti.

Per un esempio di operazione di redazione con dati di esempio, vedereEsempio di redazione inRedazione delle revisioni dei documenti.

> **Note**
>
> Per informazioni su come controllare l'accesso per eseguire questo comando PartiQL su tabelle specifiche, vedereGuida introduttiva alla modalità di autorizzazione standard in Amazon QLDB.

Argomenti

- Considerazioni e limitazioni
- Sintassi

- Argomenti
- Valore restituito
- Esempi

Considerazioni e limitazioni

Prima di iniziare con la redazione dei dati in Amazon QLDB, assicurati di esaminare le seguenti considerazioni e limitazioni:

- La proceduraREDACT_REVISION memorizzata indirizza i dati utente in una revisione individuale e inattiva del documento. Per oscurare più revisioni, è necessario eseguire la stored procedure una volta per ogni revisione. Puoi oscurare una revisione per transazione.
- Per oscurare determinati campi all'interno di una revisione del documento, è necessario utilizzare un'istruzione DML (Data Manipulation Language) separata per modificare prima la revisione. Per ulteriori informazioni, consulta Redazione di un campo particolare all'interno di una revisione.
- Dopo che QLDB ha ricevuto una richiesta di redazione, non è possibile annullare o modificare la richiesta. Per confermare se una redazione è completa, puoi verificare se la`data` struttura di una revisione è stata sostituita da un`dataHash` campo. Per ulteriori informazioni, consulta Verificare se una redazione è completa.
- La redazione non ha alcun impatto sui dati QLDB replicati al di fuori del servizio QLDB. Ciò include tutte le esportazioni verso Amazon S3 e gli stream verso Amazon Kinesis Data Streams. È necessario utilizzare altri metodi di conservazione dei dati per gestire i dati archiviati al di fuori di QLDB.
- La redazione non ha alcun impatto sui valori letterali nelle dichiarazioni PartiQL registrate nel giornale. Come best practice, si consiglia di eseguire le istruzioni con parametri di codice, utilizzando segnaposti variabili anziché valori letterali. Un segnaposto viene scritto nel diario come punto interrogativo (?) anziché come qualsiasi informazione sensibile che potrebbe richiedere una revisione.

  Per scoprire come eseguire istruzioni PartiQL in modo programmatico utilizzando il driver QLDB, consulta i tutorial per ogni linguaggio di programmazione supportato inNozioni base sul driver.

Sintassi

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

Argomenti

`*indirizzo di blocco*`

La posizione del blocco del diario della revisione del documento da oscurare. Un indirizzo è una struttura Amazon Ion con due campi:`strandId` e `sequenceNo`.

Questo è un valore letterale di Ion che viene indicato con contrassegni. Ad esempio:

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

Per informazioni su come trovare l'indirizzo del blocco, consultaInterrogazione dei metadati dei documenti.

'*table-id*'

L'ID univoco della tabella di cui si desidera oscurare la revisione del documento, indicato da virgolette singole.

Per informazioni su come trovare l'ID della tabella, consultaInterrogazione del catalogo di sistema.

'*id del documento*'

L'ID univoco del documento della revisione da oscurare, indicato da virgolette singole.

Per informazioni su come trovare l'ID del documento, consultaInterrogazione dei metadati dei documenti.

Valore restituito

Una struttura Amazon Ion che rappresenta la revisione del documento da oscurare, nel seguente formato.

```
{
  blockAddress: {
    strandId: String,
    sequenceNo: Int
  },
  tableId: String,
  documentId: String,
  version: Int
}
```

Restituisci i campi della struttura

- `blockAddress`— La posizione del blocco del diario in cui si trova la revisione da redigere. Un indirizzo ha i due campi seguenti:
  - `strandId`— L'ID univoco del filone del journal che contiene il blocco.
  - `sequenceNo`— Un numero di indice che specifica la posizione del blocco all'interno del filamento.
- `tableId`— L'ID univoco della tabella di cui stai oscurando la revisione.
- `documentId`— L'ID univoco del documento della revisione da oscurare.
- `version`— Il numero di versione della revisione del documento da oscurare.

L'esempio seguente mostra la struttura la struttura della struttura del valore di restituzione, si consiglia di eseguire le proprie funzioni.

```
{
  blockAddress: {
    strandId: "CsRnx0RDoNK6ANEEePa1ov",
    sequenceNo: 134
  },
  tableId: "6GZumdHggklLdMGyQq9DNX",
  documentId: "IXlQPSbfyKMIIsygePeKrZ",
  version: 0
}
```

Esempi

```
EXEC REDACT_REVISION `{strandId:"7z2P0AyQKWD8oFYmGNhi8D", sequenceNo:7}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

Operatori PartiQL in Amazon QLDB

PartiQL in Amazon QLDB supporta i seguenti operatori standard SQL.

> **Note**
>
> Tutti gli operatori SQL che non sono inclusi in questo elenco non sono attualmente supportati in QLDB.

Operatori aritmetici

Operatore	Descrizione
+	Add
-	Subtract
*	Multiply (Moltiplica)
/	Divide (Dividi)
%	Modulo

Operatori di confronto

Operatore	Descrizione
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not Equal to qual to

Operatori logici

Operatore	Descrizione
AND	TRUE se tutte le condizioni separate da AND sono TRUE

Operatore	Descrizione
`BETWEEN`	`TRUE` se l'operando è all'interno dell'intervallo di confronti
`IN`	`TRUE` se l'operando è uguale a uno di un elenco di espressioni
`IS`	`TRUE`se l'operando è un determinato tipo di dati, inclusoNULL oMISSING
`LIKE`	`TRUE`se l'operando corrisponde a uno schema
`NOT`	Inverte il valore di una determinata espressione booleana
`OR`	`TRUE`se una delle condizioni separate da`OR` è`TRUE`

Operatori di stringa

Operatore	Descrizione
`||`	Concatena due stringhe su entrambi i lati dell'`||`operatore e restituisce la stringa concatenata. Se una o entrambe le stringhe sono`NULL`, il risultato della concatenazione è nullo.

Parole chiave riservate in Amazon QLDB

Di seguito è riportato un elenco delle parole chiave PartiQL riservate in Amazon QLDB. È possibile utilizzare una parola chiave riservata come identificatore tra virgolette doppie (ad esempio,`"user"`). Per informazioni sulle convenzioni di quotazione di PartiQL in QLDB, vedereInterrogare Ion con PartiQL.

Important

Le parole chiave in questo elenco sono tutte considerate riservate perché PartiQL è retrocompatibile con SQL-92. Tuttavia, QLDB supporta solo un sottoinsieme di queste parole riservate. Per l'elenco delle parole chiave SQL attualmente supportate da QLDB, consulta i seguenti argomenti:

- Funzioni PartiQL
- Operatori QL PartiQL
- Comandi 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
```

Riferimento al formato dei dati Amazon Ion in Amazon QLDB

Amazon QLDB utilizza un modello di notazione dei dati che unifica Amazon Ion con un sottoinsieme di tipi PartiQL. Questa sezione fornisce una panoramica di riferimento del formato di dati del documento Ion, separata dalla sua integrazione con PartiQL.

Interrogazione di Ion con PartiQL in Amazon QLDB

Per la sintassi e la semantica dell'interrogazione dei dati Ion con PartiQL in QLDB, consultaInterrogare Ion con PartiQL il riferimento Amazon QLDB PartiQL.

Per esempi di codice che interrogano ed elaborano i dati Ion in un registro QLDB, vedereEsempi di codice Amazon Ion andUso di Amazon Ion.

Argomenti

- Cos'è Amazon Ion?
- Specificazione degli ioni
- Compatibile JSON
- Estensioni da JSON
- Esempio di testo Ion
- Riferimenti API
- Esempi di codice Amazon Ion in QLDB

Cos'è Amazon Ion?

Ion è un formato di serializzazione dei dati gerarchico open source, ricco di caratteri e autodescrittivo, originariamente sviluppato internamente ad Amazon. Si basa su un modello di dati astratto che consente di archiviare dati strutturati e non strutturati. È un superset di JSON, il che significa che qualsiasi documento JSON valido è anche un documento Ion valido. Questa guida presuppone una conoscenza operativa di base di JSON. Se non conosci già JSON, consulta Introduzione a JSON per ulteriori informazioni.

È possibile annotare i documenti Ion in modo intercambiabile in forma di testo leggibile dall'uomo o in formato con codifica binaria. Come JSON, il modulo di testo è facile da leggere e scrivere e supporta

la prototipazione rapida. La codifica binaria è più compatta ed efficiente per persistere, trasmettere e analizzare. Un processore Ion può transcodificare tra entrambi i formati per rappresentare esattamente lo stesso insieme di strutture di dati senza alcuna perdita di dati. Questa funzionalità consente alle applicazioni di ottimizzare il modo in cui elaborano i dati per diversi casi d'uso.

> **Note**
>
> Il modello di dati Ion è strettamente basato sui valori e non supporta riferimenti. Pertanto, il modello di dati può rappresentare gerarchie di dati che possono essere annidate a profondità arbitrarie, ma non grafici orientati.

Specificazione degli ioni

Per un elenco completo dei tipi di dati Ion core con descrizioni complete e dettagli di formattazione dei valori, consulta il documento sulle specifiche di Ion sul GitHub sito Amazon.

Per semplificare lo sviluppo delle applicazioni, Amazon Ion fornisce librerie client che elaborano i dati Ion per te. Per esempi di codice di casi d'uso comuni per l'elaborazione dei dati Ion, consulta Amazon Ion Cookbook su GitHub.

Compatibile JSON

Analogamente a JSON, componi documenti Amazon Ion con un set di tipi di dati primitivi e un set di tipi di contenitori definiti in modo ricorsivo. Ion include i seguenti tipi di dati JSON tradizionali:

- `null`: un valore nullo (vuoto) generico e non digitato. Inoltre, come descritto nella sezione seguente, Ion supporta un tipo nullo distinto per ogni tipo primitivo.
- `bool`: valori booleani.
- `string`: caratteri letterali di testo Unicode.
- `list`: raccolte di valori eterogenee ordinate.
- `struct`: raccolte non ordinate di coppie nome-valore. Come JSON,`struct` consente più valori per nome, ma ciò è generalmente sconsigliato.

Estensioni da JSON

Tipi di numeri

Invece delnumber tipo JSON ambiguo, Amazon Ion definisce rigorosamente i numeri come uno dei seguenti tipi:

- `int`: numeri interi firmati di dimensioni arbitrarie.
- `decimal`: numeri reali con codifica decimale di precisione arbitraria.
- `float`: numeri in virgola mobile con codifica binaria (IEEE a 64 bit).

Durante l'analisi dei documenti, un processore Ion assegna i tipi di numeri come segue:

- `int`: numeri senza esponente o punto decimale (ad esempio,`100200`).
- `decimal`: numeri con una virgola decimale e nessun esponente (ad esempio`0.00001`,`200.0`).
- `float`: numeri con un esponente, ad esempio notazione scientifica o notazione elettronica (ad esempio`2e0`,`3.1e-4`).

Nuovi tipi di dati

Amazon Ion aggiunge i seguenti tipi di dati:

- `timestamp`: momenti di data/ora/fuso orario di precisione arbitraria.
- `symbol`: atomi simbolici Unicode (ad esempio identificatori).
- `blob`: dati binari di codifica definita dall'utente.
- `clob`: dati di testo della codifica definita dall'utente.
- `sexp`: raccolte ordinate di valori con semantica definita dall'applicazione.

Tipi nulli

Oltre al tipo nullo generico definito da JSON, Amazon Ion supporta un tipo nullo distinto per ogni tipo primitivo. Ciò indica una mancanza di valore pur mantenendo un tipo di dati rigoroso.

```
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
```

Esempio di testo 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
    ],
}
```

Riferimenti API

- ion-go
- ion-java
- ion-js
- ion-pitone

Esempi di codice Amazon Ion in QLDB

Questa sezione fornisce esempi di codice che elaborano i dati di Amazon Ion leggendo e scrivendo i valori dei documenti in un registro Amazon QLDB. Gli esempi di codice utilizzano il driver QLDB per eseguire istruzioni PartiQL sul libro mastro. Questi esempi fanno parte dell'applicazione di esempio di e sono open source nelGuida introduttiva ad Amazon QLDB utilizzando un esempio di tutorial applicativo GitHub sitoAWS Samples.

Per esempi di codice generali che mostrano i casi d'uso più comuni di elaborazione dei dati Ion, consulta Amazon Ion Cookbook su GitHub.

Esecuzione del codice

Il codice tutorial per ogni linguaggio di programmazione esegue i seguenti passaggi:

1. Connect al registro`vehicle-registration` di esempio.
2. Create una tabella denominata`IonTypes`.
3. Inserisci un documento nella tabella con un solo`Name` campo.
4. Per ogni tipo di dati Ion supportato:
   a. Aggiorna il`Name` campo del documento con un valore letterale del tipo di dati.
   b. Esecuzione di query sulla tabella per ottenere l'ultima revisione del documento.
   c. Verifica che il valore di abbia`Name` mantenuto le proprietà del tipo di dati originale verificando che corrisponda al tipo previsto.
5. Getta il`IonTypes` tavolo.

> **Note**
>
> Prima di eseguire questo codice tutorial, è necessario creare un registro denominato`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()
```

Documentazione di riferimento dell'API Amazon QLDB

Questo capitolo descrive le operazioni API di basso livello per Amazon QLDB accessibili tramite HTTP,AWS Command Line Interface(AWS CLI), oppure unAWSSDK:

- Amazon QLDB— L'API di gestione delle risorse QLDB (nota anche comepiano di controllo). Questa API viene utilizzata solo per la gestione delle risorse di contabilità generale e per le operazioni di dati non transazionali. È possibile utilizzare queste operazioni per creare, eliminare, descrivere, elencare e aggiornare libri. È inoltre possibile verificare crittograficamente i dati del journal ed esportare o eseguire lo streaming di blocchi di journal.
- Sessione Amazon QLDB— L'API dei dati transazionali di QLDB. È possibile utilizzare questa API per eseguire transazioni di dati su un libro mastro conPartiQLIstruzioni .

> **Important**
>
> Invece di interagire direttamente con ilSessione QLDBAPI, si consiglia di utilizzare il driver QLDB o la shell QLDB per eseguire transazioni di dati su un libro mastro.
>
> - Se usiAWSSDK, utilizzare il driver QLDB. Il driver fornisce un livello di astrazione di alto livello sopraSessione QLDBdata API e gestisce il`SendCommand`operazione per te. Per informazioni e un elenco di linguaggi di programmazione supportati, consultaNozioni base sul driver.
> - Se usiAWS CLI, utilizzare la shell QLDB. La shell è un'interfaccia a riga di comando che utilizza il driver QLDB per interagire con un libro mastro. Per informazioni, consultare Utilizzo della shell Amazon QLDB (solo API dati).

Argomenti

Operazioni

Amazon QLDB supporta le seguenti operazioni:

- CancelJournalKinesisStream
- CreateLedger
- DeleteLedger
- DescribeJournalKinesisStream
- DescribeJournalS3Export
- DescribeLedger
- ExportJournalToS3
- GetBlock
- GetDigest
- GetRevision
- ListJournalKinesisStreamsForLedger
- ListJournalS3Exports
- ListJournalS3ExportsForLedger
- ListLedgers
- ListTagsForResource
- StreamJournalToKinesis
- TagResource
- UntagResource
- UpdateLedger
- UpdateLedgerPermissionsMode

Amazon QLDB Session supporta le seguenti operazioni:

- 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

Servizio: Amazon QLDB

Termina un determinato stream di journal Amazon QLDB. Prima che uno stream possa essere annullato, deve essere il suo stato attuale. `ACTIVE`

Non puoi riavviare uno stream dopo averlo annullato. Le risorse di streaming QLDB annullate sono soggette a un periodo di conservazione di 7 giorni, quindi vengono eliminate automaticamente dopo la scadenza di questo limite.

Sintassi della richiesta

```
DELETE /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del libro mastro.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

streamId

L'UUID (rappresentato nel testo con codifica Base62) del flusso di journal QLDB da annullare.

Vincoli di lunghezza: lunghezza fissa di 22.

Modello: `^[A-Za-z-0-9]+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "StreamId": "string"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

StreamId

L'UUID (testo con codifica Base62) dello stream di journal QLDB annullato.

Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 22.

Modello: `^[A-Za-z-0-9]+$`

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

ResourcePreconditionNotMetException

L'operazione non è riuscita perché una condizione non era soddisfatta in anticipo.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

CreateLedger

Servizio: Amazon QLDB

Crea un nuovo registro Account AWS nella tua regione corrente.

Sintassi della richiesta

```
POST /ledgers HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "Tags": {
      "string" : "string"
   }
}
```

Parametri della richiesta URI:

La richiesta non utilizza parametri URI.

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

DeletionProtection

Specifica se il libro mastro è protetto dall'eliminazione da parte di un utente qualsiasi. Se non definita al momento della creazione del libro mastro, questa caratteristica è abilitata (`true`) per impostazione predefinita.

Se la protezione dall'eliminazione è abilitata, è necessario innanzitutto disabilitarla prima di poter eliminare il libro mastro. Puoi disabilitarla chiamando l'operazione `UpdateLedger` per impostare questo parametro su `false`.

Tipo: Booleano

Campo obbligatorio: no

KmsKey

La chiave in AWS Key Management Service (AWS KMS) da usare per la crittografia dei dati inattivi nel registro. Per ulteriori informazioni, consulta Crittografia dei dati inattivi nella Guida per gli sviluppatori di Amazon QLDB.

Per specificare questo parametro, puoi utilizzare una delle opzioni seguenti:

- `AWS_OWNED_KMS_KEY`: utilizza una AWS KMS chiave posseduta e gestita da per tuo AWS conto.
- Non definito: per impostazione predefinita, utilizza una chiave KMS AWS di proprietà.
- Una chiave KMS simmetrica valida gestita dal cliente: usa le chiave KMS di crittografia simmetrica specificata nell'account che crei, possiedi e gestisci.

  Amazon QLDB non supporta le chiavi asimmetriche. Per ulteriori informazioni, consulta Uso delle chiavi simmetriche e asimmetriche nella Guida per gli sviluppatori. AWS Key Management Service

Per specificare una chiave KMS gestita dal cliente, utilizza il relativo ID chiave, il nome della risorsa Amazon (ARN), il nome dell'alias o l'ARN dell'alias. Quando utilizzi un nome alias, aggiungi il prefisso `"alias/"`. Per specificare una chiave in un'altra Account AWS, è necessario utilizzare la chiave ARN o l'alias ARN.

Per esempio:

- ID chiave: `1234abcd-12ab-34cd-56ef-1234567890ab`
- ARN chiave: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Nome alias: `alias/ExampleAlias`
- ARN alias: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

Per ulteriori informazioni, consulta Key identifiers (KeyId) nella Developer Guide. AWS Key Management Service

➨Tipo: stringa

Vincoli di lunghezza: lunghezza massima di 1600.

Campo obbligatorio: no

Name

Il nome della contabilità che desideri creare. Il nome deve essere univoco tra tutti i registri della regione Account AWS corrente.

I vincoli di denominazione per i nomi dei libri contabili sono definiti in Quote in Amazon QLDB nella Guida per gli sviluppatori di Amazon QLDB.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

PermissionsMode

Le autorizzazioni da assegnare alla contabilità che si desidera creare. Questo parametro può avere uno dei seguenti valori:

- `ALLOW_ALL`: una modalità di autorizzazione legacy che consente il controllo degli accessi con granularità a livello di API per i libri mastri.

  Questa modalità consente agli utenti che dispongono dell'autorizzazione API `SendCommand` per questo libro mastro per eseguire tutti i comandi PartiQL (quindi, `ALLOW_ALL`) su qualsiasi tabella nel libro mastro specificato. Questa modalità ignora tutte le policy di autorizzazione IAM a livello di tabella o di comando create per il libro mastro.

- `STANDARD`: (impostazione consigliata) una modalità di autorizzazione che consente il controllo degli accessi con una granularità più fine per libri mastri, tabelle e comandi PartiQL.

  Per impostazione predefinita, questa modalità nega tutte le richieste degli utenti di eseguire qualsiasi comando PartiQL su qualsiasi tabella in questo libro mastro. Per consentire l'esecuzione dei comandi PartiQL, devi creare le policy di autorizzazione IAM per risorse di tabelle e operazioni PartiQL specifiche, oltre all'autorizzazione API `SendCommand` per il libro mastro. Per informazioni, consulta Getting started with the standard permissions mode (Nozioni di base sulla modalità di autorizzazione standard) nella Guida per gli sviluppatori di Amazon QLDB.

> **Note**
>
> Ti consigliamo di utilizzare la modalità di autorizzazione STANDARD per incrementare la sicurezza dei dati nel libro mastro.

- Tipo: stringa

Valori validi: `ALLOW_ALL | STANDARD`

Campo obbligatorio: sì

Tags

Le coppie chiave-valore da aggiungere come tag al libro mastro che si desidera creare. Le chiavi dei tag prevedono una distinzione tra lettere maiuscole e minuscole. I valori dei tag fanno distinzione tra maiuscole e minuscole e possono essere nulli.

Tipo: mappatura stringa a stringa

Voci sulla mappa: numero minimo di 0 elementi. Numero massimo di 200 elementi.

Limitazioni di lunghezza della chiave: la lunghezza minima è 1. La lunghezza massima è 128 caratteri.

Limiti di lunghezza del valore: lunghezza minima di 0. La lunghezza massima è 256 caratteri.

Campo obbligatorio: no

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "CreationDateTime": number,
   "DeletionProtection": boolean,
   "KmsKeyArn": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "State": "string"
```

```
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

Arn

L'Amazon Resource Name (ARN) per il registro.

Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

CreationDateTime

La data e l'ora, in formato epoch time, in cui è stato creato il libro mastro. (Il formato dell'ora epoch è il numero di secondi trascorsi dalle 00:00 del 1° gennaio 1970 UTC.)

Tipo: Timestamp

DeletionProtection

Specifica se il libro mastro è protetto dall'eliminazione da parte di un utente qualsiasi. Se non definita al momento della creazione del libro mastro, questa caratteristica è abilitata (`true`) per impostazione predefinita.

Se la protezione dall'eliminazione è abilitata, è necessario innanzitutto disabilitarla prima di poter eliminare il libro mastro. Puoi disabilitarla chiamando l'operazione `UpdateLedger` per impostare questo parametro su `false`.

Tipo: Booleano

KmsKeyArn

L'ARN della chiave KMS gestita dal cliente che il registro utilizza per la crittografia a riposo. Se questo parametro non è definito, il registro utilizza una chiave KMS AWS proprietaria per la crittografia.

Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

Name

Il nome del libro mastro.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

La modalità di autorizzazione del libro mastro che hai creato.

- Tipo: stringa

Valori validi: `ALLOW_ALL | STANDARD`

State

Lo stato attuale del libro mastro.

- Tipo: stringa

Valori validi: `CREATING | ACTIVE | DELETING | DELETED`

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

LimitExceededException

Hai raggiunto il limite del numero massimo di risorse consentite.

Codice di stato HTTP: 400

ResourceAlreadyExistsException

La risorsa specificata esiste già.

Codice di stato HTTP: 409

ResourceInUseException

La risorsa specificata non può essere modificata in questo momento.

Codice di stato HTTP: 409

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

DeleteLedger

Servizio: Amazon QLDB

Elimina un registro e tutto il suo contenuto. Questa operazione è irreversibile.

Se la protezione dall'eliminazione è abilitata, è necessario innanzitutto disabilitarla prima di poter eliminare il libro mastro. Puoi disabilitarla chiamando l'operazione `UpdateLedger` per impostare questo parametro su `false`.

Sintassi della richiesta

```
DELETE /ledgers/name HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del libro contabile che si desidera eliminare.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
```

Elementi di risposta

Se l'operazione riesce, il servizio invia una risposta HTTP 200 con un corpo HTTP vuoto.

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

ResourceInUseException

La risorsa specificata non può essere modificata in questo momento.

Codice di stato HTTP: 409

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

ResourcePreconditionNotMetException

L'operazione non è riuscita perché una condizione non è stata soddisfatta in anticipo.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

DescribeJournalKinesisStream

Servizio: Amazon QLDB

Restituisce informazioni dettagliate su un determinato flusso di journal Amazon QLDB. L'output include Amazon Resource Name (ARN), il nome dello stream, lo stato corrente, l'ora di creazione e i parametri della richiesta di creazione dello stream originale.

Questa azione non restituisce alcun stream di journal scaduto. Per ulteriori informazioni, consulta Scadenza per i flussi di terminali nella Amazon QLDB Developer Guide.

Sintassi della richiesta

```
GET /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del libro mastro.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

streamId

L'UUID (rappresentato nel testo con codifica Base62) del flusso di journal QLDB da descrivere.

Vincoli di lunghezza: lunghezza fissa di 22.

Modello: `^[A-Za-z-0-9]+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
   }
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

Stream

Informazioni sullo stream del journal QLDB restituito da una richiesta. `DescribeJournalS3Export`

Tipo: oggetto JournalKinesisStreamDescription

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

ResourcePreconditionNotMetException

L'operazione non è riuscita perché una condizione non era soddisfatta in anticipo.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

DescribeJournalS3Export

Servizio: Amazon QLDB

Restituisce informazioni su un processo di esportazione del diario, tra cui il nome del libro mastro, l'ID di esportazione, l'ora di creazione, lo stato corrente e i parametri della richiesta di creazione dell'esportazione originale.

Questa azione non restituisce alcun processo di esportazione scaduto. Per ulteriori informazioni, consulta la scadenza dei lavori di esportazione nella Amazon QLDB Developer Guide.

Se il processo di esportazione con quello specificato `ExportId` non esiste, viene generato. `ResourceNotFoundException`

Se il libro mastro con quanto indicato `Name` non esiste, viene generato. `ResourceNotFoundException`

Sintassi della richiesta

```
GET /ledgers/name/journal-s3-exports/exportId HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

exportId

L'UUID (rappresentato nel testo codificato in Base62) del processo di esportazione del diario da descrivere.

Vincoli di lunghezza: lunghezza fissa di 22.

Modello: `^[A-Za-z-0-9]+$`

Campo obbligatorio: sì

name

Il nome del libro mastro.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
   }
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

ExportDescription

Informazioni sul processo di esportazione del diario restituito da una `DescribeJournalS3Export` richiesta.

Tipo: oggetto JournalS3ExportDescription

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

DescribeLedger

Servizio: Amazon QLDB

Restituisce informazioni su un registro, inclusi lo stato, la modalità di autorizzazione, le impostazioni di crittografia a riposo e quando è stato creato.

Sintassi della richiesta

```
GET /ledgers/name HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del libro mastro che si desidera descrivere.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

Arn

L'Amazon Resource Name (ARN) per il registro.

Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

CreationDateTime

La data e l'ora, in formato epoch time, in cui è stato creato il libro mastro. (Il formato dell'ora epoch è il numero di secondi trascorsi dalle 00:00 del 1° gennaio 1970 UTC.)

Tipo: Timestamp

DeletionProtection

Specifica se il libro mastro è protetto dall'eliminazione da parte di un utente qualsiasi. Se non definita al momento della creazione del libro mastro, questa caratteristica è abilitata (`true`) per impostazione predefinita.

Se la protezione dall'eliminazione è abilitata, è necessario innanzitutto disabilitarla prima di poter eliminare il libro mastro. Puoi disabilitarla chiamando l'operazione `UpdateLedger` per impostare questo parametro su `false`.

Tipo: Booleano

EncryptionDescription

Informazioni sulla crittografia dei dati inattivi nel registro. Ciò include lo stato corrente, la AWS KMS chiave e il momento in cui la chiave è diventata inaccessibile (in caso di errore). Se questo parametro non è definito, il registro utilizza una chiave KMS AWS proprietaria per la crittografia.

Tipo: oggetto LedgerEncryptionDescription

Name

Il nome del libro mastro.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

La modalità di autorizzazione del registro.

- Tipo: stringa

Valori validi: `ALLOW_ALL | STANDARD`

State

Lo stato attuale del libro mastro.

- Tipo: stringa

Valori validi: `CREATING | ACTIVE | DELETING | DELETED`

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

ExportJournalToS3

Servizio: Amazon QLDB

Esporta il contenuto del diario entro un intervallo di data e ora da un registro in un bucket Amazon Simple Storage Service (Amazon S3) specificato. Un processo di esportazione del diario può scrivere gli oggetti dati nella rappresentazione testuale o binaria del formato Amazon Ion o nel formato di testo JSON Lines.

Se il registro con il dato `Name` non esiste, viene generato. `ResourceNotFoundException`

Se il registro con quanto indicato `Name` è in `CREATING` stato, viene generato. `ResourcePreconditionNotMetException`

È possibile avviare fino a due richieste di esportazione simultanee delle scritture contabili per ogni libro contabile. Oltre questo limite, vengono inviate le richieste di esportazione delle scritture contabili. `LimitExceededException`

Sintassi della richiesta

```
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"
   }
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del libro mastro.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

ExclusiveEndTime

La data e l'ora di fine esclusive per la gamma di contenuti del diario da esportare.

`ExclusiveEndTime` deve essere in formato data e ora `ISO 8601` e in Universal Coordinated Time (UTC). Ad esempio: `2019-06-13T21:36:34Z`.

`ExclusiveEndTime`Deve essere inferiore o uguale alla data e all'ora UTC correnti.

Tipo: Timestamp

Campo obbligatorio: sì

InclusiveStartTime

La data e l'ora di inizio incluse per l'intervallo di contenuti del diario da esportare.

`InclusiveStartTime` deve essere in formato data e ora `ISO 8601` e in Universal Coordinated Time (UTC). Ad esempio: `2019-06-13T21:36:34Z`.

`InclusiveStartTime`Deve essere prima`ExclusiveEndTime`.

Se fornisci un codice `InclusiveStartTime` che precede il registro`CreationDateTime`, Amazon QLDB lo imposta come predefinito sul libro mastro. `CreationDateTime`

Tipo: Timestamp

Campo obbligatorio: sì

OutputFormat

Il formato di output dei dati del diario esportati. Un processo di esportazione del diario può scrivere gli oggetti dati nella rappresentazione testuale o binaria del formato Amazon Ion o nel formato di testo JSON Lines.

Impostazione predefinita: `ION_TEXT`

Nel formato JSON Lines, ogni blocco di journal in un oggetto dati esportato è un oggetto JSON valido delimitato da una nuova riga. Puoi utilizzare questo formato per integrare direttamente le esportazioni JSON con strumenti di analisi come Amazon Athena AWS Glue e perché questi servizi possono analizzare automaticamente JSON delimitato da nuove righe.

- Tipo: stringa

Valori validi: `ION_BINARY | ION_TEXT | JSON`

Campo obbligatorio: no

RoleArn

L'Amazon Resource Name (ARN) del ruolo IAM che concede le autorizzazioni QLDB per un processo di esportazione di riviste per eseguire le seguenti operazioni:

- Scrivi oggetti nel tuo bucket Amazon S3.
- (Facoltativo) Utilizza la chiave gestita dal cliente AWS Key Management Service (AWS KMS) per la crittografia lato server dei dati esportati.

Per passare un ruolo a QLDB quando si richiede l'esportazione di un journal, è necessario disporre delle autorizzazioni per eseguire `iam:PassRole` l'azione sulla risorsa del ruolo IAM. Questo è necessario per tutte le richieste di esportazione del journal.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

Campo obbligatorio: sì

S3ExportConfiguration

Le impostazioni di configurazione della destinazione del bucket Amazon S3 per la tua richiesta di esportazione.

Tipo: oggetto S3ExportConfiguration

Campo obbligatorio: sì

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "ExportId": "string"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

ExportId

L'UUID (rappresentato nel testo con codifica Base62) che QLDB assegna a ogni processo di esportazione del diario.

Per descrivere la richiesta di esportazione e verificare lo stato del lavoro, è possibile utilizzare to call. `ExportId` `DescribeJournalS3Export`

Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 22.

Modello: `^[A-Za-z-0-9]+$`

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

ResourcePreconditionNotMetException

L'operazione non è riuscita perché una condizione non era stata soddisfatta in anticipo.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetBlock

Servizio: Amazon QLDB

Restituisce un oggetto blocco a un indirizzo specificato in un diario. Restituisce inoltre una prova del blocco specificato per la verifica, se `DigestTipAddress` fornita.

Per informazioni sul contenuto dei dati in un blocco, consulta i contenuti del Journal nella Amazon QLDB Developer Guide.

Se il registro specificato non esiste o è in `DELETING` stato, viene generato. `ResourceNotFoundException`

Se il registro specificato è in `CREATING` stato, viene generato. `ResourcePreconditionNotMetException`

Se non esiste alcun blocco con l'indirizzo specificato, viene generato. `InvalidParameterException`

Sintassi della richiesta

```
POST /ledgers/name/block HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del libro mastro.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

BlockAddress

La posizione del blocco che desideri richiedere. Un indirizzo è una struttura Amazon Ion con due campi: `strandId` e `sequenceNo`.

Ad esempio: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`.

Tipo: oggetto ValueHolder

Campo obbligatorio: sì

DigestTipAddress

L'ultima ubicazione del blocco coperta dal digest per la quale richiedere una prova. Un indirizzo è una struttura Amazon Ion con due campi: `strandId` e `sequenceNo`.

Ad esempio: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:49}`.

Tipo: oggetto ValueHolder

Campo obbligatorio: no

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "Block": {
      "IonText": "string"
   },
   "Proof": {
      "IonText": "string"
   }
```

```
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

Block

L'oggetto di dati a blocchi in formato Amazon Ion.

Tipo: oggetto ValueHolder

Proof

L'oggetto di prova in formato Amazon Ion restituito da una `GetBlock` richiesta. Una dimostrazione contiene l'elenco dei valori hash necessari per ricalcolare il digest specificato utilizzando un albero Merkle, a partire dal blocco specificato.

Tipo: oggetto ValueHolder

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

ResourcePreconditionNotMetException

L'operazione non è riuscita perché una condizione non era soddisfatta in anticipo.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetDigest

Servizio: Amazon QLDB

Restituisce il riassunto di un libro mastro nell'ultimo blocco commesso nel diario. La risposta include un valore hash a 256 bit e un indirizzo di blocco.

Sintassi della richiesta

```
POST /ledgers/name/digest HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del libro mastro.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "Digest": blob,
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

Digest

Il valore hash a 256 bit che rappresenta il digest restituito da una richiesta. `GetDigest`

Tipo: oggetto dati binari con codifica Base64

Vincoli di lunghezza: lunghezza fissa di 32.

DigestTipAddress

L'ultima posizione del blocco coperta dal digest che hai richiesto. Un indirizzo è una struttura Amazon Ion con due campi: `strandId` e `sequenceNo`.

Tipo: oggetto ValueHolder

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

ResourcePreconditionNotMetException

L'operazione non è riuscita perché una condizione non era soddisfatta in anticipo.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetRevision

Servizio: Amazon QLDB

Restituisce un oggetto dati di revisione per un ID documento e un indirizzo di blocco specificati. Restituisce inoltre una prova della revisione specificata per la verifica, se `DigestTipAddress` fornita.

Sintassi della richiesta

```
POST /ledgers/name/revision HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   },
   "DocumentId": "string"
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del libro mastro.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

BlockAddress

La posizione del blocco della revisione del documento da verificare. Un indirizzo è una struttura Amazon Ion con due campi: `strandId` e `sequenceNo`.

Ad esempio: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`.

Tipo: oggetto ValueHolder

Campo obbligatorio: sì

DigestTipAddress

L'ultima ubicazione del blocco coperta dal digest per la quale richiedere una prova. Un indirizzo è una struttura Amazon Ion con due campi: `strandId` e `sequenceNo`.

Ad esempio: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:49}`.

Tipo: oggetto ValueHolder

Campo obbligatorio: no

DocumentId

L'UUID (rappresentato nel testo con codifica Base62) del documento da verificare.

Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 22.

Modello: `^[A-Za-z-0-9]+$`

Campo obbligatorio: sì

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "Proof": {
      "IonText": "string"
   },
   "Revision": {
```

```
      "IonText": "string"
   }
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

Proof

L'oggetto di prova in formato Amazon Ion restituito da una `GetRevision` richiesta. Una bozza contiene l'elenco dei valori hash necessari per ricalcolare il digest specificato utilizzando un albero Merkle, a partire dalla revisione del documento specificata.

Tipo: oggetto ValueHolder

Revision

L'oggetto dei dati di revisione del documento in formato Amazon Ion.

Tipo: oggetto ValueHolder

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

ResourcePreconditionNotMetException

L'operazione non è riuscita perché una condizione non era soddisfatta in anticipo.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

ListJournalKinesisStreamsForLedger

Servizio: Amazon QLDB

Restituisce tutti i flussi di journal Amazon QLDB per un determinato libro mastro.

Questa azione non restituisce alcun flusso di journal scaduto. Per ulteriori informazioni, consulta Scadenza per i flussi di terminali nella Amazon QLDB Developer Guide.

Questa azione restituisce un massimo di articoli. `MaxResults` È impaginato in modo da poter recuperare tutti gli elementi `ListJournalKinesisStreamsForLedger` chiamando più volte.

Sintassi della richiesta

```
GET /ledgers/name/journal-kinesis-streams?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del libro mastro.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

MaxResults

Il numero massimo di risultati da restituire in una singola richiesta. `ListJournalKinesisStreamsForLedger` (Il numero effettivo di risultati restituiti potrebbe essere inferiore.)

Intervallo valido: valore minimo di 1. valore massimo pari a 100.

NextToken

Un token di impaginazione, che indica che si desidera recuperare la pagina successiva di risultati. Se hai ricevuto un valore per `NextToken` nella risposta di una

`ListJournalKinesisStreamsForLedger` chiamata precedente, dovresti usare quel valore come input qui.

Vincoli di lunghezza: lunghezza minima di 4. La lunghezza massima è 1024 caratteri.

Modello: `^[A-Za-z-0-9+/=]+$`

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
      }
   ]
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

NextToken

- Se `NextToken` è vuoto, l'ultima pagina dei risultati è stata elaborata e non ci sono altri risultati da recuperare.
- Se non *`NextToken`* è vuoto, sono disponibili altri risultati. Per recuperare la pagina successiva di risultati, utilizzare il valore di `NextToken` in una `ListJournalKinesisStreamsForLedger` chiamata successiva.

Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 4. La lunghezza massima è 1024 caratteri.

Modello: `^[A-Za-z-0-9+/=]+$`

Streams

I flussi di journal QLDB attualmente associati al registro specificato.

Tipo: matrice di oggetti JournalKinesisStreamDescription

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

ResourcePreconditionNotMetException

L'operazione non è riuscita perché una condizione non era soddisfatta in anticipo.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

ListJournalS3Exports

Servizio: Amazon QLDB

Restituisce tutti i lavori di esportazione delle scritture contabili per tutti i libri contabili associati alla regione corrente Account AWS e alla regione.

Questa azione restituisce un massimo di `MaxResults` elementi ed è suddivisa in pagine in modo da poter recuperare tutti gli elementi chiamando più volte. `ListJournalS3Exports`

Questa azione non restituisce alcun processo di esportazione scaduto. Per ulteriori informazioni, consulta la scadenza dei lavori di esportazione nella Amazon QLDB Developer Guide.

Sintassi della richiesta

```
GET /journal-s3-exports?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

MaxResults

Il numero massimo di risultati da restituire in una singola `ListJournalS3Exports` richiesta. (Il numero effettivo di risultati restituiti potrebbe essere inferiore.)

Intervallo valido: valore minimo di 1. valore massimo pari a 100.

NextToken

Un token di impaginazione, che indica che si desidera recuperare la pagina successiva di risultati. Se hai ricevuto un valore per `NextToken` nella risposta di una `ListJournalS3Exports` chiamata precedente, allora dovresti usare quel valore come input qui.

Vincoli di lunghezza: lunghezza minima di 4. La lunghezza massima è 1024 caratteri.

Modello: `^[A-Za-z-0-9+/=]+$`

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

JournalS3Exports

I lavori di esportazione delle scritture contabili per tutti i libri contabili associati alla regione corrente Account AWS e alla regione.

Tipo: matrice di oggetti JournalS3ExportDescription

NextToken

- Se `NextToken` è vuoto, significa che l'ultima pagina dei risultati è stata elaborata e non ci sono altri risultati da recuperare.
- Se non *`NextToken`* è vuoto, ci sono altri risultati disponibili. Per recuperare la pagina successiva di risultati, usa il valore di `NextToken` in una `ListJournalS3Exports` chiamata successiva.

Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 4. La lunghezza massima è 1024 caratteri.

Modello: `^[A-Za-z-0-9+/=]+$`

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

ListJournalS3ExportsForLedger

Servizio: Amazon QLDB

Restituisce tutti i lavori di esportazione delle scritture contabili per un registro specificato.

Questa azione restituisce un massimo di `MaxResults` elementi ed è suddivisa in pagine in modo da poter recuperare tutti gli elementi chiamando più volte. `ListJournalS3ExportsForLedger`

Questa azione non restituisce alcun processo di esportazione scaduto. Per ulteriori informazioni, consulta la scadenza dei lavori di esportazione nella Amazon QLDB Developer Guide.

Sintassi della richiesta

```
GET /ledgers/name/journal-s3-exports?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

MaxResults

Il numero massimo di risultati da restituire in una singola `ListJournalS3ExportsForLedger` richiesta. (Il numero effettivo di risultati restituiti potrebbe essere inferiore.)

Intervallo valido: valore minimo di 1. valore massimo pari a 100.

name

Il nome del libro mastro.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

NextToken

Un token di impaginazione, che indica che si desidera recuperare la pagina successiva di risultati. Se hai ricevuto un valore per `NextToken` nella risposta di una `ListJournalS3ExportsForLedger` chiamata precedente, allora dovresti usare quel valore come input qui.

Vincoli di lunghezza: lunghezza minima di 4. La lunghezza massima è 1024 caratteri.

Modello: `^[A-Za-z-0-9+/=]+$`

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

JournalS3Exports

I lavori di esportazione delle scritture contabili attualmente associati al libro mastro specificato.

Tipo: matrice di oggetti JournalS3ExportDescription

NextToken

- Se `NextToken` è vuoto, significa che l'ultima pagina dei risultati è stata elaborata e non ci sono altri risultati da recuperare.
- Se non *`NextToken`* è vuoto, ci sono altri risultati disponibili. Per recuperare la pagina successiva di risultati, usa il valore di `NextToken` in una `ListJournalS3ExportsForLedger` chiamata successiva.

Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 4. La lunghezza massima è 1024 caratteri.

Modello: `^[A-Za-z-0-9+/=]+$`

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

ListLedgers

Servizio: Amazon QLDB

Restituisce tutti i libri contabili associati alla corrente Account AWS e alla regione.

Questa azione restituisce un massimo di `MaxResults` elementi ed è suddivisa in pagine in modo da poter recuperare tutti gli elementi chiamando più volte. `ListLedgers`

Sintassi della richiesta

```
GET /ledgers?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

MaxResults

Il numero massimo di risultati da restituire in una singola richiesta. `ListLedgers` (Il numero effettivo di risultati restituiti potrebbe essere inferiore.)

Intervallo valido: valore minimo di 1. valore massimo pari a 100.

NextToken

Un token di impaginazione, che indica che si desidera recuperare la pagina successiva di risultati. Se hai ricevuto un valore per `NextToken` nella risposta di una `ListLedgers` chiamata precedente, allora dovresti usare quel valore come input qui.

Vincoli di lunghezza: lunghezza minima di 4. La lunghezza massima è 1024 caratteri.

Modello: `^[A-Za-z-0-9+/=]+$`

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "Ledgers": [
      {
         "CreationDateTime": number,
         "Name": "string",
         "State": "string"
      }
   ],
   "NextToken": "string"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

Ledgers

I libri contabili associati alla corrente Account AWS e alla regione.

Tipo: matrice di oggetti LedgerSummary

NextToken

Un token di impaginazione, che indica se ci sono più risultati disponibili:

- Se `NextToken` è vuoto, significa che l'ultima pagina dei risultati è stata elaborata e non ci sono altri risultati da recuperare.
- Se non *`NextToken`* è vuoto, ci sono altri risultati disponibili. Per recuperare la pagina successiva di risultati, usa il valore di `NextToken` in una `ListLedgers` chiamata successiva.

Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 4. La lunghezza massima è 1024 caratteri.

Modello: `^[A-Za-z-0-9+/=]+$`

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

ListTagsForResource

Servizio: Amazon QLDB

Restituisce tutti i tag per una risorsa Amazon QLDB specificata.

Sintassi della richiesta

```
GET /tags/resourceArn HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

resourceArn

L'Amazon Resource Name (ARN) per cui elencare i tag. Per esempio:

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

Tags

I tag attualmente associati alla risorsa Amazon QLDB specificata.

Tipo: mappatura stringa a stringa

Voci sulla mappa: numero minimo di 0 elementi. Numero massimo di 200 elementi.

Limitazioni di lunghezza della chiave: la lunghezza minima è 1. La lunghezza massima è 128 caratteri.

Limiti di lunghezza del valore: lunghezza minima di 0. La lunghezza massima è 256 caratteri.

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2

- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

StreamJournalToKinesis

Servizio: Amazon QLDB

Crea uno stream di journal per un determinato registro Amazon QLDB. Il flusso acquisisce ogni revisione del documento di cui viene eseguito il commit al journal del libro mastro e invia i dati a una specifica risorsa Amazon Kinesis Data Streams.

Sintassi della richiesta

```
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"
   }
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del libro mastro.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

ExclusiveEndTime

Data e ora esclusive che specificano quando termina il flusso. Se non si definisce questo parametro, il flusso viene eseguito a tempo indeterminato fino a quando non lo si annulla.

`ExclusiveEndTime` deve essere in formato data e ora `ISO 8601` e in Universal Coordinated Time (UTC). Ad esempio: `2019-06-13T21:36:34Z`.

Tipo: Timestamp

Campo obbligatorio: no

InclusiveStartTime

Data e ora di inizio inclusive da cui iniziare lo streaming dei dati del journal. Questo parametro deve essere in formato data e ora `ISO 8601` e in Universal Coordinated Time (UTC). Ad esempio: `2019-06-13T21:36:34Z`.

`InclusiveStartTime` non può essere in futuro e deve essere precedente a `ExclusiveEndTime`.

Se si fornisce un `InclusiveStartTime` che è precedente a `CreationDateTime` del libro mastro, QLDB viene effettivamente impostato in modo predefinito su `CreationDateTime` del libro mastro.

Tipo: Timestamp

Campo obbligatorio: sì

KinesisConfiguration

Le impostazioni di configurazione della destinazione Kinesis Data Streams per la richiesta di flusso.

Tipo: oggetto KinesisConfiguration

Campo obbligatorio: sì

RoleArn

L'Amazon Resource Name (ARN) del ruolo IAM che concede autorizzazioni QLDB per un flusso di journal per scrivere record di dati in una risorsa Kinesis Data Streams.

Per passare un ruolo a QLDB quando si richiede un flusso journal, è necessario disporre delle autorizzazioni per eseguire l'operazione `iam:PassRole` sulla risorsa del ruolo IAM. Questa operazione è necessaria per tutte le richieste di flusso journal.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

Campo obbligatorio: sì

StreamName

Il nome che si desidera assegnare al flusso del journal QLDB. I nomi definiti dall'utente possono aiutare a identificare e indicare lo scopo di un flusso.

Il nome del flusso deve essere univoco tra gli altri flussi attivi per un determinato libro mastro. I nomi dei flussi hanno gli stessi vincoli per la denominazione dei nomi dei libri mastri, come definito nell'argomento relativo alle quote in Amazon QLDB in Amazon QLDB Developer Guide.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

Tags

Le coppie chiave-valore da aggiungere come tag allo stream che desideri creare. Le chiavi dei tag prevedono una distinzione tra lettere maiuscole e minuscole. I valori dei tag fanno distinzione tra maiuscole e minuscole e possono essere nulli.

Tipo: mappatura stringa a stringa

Voci sulla mappa: numero minimo di 0 elementi. Numero massimo di 200 elementi.

Limitazioni di lunghezza della chiave: la lunghezza minima è 1. La lunghezza massima è 128 caratteri.

Limiti di lunghezza del valore: lunghezza minima di 0. La lunghezza massima è 256 caratteri.

Campo obbligatorio: no

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "StreamId": "string"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

StreamId

L'UUID (rappresentato nel testo con codifica Base62) che QLDB assegna a ogni flusso di journal QLDB.

- Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 22.

Modello: `^[A-Za-z-0-9]+$`

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

ResourcePreconditionNotMetException

L'operazione non è riuscita perché una condizione non era soddisfatta in anticipo.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

TagResource

Servizio: Amazon QLDB

Aggiunge uno o più tag a una risorsa Amazon QLDB specificata.

Una risorsa può avere fino a 50 tag. Se provi a creare più di 50 tag per una risorsa, la richiesta ha esito negativo e restituisce un errore.

Sintassi della richiesta

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

resourceArn

L'Amazon Resource Name (ARN) a cui desideri aggiungere i tag. Per esempio:

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

Tags

Le coppie chiave-valore da aggiungere come tag alla risorsa QLDB specificata. Le chiavi dei tag prevedono una distinzione tra lettere maiuscole e minuscole. Se si specifica una chiave già

esistente per la risorsa, la richiesta ha esito negativo e restituisce un errore. I valori dei tag fanno distinzione tra maiuscole e minuscole e possono essere nulli.

Tipo: mappatura stringa a stringa

Voci sulla mappa: numero minimo di 0 elementi. Numero massimo di 200 elementi.

Limitazioni di lunghezza della chiave: la lunghezza minima è 1. La lunghezza massima è 128 caratteri.

Limiti di lunghezza del valore: lunghezza minima di 0. La lunghezza massima è 256 caratteri.

Campo obbligatorio: sì

Sintassi della risposta

```
HTTP/1.1 200
```

Elementi di risposta

Se l'operazione riesce, il servizio invia una risposta HTTP 200 con un corpo HTTP vuoto.

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

UntagResource

Servizio: Amazon QLDB

Rimuove uno o più tag da una risorsa Amazon QLDB specificata. Puoi specificare fino a 50 chiavi di tag da rimuovere.

Sintassi della richiesta

```
DELETE /tags/resourceArn?tagKeys=TagKeys HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

resourceArn

L'Amazon Resource Name (ARN) da cui rimuovere i tag. Per esempio:

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

Campo obbligatorio: sì

TagKeys

L'elenco delle chiavi dei tag da rimuovere.

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 200 elementi.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 128 caratteri.

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
```

Elementi di risposta

Se l'operazione riesce, il servizio invia una risposta HTTP 200 con un corpo HTTP vuoto.

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

UpdateLedger

Servizio: Amazon QLDB

Aggiorna le proprietà su un libro mastro.

Sintassi della richiesta

```
PATCH /ledgers/name HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string"
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del libro mastro.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

DeletionProtection

Specifica se il libro mastro è protetto dall'eliminazione da parte di un utente qualsiasi. Se non definita al momento della creazione del libro mastro, questa caratteristica è abilitata (`true`) per impostazione predefinita.

Se la protezione dall'eliminazione è abilitata, è necessario innanzitutto disabilitarla prima di poter eliminare il libro mastro. Puoi disabilitarla chiamando l'operazione `UpdateLedger` per impostare questo parametro su `false`.

Tipo: Booleano

Campo obbligatorio: no

KmsKey

La chiave in AWS Key Management Service (AWS KMS) da utilizzare per la crittografia dei dati inattivi nel registro. Per ulteriori informazioni, consulta Crittografia dei dati inattivi nella Guida per gli sviluppatori di Amazon QLDB.

Per specificare questo parametro, puoi utilizzare una delle opzioni seguenti:

- `AWS_OWNED_KMS_KEY`: utilizza una AWS KMS chiave posseduta e gestita da per tuo AWS conto.
- Non definito: non apporta modifiche alla chiave KMS del registro.
- Una chiave KMS simmetrica valida gestita dal cliente: usa le chiave KMS di crittografia simmetrica specificata nell'account che crei, possiedi e gestisci.

  Amazon QLDB non supporta le chiavi asimmetriche. Per ulteriori informazioni, consulta Uso delle chiavi simmetriche e asimmetriche nella Guida per gli sviluppatori. AWS Key Management Service

Per specificare una chiave KMS gestita dal cliente, utilizza il relativo ID chiave, il nome della risorsa Amazon (ARN), il nome dell'alias o l'ARN dell'alias. Quando utilizzi un nome alias, aggiungi il prefisso `"alias/"`. Per specificare una chiave in un'altra Account AWS, è necessario utilizzare la chiave ARN o l'alias ARN.

Per esempio:

- ID chiave: `1234abcd-12ab-34cd-56ef-1234567890ab`
- ARN chiave: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Nome alias: `alias/ExampleAlias`
- ARN alias: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

Per ulteriori informazioni, consulta Key identifiers (KeyId) nella Developer Guide. AWS Key Management Service

➨Tipo: stringa

Vincoli di lunghezza: lunghezza massima di 1600.

Campo obbligatorio: no

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

Arn

L'Amazon Resource Name (ARN) per il registro.

Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

CreationDateTime

La data e l'ora, in formato epoch time, in cui è stato creato il libro mastro. (Il formato dell'ora epoch è il numero di secondi trascorsi dalle 00:00 del 1° gennaio 1970 UTC.)

Tipo: Timestamp

DeletionProtection

Specifica se il libro mastro è protetto dall'eliminazione da parte di un utente qualsiasi. Se non definita al momento della creazione del libro mastro, questa caratteristica è abilitata (`true`) per impostazione predefinita.

Se la protezione dall'eliminazione è abilitata, è necessario innanzitutto disabilitarla prima di poter eliminare il libro mastro. Puoi disabilitarla chiamando l'operazione `UpdateLedger` per impostare questo parametro su `false`.

Tipo: Booleano

EncryptionDescription

Informazioni sulla crittografia dei dati inattivi nel registro. Ciò include lo stato corrente, la AWS KMS chiave e il momento in cui la chiave è diventata inaccessibile (in caso di errore).

Tipo: oggetto LedgerEncryptionDescription

Name

Il nome del libro mastro.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

State

Lo stato attuale del libro mastro.

- Tipo: stringa

Valori validi: `CREATING | ACTIVE | DELETING | DELETED`

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

UpdateLedgerPermissionsMode

Servizio: Amazon QLDB

Aggiorna la modalità di autorizzazione di un libro mastro.

> **Important**
>
> Prima di passare alla modalità STANDARD autorizzazioni, devi prima creare tutte le politiche IAM e i tag di tabella richiesti per evitare interruzioni per gli utenti. Per ulteriori informazioni, consulta la sezione Migrazione alla modalità di autorizzazione standard nella Amazon QLDB Developer Guide.

Sintassi della richiesta

```
PATCH /ledgers/name/permissions-mode HTTP/1.1
Content-type: application/json

{
   "PermissionsMode": "string"
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del libro mastro.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

PermissionsMode

La modalità di autorizzazione da assegnare al registro. Questo parametro può avere uno dei seguenti valori:

- `ALLOW_ALL`: una modalità di autorizzazione legacy che consente il controllo degli accessi con granularità a livello di API per i libri mastri.

  Questa modalità consente agli utenti che dispongono dell'autorizzazione API `SendCommand` per questo libro mastro per eseguire tutti i comandi PartiQL (quindi, `ALLOW_ALL`) su qualsiasi tabella nel libro mastro specificato. Questa modalità ignora tutte le policy di autorizzazione IAM a livello di tabella o di comando create per il libro mastro.

- `STANDARD`: (impostazione consigliata) una modalità di autorizzazione che consente il controllo degli accessi con una granularità più fine per libri mastri, tabelle e comandi PartiQL.

  Per impostazione predefinita, questa modalità nega tutte le richieste degli utenti di eseguire qualsiasi comando PartiQL su qualsiasi tabella in questo libro mastro. Per consentire l'esecuzione dei comandi PartiQL, devi creare le policy di autorizzazione IAM per risorse di tabelle e operazioni PartiQL specifiche, oltre all'autorizzazione API `SendCommand` per il libro mastro. Per informazioni, consulta Getting started with the standard permissions mode (Nozioni di base sulla modalità di autorizzazione standard) nella Guida per gli sviluppatori di Amazon QLDB.

> **Note**
>
> Ti consigliamo di utilizzare la modalità di autorizzazione `STANDARD` per incrementare la sicurezza dei dati nel libro mastro.

Tipo: stringa

Valori validi: `ALLOW_ALL | STANDARD`

Campo obbligatorio: sì

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "Arn": "string",
   "Name": "string",
   "PermissionsMode": "string"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

Arn

L'Amazon Resource Name (ARN) per il registro.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

Name

Il nome del libro mastro.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

L'attuale modalità di autorizzazione del registro.

- Tipo: stringa

Valori validi: `ALLOW_ALL | STANDARD`

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

InvalidParameterException

Uno o più parametri nella richiesta non sono validi.

Codice di stato HTTP: 400

ResourceNotFoundException

La risorsa specificata non esiste.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

Sessione Amazon QLDB

Le seguenti operazioni sono supportate da Amazon QLDB Session:

- SendCommand

SendCommand

Servizio: Amazon QLDB Session

Invia un comando a un registro Amazon QLDB.

> **Note**
>
> Invece di interagire direttamente con questa API, consigliamo di utilizzare il driver QLDB o la shell QLDB per eseguire transazioni di dati su un registro.
>
> - Se lavori con un AWS SDK, usa il driver QLDB. Il driver fornisce un livello di astrazione di alto livello sopra questa API di dati di sessione QLDB e gestisce l'operazione per te. `SendCommand` Per informazioni e un elenco dei linguaggi di programmazione supportati, consulta la sezione Guida introduttiva al driver nella Amazon QLDB Developer Guide.
> - Se stai lavorando con AWS Command Line Interface (AWS CLI), usa la shell QLDB. La shell è un'interfaccia a riga di comando che utilizza il driver QLDB per interagire con un registro. Per informazioni, consulta Accedere ad Amazon QLDB usando la shell QLDB.

Sintassi della richiesta

```
{
   "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": {
   }
}
```

Parametri della richiesta

Per informazioni sui parametri comuni per tutte le azioni, consulta Parametri comuni.

La richiesta accetta i seguenti dati in formato JSON.

AbortTransaction

Comando per interrompere la transazione corrente.

Tipo: oggetto AbortTransactionRequest

Campo obbligatorio: no

CommitTransaction

Comando per confermare la transazione specificata.

Tipo: oggetto CommitTransactionRequest

Campo obbligatorio: no

EndSession

Comando per terminare la sessione corrente.

Tipo: oggetto EndSessionRequest

Campo obbligatorio: no

ExecuteStatement

Comando per eseguire un'istruzione nella transazione specificata.

Tipo: oggetto ExecuteStatementRequest

Campo obbligatorio: no

FetchPage

Comando per recuperare una pagina.

Tipo: oggetto FetchPageRequest

Campo obbligatorio: no

SessionToken

Specifica il token di sessione per il comando corrente. Un token di sessione è costante per tutta la durata della sessione.

Per ottenere un token di sessione, esegui il `StartSession` comando. Questo `SessionToken` è necessario per ogni comando successivo che viene emesso durante la sessione corrente.

- Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 4. La lunghezza massima è 1024 caratteri.

Modello: `^[A-Za-z-0-9+/=]+$`

Campo obbligatorio: no

StartSession

Comando per iniziare una nuova sessione. Un token di sessione viene ottenuto come parte della risposta.

Tipo: oggetto StartSessionRequest

Campo obbligatorio: no

StartTransaction

Comando per iniziare una nuova transazione.

Tipo: oggetto StartTransactionRequest

Campo obbligatorio: no

Sintassi della risposta

```
{
   "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"
      }
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

AbortTransaction

Contiene i dettagli della transazione interrotta.

Tipo: oggetto AbortTransactionResult

CommitTransaction

Contiene i dettagli della transazione confermata.

Tipo: oggetto CommitTransactionResult

EndSession

Contiene i dettagli della sessione terminata.

Tipo: oggetto EndSessionResult

ExecuteStatement

Contiene i dettagli dell'istruzione eseguita.

Tipo: oggetto ExecuteStatementResult

FetchPage

Contiene i dettagli della pagina recuperata.

Tipo: oggetto FetchPageResult

StartSession

Contiene i dettagli della sessione avviata che include un token di sessione. Questo `SessionToken` è necessario per ogni comando successivo che viene emesso durante la sessione corrente.

Tipo: oggetto StartSessionResult

StartTransaction

Contiene i dettagli della transazione avviata.

Tipo: oggetto StartTransactionResult

Errori

Per informazioni sugli errori comuni a tutte le operazioni, consultare Errori comuni.

BadRequestException

Restituito se la richiesta non è valida o contiene un errore, ad esempio un valore di parametro non valido o un parametro obbligatorio mancante.

Codice di stato HTTP: 400

CapacityExceededException

Restituito quando la richiesta supera la capacità di elaborazione del libro mastro.

Codice di stato HTTP: 400

InvalidSessionException

Restituito se la sessione non esiste più perché è scaduta o è scaduta.

Codice di stato HTTP: 400

LimitExceededException

Restituito se viene superato un limite di risorse, ad esempio il numero di sessioni attive.

Codice di stato HTTP: 400

OccConflictException

Restituito quando una transazione non può essere scritta nel journal a causa di un errore nella fase di verifica dell'Optimistic Concurrency Control (OCC).

Codice di stato HTTP: 400

RateExceededException

Restituito quando la frequenza delle richieste supera il throughput consentito.

Codice di stato HTTP: 400

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3

- AWS SDK per Python
- AWS SDK per Ruby V3

Tipi di dati

I tipi di dati seguenti sono supportati da Amazon QLDB:

- JournalKinesisStreamDescription
- JournalS3ExportDescription
- KinesisConfiguration
- LedgerEncryptionDescription
- LedgerSummary
- S3EncryptionConfiguration
- S3ExportConfiguration
- ValueHolder

I tipi di dati seguenti sono supportati da 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

Servizio: Amazon QLDB

Informazioni su uno stream di journal Amazon QLDB, tra cui Amazon Resource Name (ARN), nome dello stream, ora di creazione, stato corrente e parametri della richiesta di creazione dello stream originale.

Indice

KinesisConfiguration

Le impostazioni di configurazione della destinazione Amazon Kinesis Data Streams per un flusso di journal QLDB.

Tipo: oggetto KinesisConfiguration

Campo obbligatorio: sì

LedgerName

Il nome del libro mastro.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

RoleArn

L'Amazon Resource Name (ARN) del ruolo IAM che concede autorizzazioni QLDB per un flusso di journal per scrivere record di dati in una risorsa Kinesis Data Streams.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

Campo obbligatorio: sì

Status

Lo stato attuale dello stream del journal QLDB.

- Tipo: stringa

  Valori validi: `ACTIVE | COMPLETED | CANCELED | FAILED | IMPAIRED`

  Campo obbligatorio: sì

StreamId

L'UUID (rappresentato nel testo con codifica Base62) del flusso di journal QLDB.

- Tipo: stringa

  Vincoli di lunghezza: lunghezza fissa di 22.

  Modello: `^[A-Za-z-0-9]+$`

  Campo obbligatorio: sì

StreamName

Il nome definito dall'utente del flusso di journal QLDB.

- Tipo: stringa

  Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

  Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

  Campo obbligatorio: sì

Arn

L'Amazon Resource Name (ARN) del flusso del journal QLDB.

- Tipo: stringa

  Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

  Campo obbligatorio: no

CreationTime

La data e l'ora, in formato epoch time, in cui è stato creato lo stream del journal QLDB. (Il formato dell'ora Epoch è il numero di secondi trascorsi dalle 00:00 del 1° gennaio 1970 UTC.)

Tipo: Timestamp

Campo obbligatorio: no

ErrorCause

Il messaggio di errore che descrive il motivo per cui uno stream ha lo stato o. `IMPAIRED` `FAILED` Questo non è applicabile agli stream con altri valori di stato.

-Tipo: stringa

Valori validi: `KINESIS_STREAM_NOT_FOUND | IAM_PERMISSION_REVOKED`

Campo obbligatorio: no

ExclusiveEndTime

Data e ora esclusive che specificano quando termina il flusso. Se questo parametro non è definito, lo stream viene eseguito all'infinito finché non viene annullato.

Tipo: Timestamp

Campo obbligatorio: no

InclusiveStartTime

Data e ora di inizio inclusive da cui iniziare lo streaming dei dati del journal.

Tipo: Timestamp

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

JournalS3ExportDescription

Servizio: Amazon QLDB

Informazioni su un processo di esportazione di riviste, tra cui il nome del libro mastro, l'ID di esportazione, l'ora di creazione, lo stato corrente e i parametri della richiesta di creazione dell'esportazione originale.

Indice

ExclusiveEndTime

La data e l'ora di fine esclusive per la gamma di contenuti del diario specificata nella richiesta di esportazione originale.

Tipo: Timestamp

Campo obbligatorio: sì

ExportCreationTime

La data e l'ora, in formato epoch time, in cui è stato creato il processo di esportazione. (Il formato dell'ora Epoch è il numero di secondi trascorsi dalle 00:00 del 1° gennaio 1970 UTC.)

Tipo: Timestamp

Campo obbligatorio: sì

ExportId

L'UUID (rappresentato nel testo codificato in Base62) del processo di esportazione della rivista.

-Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 22.

Modello: `^[A-Za-z-0-9]+$`

Campo obbligatorio: sì

InclusiveStartTime

La data e l'ora di inizio incluse per l'intervallo di contenuti del diario specificato nella richiesta di esportazione originale.

Tipo: Timestamp

Campo obbligatorio: sì

LedgerName

Il nome del libro mastro.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

RoleArn

L'Amazon Resource Name (ARN) del ruolo IAM che concede le autorizzazioni QLDB per un processo di esportazione di riviste per eseguire le seguenti operazioni:

- Scrivi oggetti nel tuo bucket Amazon Simple Storage Service (Amazon S3).
- (Facoltativo) Utilizza la chiave gestita dal cliente AWS Key Management Service (AWS KMS) per la crittografia lato server dei dati esportati.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

Campo obbligatorio: sì

S3ExportConfiguration

La posizione del bucket Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) in cui un processo di esportazione del diario scrive il contenuto del diario.

Tipo: oggetto S3ExportConfiguration

Campo obbligatorio: sì

Status

Lo stato attuale del processo di esportazione del diario.

- Tipo: stringa

Valori validi: `IN_PROGRESS | COMPLETED | CANCELLED`

Campo obbligatorio: sì

OutputFormat

Il formato di output dei dati del diario esportati.

Tipo: stringa

Valori validi: `ION_BINARY | ION_TEXT | JSON`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

KinesisConfiguration

Servizio: Amazon QLDB

Le impostazioni di configurazione della destinazione Amazon Kinesis Data Streams per un flusso del journal di Amazon QLDB.

Indice

StreamArn

Prendi nota dell'Amazon Resource Name (ARN) della risorsa Kinesis Data Streams.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

Campo obbligatorio: sì

AggregationEnabled

Consente a QLDB di pubblicare più record di dati in un singolo record Kinesis Data Streams, aumentando il numero di record inviati per chiamata API.

Impostazione predefinita: `True`

> **Important**
>
> L'aggregazione dei record ha importanti implicazioni per l'elaborazione dei record e richiede la disaggregazione nel consumer di flusso. Per ulteriori informazioni, consulta Concetti chiave KPL e Disaggregazione del consumatore nella Guida per sviluppatori di Amazon Kinesis Data Streams.

Tipo: Booleano

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

LedgerEncryptionDescription

Servizio: Amazon QLDB

Informazioni sulla crittografia dei dati inattivi in un registro Amazon QLDB. Ciò include lo stato corrente, la chiave in AWS Key Management Service (AWS KMS) e quando la chiave è diventata inaccessibile (in caso di errore).

Per ulteriori informazioni, consulta Crittografia dei dati inattivi nella Guida per gli sviluppatori di Amazon QLDB.

Indice

EncryptionStatus

Lo stato attuale della crittografia a riposo per il registro. Può essere uno dei seguenti valori:

- `ENABLED`: La crittografia è completamente abilitata utilizzando la chiave specificata.
- `UPDATING`: Il registro elabora attivamente la modifica della chiave specificata.

  Le modifiche principali in QLDB sono asincrone. Il registro è completamente accessibile senza alcun impatto sulle prestazioni durante l'elaborazione della modifica chiave. Il tempo necessario per aggiornare una chiave varia a seconda delle dimensioni del registro.
- `KMS_KEY_INACCESSIBLE`: la chiave KMS gestita dal cliente specificata non è accessibile e il registro è danneggiato. La chiave è stata disabilitata o eliminata oppure le concessioni sulla chiave sono state revocate. Quando un registro è danneggiato, non è accessibile e non accetta richieste di lettura o scrittura.

  Un registro danneggiato torna automaticamente allo stato attivo dopo aver ripristinato le concessioni sulla chiave o aver riattivato la chiave che era disattivata. Tuttavia, l'eliminazione di una chiave KMS gestita dal cliente è irreversibile. Dopo l'eliminazione di una chiave, non è più possibile accedere ai registri protetti con tale chiave e i dati diventano irrecuperabili in modo permanente.

➨Tipo: stringa

Valori validi: `ENABLED | UPDATING | KMS_KEY_INACCESSIBLE`

Campo obbligatorio: sì

KmsKeyArn

L'Amazon Resource Name (ARN) della chiave KMS gestita dal cliente che il registro utilizza per la crittografia a riposo. Se questo parametro non è definito, il registro utilizza una chiave KMS di AWS proprietà per la crittografia. Verrà visualizzato `AWS_OWNED_KMS_KEY` quando si aggiorna la configurazione di crittografia del registro con la AWS chiave KMS proprietaria.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

Campo obbligatorio: sì

InaccessibleKmsKeyDateTime

La data e l'ora, in formato epoch time, in cui la AWS KMS chiave è diventata inaccessibile per la prima volta, in caso di errore. (Il formato dell'ora epoch è il numero di secondi trascorsi dalle 00:00 del 1° gennaio 1970 UTC.)

Questo parametro non è definito se la chiave è accessibile. AWS KMS

Tipo: Timestamp

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

LedgerSummary

Servizio: Amazon QLDB

Informazioni su un libro mastro, inclusi il nome, lo stato e la data di creazione.

Indice

CreationDateTime

La data e l'ora, in formato epocale, in cui è stato creato il libro contabile. (Il formato dell'ora epoch è il numero di secondi trascorsi dalle 00:00 del 1° gennaio 1970 UTC.)

Tipo: Timestamp

Campo obbligatorio: no

Name

Il nome del libro mastro.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: no

State

Lo stato attuale del libro mastro.

-Tipo: stringa

Valori validi: `CREATING | ACTIVE | DELETING | DELETED`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++

- AWS SDK per Java V2
- AWS SDK per Ruby V3

S3EncryptionConfiguration

Servizio: Amazon QLDB

Le impostazioni di crittografia utilizzate da un processo di esportazione del diario per scrivere dati in un bucket Amazon Simple Storage Service (Amazon S3).

Indice

ObjectEncryptionType

Il tipo di crittografia degli oggetti Amazon S3.

Per ulteriori informazioni sulle opzioni di crittografia lato server in Amazon S3, consulta Protection Data Using Server-Side Encryption nella Amazon S3 Developer Guide.

- Tipo: stringa

Valori validi: `SSE_KMS | SSE_S3 | NO_ENCRYPTION`

Campo obbligatorio: sì

KmsKeyArn

L'Amazon Resource Name (ARN) di una chiave di crittografia simmetrica in (). AWS Key Management Service AWS KMS Amazon S3 non supporta chiavi KMS asimmetriche.

È necessario fornire un `KmsKeyArn` se si specifica come. `SSE_KMS` `ObjectEncryptionType`

`KmsKeyArn`non è obbligatorio se si specifica `SSE_S3` come`ObjectEncryptionType`.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 1600 caratteri.

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++

- AWS SDK per Java V2
- AWS SDK per Ruby V3

S3ExportConfiguration

Servizio: Amazon QLDB

La posizione del bucket Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) in cui un processo di esportazione del diario scrive il contenuto del diario.

Indice

Bucket

Il nome del bucket Amazon S3 in cui un processo di esportazione del diario scrive il contenuto del journal.

Il nome del bucket deve essere conforme alle convenzioni di denominazione dei bucket di Amazon S3. Per ulteriori informazioni, consulta Restrizioni e limitazioni dei bucket nella Amazon S3 Developer Guide.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 3. Lunghezza massima di 255.

Modello: `^[A-Za-z-0-9-_.]+$`

Campo obbligatorio: sì

EncryptionConfiguration

Le impostazioni di crittografia utilizzate da un processo di esportazione del diario per scrivere dati in un bucket Amazon S3.

Tipo: oggetto S3EncryptionConfiguration

Campo obbligatorio: sì

Prefix

Il prefisso per il bucket Amazon S3 in cui un processo di esportazione del diario scrive il contenuto del journal.

Il prefisso deve essere conforme alle regole e alle restrizioni di denominazione delle chiavi di Amazon S3. Per ulteriori informazioni, consulta Object Key and Metadata nella Amazon S3 Developer Guide.

Di seguito sono riportati alcuni esempi di valori validi`Prefix`:

- `JournalExports-ForMyLedger/Testing/`
- `JournalExports`
- `My:Tests/`

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. La lunghezza massima è 128 caratteri.

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

ValueHolder

Servizio: Amazon QLDB

Una struttura che può contenere un valore in più formati di codifica.

Indice

IonText

Un valore di testo non crittografato di Amazon Ion contenuto in una `ValueHolder` struttura.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 1048576.

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli SDK specifici della lingua, consulta quanto segue AWS :

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

Sessione Amazon QLDB

I tipi di dati seguenti sono supportati da Amazon QLDB Session:

- AbortTransactionRequest
- AbortTransactionResult
- CommitTransactionRequest
- CommitTransactionResult
- EndSessionRequest
- EndSessionResult
- ExecuteStatementRequest
- ExecuteStatementResult

- FetchPageRequest
- FetchPageResult
- IOUsage
- Page
- StartSessionRequest
- StartSessionResult
- StartTransactionRequest
- StartTransactionResult
- TimingInformation
- ValueHolder

AbortTransactionRequest

Servizio: Amazon QLDB Session

Contiene i dettagli della transazione da interrompere.

Indice

I membri di questa struttura di eccezioni dipendono dal contesto.

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

AbortTransactionResult

Servizio: Amazon QLDB Session

Contiene i dettagli della transazione interrotta.

Indice

TimingInformation

Contiene informazioni sulle prestazioni del comando sul lato server.

Tipo: oggetto TimingInformation

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

CommitTransactionRequest

Servizio: Amazon QLDB Session

Contiene i dettagli della transazione da eseguire.

Indice

CommitDigest

Specifica il digest di commit per la transazione da confermare. Per ogni transazione attiva, è necessario passare il digest di commit. QLDB `CommitDigest` convalida e rifiuta il commit con un errore se il digest calcolato sul client non corrisponde al digest calcolato da QLDB.

Lo scopo del `CommitDigest` parametro è garantire che QLDB esegua una transazione se e solo se il server ha elaborato l'esatto set di istruzioni inviate dal client, nello stesso ordine in cui il client le ha inviate e senza duplicati.

Tipo: oggetto dati binari con codifica Base64

Campo obbligatorio: sì

TransactionId

Speciifica l'ID della transazione da confermare.

-Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 22.

Modello: `^[A-Za-z-0-9]+$`

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

CommitTransactionResult

Servizio: Amazon QLDB Session

Contiene i dettagli della transazione confermata.

Indice

CommitDigest

Il riepilogo di commit della transazione impegnata.

Tipo: oggetto dati binari con codifica Base64

Campo obbligatorio: no

ConsumedIOs

Contiene metriche sul numero di richieste di I/O che sono state consumate.

Tipo: oggetto IOUsage

Campo obbligatorio: no

TimingInformation

Contiene informazioni sulle prestazioni lato server per il comando.

Tipo: oggetto TimingInformation

Campo obbligatorio: no

TransactionId

L'ID della transazione confermata.

- Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 22.

Modello: `^[A-Za-z-0-9]+$`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

EndSessionRequest

Servizio: Amazon QLDB Session

Specifica una richiesta per terminare la sessione.

Indice

I membri di questa struttura di eccezioni dipendono dal contesto.

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

EndSessionResult

Servizio: Amazon QLDB Session

Contiene i dettagli della sessione terminata.

Indice

TimingInformation

Contiene informazioni sulle prestazioni del comando sul lato server.

Tipo: oggetto TimingInformation

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

ExecuteStatementRequest

Servizio: Amazon QLDB Session

Specifica una richiesta di esecuzione di un'istruzione.

Indice

Statement

Specifica la dichiarazione della richiesta.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100000.

Campo obbligatorio: sì

TransactionId

Speciifica l'ID della transazione della richiesta.

- Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 22.

Modello: `^[A-Za-z-0-9]+$`

Campo obbligatorio: sì

Parameters

Specificate i parametri per l'istruzione parametrizzata nella richiesta.

Tipo: matrice di oggetti ValueHolder

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli SDK specifici della lingua, consulta quanto segue AWS :

- AWS SDK per C++
- AWS SDK per Java V2

- AWS SDK per Ruby V3

ExecuteStatementResult

Servizio: Amazon QLDB Session

Contiene i dettagli dell'istruzione eseguita.

Indice

ConsumedIOs

Contiene metriche sul numero di richieste di I/O che sono state utilizzate.

Tipo: oggetto IOUsage

Campo obbligatorio: no

FirstPage

Contiene i dettagli della prima pagina recuperata.

Tipo: oggetto Page

Campo obbligatorio: no

TimingInformation

Contiene informazioni sulle prestazioni del comando sul lato server.

Tipo: oggetto TimingInformation

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

FetchPageRequest

Servizio: Amazon QLDB Session

Specificate i dettagli della pagina da recuperare.

Indice

NextPageToken

Speciifica il token della pagina successiva della pagina da recuperare.

- Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 4. La lunghezza massima è 1024 caratteri.

Modello: `^[A-Za-z-0-9+/=]+$`

Campo obbligatorio: sì

TransactionId

Specificate l'ID della transazione della pagina da recuperare.

- Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 22.

Modello: `^[A-Za-z-0-9]+$`

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

FetchPageResult

Servizio: Amazon QLDB Session

Contiene la pagina che è stata recuperata.

Indice

ConsumedIOs

Contiene metriche sul numero di richieste di I/O utilizzate.

Tipo: oggetto IOUsage

Campo obbligatorio: no

Page

Contiene i dettagli della pagina recuperata.

Tipo: oggetto Page

Campo obbligatorio: no

TimingInformation

Contiene informazioni sulle prestazioni del comando sul lato server.

Tipo: oggetto TimingInformation

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

IOUsage

Servizio: Amazon QLDB Session

Contiene le metriche di utilizzo dell'I/O per un comando che è stato richiamato.

Indice

ReadIOs

Il numero di richieste di I/O di lettura effettuate dal comando.

Tipo: long

Campo obbligatorio: no

WriteIOs

Il numero di richieste di I/O di scrittura effettuate dal comando.

Tipo: long

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

Page

Servizio: Amazon QLDB Session

Contiene i dettagli della pagina recuperata.

Indice

NextPageToken

Il token della pagina successiva.

Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 4. La lunghezza massima è 1024 caratteri.

Modello: `^[A-Za-z-0-9+/=]+$`

Campo obbligatorio: no

Values

Una struttura che contiene valori in più formati di codifica.

Tipo: matrice di oggetti ValueHolder

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

StartSessionRequest

Servizio: Amazon QLDB Session

Specifica una richiesta per iniziare una nuova sessione.

Indice

LedgerName

Il nome del libro mastro su cui iniziare una nuova sessione.

Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 32 caratteri.

Modello: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

StartSessionResult

Servizio: Amazon QLDB Session

Contiene i dettagli della sessione avviata.

Indice

SessionToken

Token di sessione della sessione iniziata. Questo `SessionToken` è necessario per ogni comando successivo emesso durante la sessione corrente.

- Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 4. La lunghezza massima è 1024 caratteri.

Modello: `^[A-Za-z-0-9+/=]+$`

Campo obbligatorio: no

TimingInformation

Contiene informazioni sulle prestazioni lato server per il comando.

Tipo: oggetto TimingInformation

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

StartTransactionRequest

Servizio: Amazon QLDB Session

Specifica una richiesta per avviare una transazione.

Indice

I membri di questa struttura di eccezioni dipendono dal contesto.

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

StartTransactionResult

Servizio: Amazon QLDB Session

Contiene i dettagli della transazione avviata.

Indice

TimingInformation

Contiene informazioni sulle prestazioni del comando sul lato server.

Tipo: oggetto TimingInformation

Campo obbligatorio: no

TransactionId

L'ID della transazione avviata.

-Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 22.

Modello: `^[A-Za-z-0-9]+$`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

TimingInformation

Servizio: Amazon QLDB Session

Contiene informazioni sulle prestazioni lato server per un comando. Amazon QLDB acquisisce informazioni sulla tempistica tra il momento in cui riceve la richiesta e quello in cui invia la risposta corrispondente.

Indice

ProcessingTimeMilliseconds

La quantità di tempo impiegata da QLDB per l'elaborazione del comando, misurata in millisecondi.

Tipo: long

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli SDK specifici della lingua AWS , consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

ValueHolder

Servizio: Amazon QLDB Session

Una struttura che può contenere un valore in più formati di codifica.

Indice

IonBinary

Un valore binario Amazon Ion contenuto in una `ValueHolder` struttura.

Tipo: oggetto dati binari con codifica Base64

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 131072.

Campo obbligatorio: no

IonText

Un valore di testo non crittografato di Amazon Ion contenuto in una `ValueHolder` struttura.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 1048576.

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli SDK specifici della lingua, consulta quanto segue AWS :

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

Errori comuni

In questa sezione sono riportati gli errori comuni delle azioni API per tutti i servizi AWS. Per gli errori specifici di un'azione API per questo servizio, consulta l'argomento per quell'azione API.

AccessDeniedException

Non disponi dell'autorizzazione di accesso sufficiente per eseguire questa operazione.

Codice di stato HTTP: 400

IncompleteSignature

La firma della richiesta non è conforme agli standard AWS.

Codice di stato HTTP: 400

InternalFailure

L'elaborazione della richiesta non è riuscita a causa di un errore, un'eccezione o un guasto interno sconosciuto.

Codice di stato HTTP: 500

InvalidAction

L'azione o l'operazione richiesta non è valida. Verifica che l'operazione sia digitata correttamente.

Codice di stato HTTP: 400

InvalidClientTokenId

Il certificato X.509 o l'ID chiave di accesso AWS forniti non sono presenti nei nostri record.

Codice di stato HTTP: 403

NotAuthorized

Non disponi delle autorizzazioni per eseguire questa azione.

Codice di stato HTTP: 400

OptInRequired

L'ID chiave di accesso AWS necessita di una sottoscrizione al servizio.

Codice di stato HTTP: 403

RequestExpired

La richiesta ha raggiunto il servizio più di 15 minuti dopo il date stamp della richiesta o più di 15 minuti dopo la data di scadenza della richiesta (ad esempio per URL prefirmati) oppure il date stamp della richiesta è più di 15 minuti nel futuro.

Codice di stato HTTP: 400

ServiceUnavailable

La richiesta non è riuscita a causa di un errore temporaneo del server.

Codice di stato HTTP: 503

ThrottlingException

La richiesta è stata negata a causa del throttling della richiesta.

Codice di stato HTTP: 400

ValidationError

L'input non riesce a soddisfare i vincoli specificati da un servizio AWS.

Codice di stato HTTP: 400

Parametri comuni

L'elenco seguente contiene i parametri utilizzati da tutte le azioni per firmare le richieste di Signature Version 4 con una stringa di query. Qualsiasi parametro specifico di un'operazione è riportato nell'argomento relativo all'operazione. Per ulteriori informazioni sull'utilizzo di Signature Version 4, consulta la pagina Firma delle richieste API AWS nella Guida per l'utente di IAM.

Action

azione da eseguire.

Tipo: stringa

Campo obbligatorio: sì

Version

Versione dell'API per cui è scritta la richiesta, espressa nel formato AAAA-MM-GG.

Tipo: stringa

Campo obbligatorio: sì

X-Amz-Algorithm

Algoritmo hash utilizzato per creare la firma della richiesta.

Condition: specifica questo parametro quando includi le informazioni di autenticazione in una stringa di query anziché nell'intestazione di autorizzazione HTTP.

Tipo: stringa

Valori validi: `AWS4-HMAC-SHA256`

Obbligatorio: condizionale

X-Amz-Credential

Il valore dell'ambito delle credenziali, che è una stringa che include la chiave di accesso, la data, la regione di destinazione, il servizio richiesto e una stringa di terminazione ("aws4_request"). Il valore viene espresso nel seguente formato: chiave_accesso/AAAAMMGG/regione/servizio/aws4_request.

Per ulteriori informazioni, consulta la pagina Creazione di una richiesta API AWS firmata nella Guida per l'utente di IAM.

Condition: specifica questo parametro quando includi le informazioni di autenticazione in una stringa di query anziché nell'intestazione di autorizzazione HTTP.

Tipo: stringa

Obbligatorio: condizionale

X-Amz-Date

La data utilizzata per creare la firma. Il formato deve essere il formato di base ISO 8601 (YYYYMMDD'T'HHMMSS'Z'). Ad esempio, la seguente combinazione data/ora è un valore X-Amz-Date valido: `20120325T120000Z`.

Condition: X-Amz-Date è facoltativo per tutte le richieste; può essere utilizzato per sovrascrivere la data utilizzata per firmare le richieste. Se l'intestazione Date è specificata nel formato base ISO 8601, X-Amz-Date non è richiesto. Quando utilizzi X-Amz-Date, sostituisce sempre il valore dell'intestazione Date. Per ulteriori informazioni, consulta la pagina Elementi di una firma di richiesta API AWS nella Guida per l'utente di IAM.

Tipo: stringa

Obbligatorio: condizionale

X-Amz-Security-Token

Il token di sicurezza provvisorio ottenuto tramite una chiamata ad AWS Security Token Service (AWS STS). Per un elenco di servizi che supportano le credenziali di sicurezza temporanee da AWS STS, consulta la pagina Servizi AWS che funzionano con IAM nella Guida per l'utente di IAM.

Condizione: se utilizzi le credenziali di sicurezza temporanee fornite da AWS STS, devi includere il token di sicurezza.

Tipo: stringa

Obbligatorio: condizionale

X-Amz-Signature

Specifica la firma con codifica esadecimale calcolata dalla stringa da firmare e dalla chiave di firma derivata.

Condition: specifica questo parametro quando includi le informazioni di autenticazione in una stringa di query anziché nell'intestazione di autorizzazione HTTP.

Tipo: stringa

Obbligatorio: condizionale

X-Amz-SignedHeaders

Specifica tutte le intestazioni HTTP incluse come parte della richiesta canonica. Per ulteriori informazioni sulla specifica delle intestazioni firmate, consulta la pagina Creazione di una richiesta API AWS firmata nella Guida per l'utente di IAM.

Condition: specifica questo parametro quando includi le informazioni di autenticazione in una stringa di query anziché nell'intestazione di autorizzazione HTTP.

Tipo: stringa

Obbligatorio: condizionale

Quote e limiti in Amazon QLDB

Questa sezione descrive le quote correnti, definite anche limiti, in Amazon QLDB.

Argomenti

Quote di default

QLDB ha le seguenti quote predefinite, elencate anche negli endpoint e nelle quote di Amazon QLDB nel Riferimenti generali di AWS. Queste quote si intendonoAccount AWS per regione. Per richiedere un aumento delle quote per il tuo account in una regione, utilizza la console Service Quotas.

Accedere allaAWS Management Console e aprire la console Service Quotas all'indirizzo https://console.aws.amazon.com/servicequotas/.

Risorsa	Quota predefinita
Il numero massimo di registri attivi che puoi creare in questo account nella regione corrente	5
Il numero massimo di esportazioni di registrazioni attive in Amazon S3 per contabilità	2
Il numero massimo di flussi di registrazioni attivi verso Kinesis Data Streams per libro contabile	5

Quote fisse

Oltre alle quote predefinite, QLDB dispone delle seguenti quote fisse per libro contabile. Queste quote non possono essere aumentate utilizzando Service Quotas:

Risorsa	Quota fissa
Numero di sessioni attive simultanee	1500
Numero di tabelle attive	20
Numero totale di tabelle (attive e inattive) **Note** In QLDB, le tabelle eliminate sono considerate inattive e vengono conteggiate in questa quota totale.	40
Numero di indici per tabella	5
Numero di documenti in una transazione	40
Numero di revisioni da redigere in una transazione	1
Dimensione del documento (codificato in `IonBinary` formato)	128 KB
Dimensione dei parametri della dichiarazione (`IonBinary` formato)	128 KB
Dimensione dei parametri della dichiarazione (`IonText`formato)	1 MB
Lunghezza della stringa di dichiarazione	100.000 caratteri
Dimensione della transazione	4 MB

Risorsa	Quota fissa
Timeout delle transazioni	30 secondi
Periodo di scadenza per i lavori completati di esportazione delle scritture	7 giorni
Periodo di scadenza per i flussi di journal terminali	7 giorni

Quota contabilità

Per richiedere un aumento delle quote contabilità per il tuo account in una regione, puoi utilizzare la console Service Quotas.

Apri la console Service Quotas all'indirizzo https://console.aws.amazon.com/servicequotas/

Alcuni casi d'uso di QLDB richiedono un numero crescente di registriAccount AWS per regione in base alla crescita del business. Ad esempio, potresti aver bisogno di creare registri dedicati per isolare clienti o dati. In questo caso, considera l'utilizzo di un'architettura multi-account per lavorare con le quote QLDB. Per ulteriori informazioni, consulta Account Silo Isolation nelAWS white paper SaaS Tenant Isolation Strategies.

Dimensioni dei documenti

La dimensione massima di un documento codificato nel`IonBinary` formato è 128 KB. Non possiamo fornire un limite esatto per la dimensione di un documento in`IonText` formato perché la conversione da testo a file binario varia notevolmente in base alla struttura di ciascun documento. QLDB supporta documenti con contenuto aperto, quindi ogni struttura di documento univoca altera il calcolo delle dimensioni.

Dimensioni delle transazioni

La dimensione massima per una transazione in QLDB è 4 MB. La dimensione di una transazione viene calcolata in base alla somma dei seguenti fattori.

Delta

Le modifiche al documento generate da tutte le dichiarazioni all'interno della transazione. In una transazione che ha un impatto su più documenti, la dimensione delta totale è la somma del delta individuale di ciascun documento interessato.

Metadati

I metadati delle transazioni generati dal sistema e associati a ciascun documento interessato.

Indici

Se un indice è definito su una tabella interessata dalla transazione, anche la voce dell'indice associata genera un delta.

Cronologia

Poiché tutte le revisioni dei documenti vengono mantenute in QLDB, anche tutte le transazioni vengono aggiunte alla cronologia.

Inserimenti: ogni documento inserito in una tabella ha anche una copia inserita nella tabella della cronologia. Ad esempio, un documento da 100 KB appena inserito genera un minimo di 200 KB di delta in una transazione. (Si tratta di una stima approssimativa che non include metadati o indici).

Aggiornamenti: qualsiasi aggiornamento del documento, anche per un singolo campo, crea una nuova revisione dell'intero documento nella cronologia, più o meno il delta dell'aggiornamento. Ciò significa che un piccolo aggiornamento in un documento di grandi dimensioni genererebbe comunque un grande delta di transazioni. Ad esempio, l'aggiunta di 2 KB di dati in un documento esistente da 100 KB crea una nuova revisione di 102 KB nella cronologia. Ciò equivale ad almeno 104 KB di delta totali in una transazione. (Ancora una volta, questa stima non include metadati o indici).

Eliminazioni: analogamente agli aggiornamenti, qualsiasi transazione di eliminazione crea una nuova revisione del documento nella cronologia. Tuttavia, la`DELETE` revisione appena creata è più piccola del documento originale perché contiene dati utente nulli e contiene solo metadati.

Vincoli per la denominazione

Nella seguente tabella sono descritti i vincoli per la denominazione all'interno di Amazon QLDB.

Nome del contabilità	• Devono contenere solo da 1 a 32 caratteri alfanumerici o trattini.

Nome del flusso di diari	• Deve contenere una lettera o un numero per il primo e l'ultimo carattere. • Non devono essere tutti numeri. • Non possono contenere due trattini consecutivi. • Applica la distinzione tra lettere.
Nome tabella	• Devono contenere solo da 1 a 128 caratteri alfanumerici o trattini. • Deve contenere una lettera o un trattino. • Può contenere qualsiasi combinazione di caratteri alfanumerici e caratteri di sottolineatura per i caratteri rimanenti. • Applica la distinzione tra lettere. • Non deve essere una parola riservata QLDB PartiQL.

Informazioni correlate ad Amazon QLDB

Le risorse correlate seguenti possono rivelarsi utili durante l'utilizzo di questo servizio.

Argomenti

- Documentazione tecnica
- GitHub repository
- AWSpost e articoli di blog
- Media
- Risorse AWS generali

Documentazione tecnica

- Domande frequenti su Amazon QLDB: domande frequenti sul prodotto.
- Prezzi Amazon QLDB: informazioniAWS sui prezzi ed esempi.
- AWS re:Post— forumAWS della community per domande e risposte (Q&A).
- Amazon Ion: guide per sviluppatori, guide per l'utente e riferimenti per il formato dati Amazon Ion.
- PartiQL: un documento di specifiche e tutorial generali per il linguaggio di interrogazione PartiQL.
- Workshop QLDB: workshop che forniscono esempi pratici e pratici di utilizzo di Amazon QLDB per creare system-of-record applicazioni, inclusi i seguenti laboratori:
  - Apprendazioni QLDB
  - Utilizzo di Amazon Ion e conversione di Ion da e verso JSON (Java)
  - UtilizzoAWS Glue di Amazon Athena per abilitare i dati QLDB per un data lake
  - FIAmazon QLDB
- Dati di qualità a prova di manomissione con Amazon QLDB: un'implementazione diAWS soluzioni che mostra come impedire agli aggressori di manomettere dati di qualità utilizzando QLDB per mantenere una cronologia accurata delle modifiche ai dati. AWS Le implementazioni delle soluzioni ti aiutano a risolvere i problemi più comuni e a creare più rapidamenteAWS.

GitHub repository

Driver

- Driver.NET: un'implementazione .NET del driver QLDB.
- Driver Go: un'implementazione Go del driver QLDB.
- Driver Java: un'implementazione Java del driver QLDB.
- Driver Node.js: un'implementazione Node.js del driver QLDB.
- Driver Python: un'implementazione Python del driver QLDB.

Shell della riga di comando

- Shell QLDB: un'implementazione Python e Rust di un'interfaccia a riga di comando per l'API dei dati transazionali QLDB.

Applicazioni di esempio

- Applicazione Java DMV: un'applicazione tutorial basata su un caso d'uso del dipartimento dei veicoli a motore (DMV). Dimostra le operazioni di base e le migliori pratiche per l'utilizzo di QLDB e del driver QLDB per Java.
- Applicazione.NET DMV: un'applicazione tutorial basata su DMV che dimostra le operazioni di base e le best practice per l'utilizzo di QLDB e del driver QLDB per .NET.
- Applicazione DMV Node.js: un'applicazione tutorial basata su DMV che dimostra le operazioni di base e le best practice per l'utilizzo di QLDB e del driver QLDB per Node.js.
- Applicazione Python DMV: un'applicazione tutorial basata su DMV che dimostra le operazioni di base e le migliori pratiche per l'utilizzo di QLDB e del driver QLDB per Python.
- Ledger loader: un framework Java per il caricamento asincrono di dati in un registro QLDB ad alta velocità utilizzando un canale di distribuzione supportato (AWS DMSAmazon SQS, Amazon SNS, Kinesis Data Streams, Amazon MSK o EventBridge).
- Processore di esportazione: un framework Java estensibile che gestisce il lavoro di elaborazione delle esportazioni QLDB in Amazon S3 leggendo l'output dell'esportazione e iterando i blocchi esportati in sequenza.
- Streaming QLDB di esempio Lambda in Python: un'applicazione che dimostra come utilizzare flussi QLDB utilizzando unaAWS Lambda funzione per inviare dati QLDB a un argomento Amazon SNS, a cui è sottoscritta una coda Amazon SQS.

- Esempio di OpenSearch integrazione di flussi QLDB: un'applicazione Python che dimostra come integrare Amazon OpenSearch Service con QLDB utilizzando gli stream.
- Applicazione in partita doppia: un'applicazione Java che dimostra come modellare un'applicazione di registro finanziario a partita doppia utilizzando QLDB.
- QLDB KVS per Node.js — Una semplice libreria di interfaccia di archiviazione chiave-valore per QLDB con funzioni extra per la verifica dei documenti.

Amazon PartiQL e

- Librerie Amazon Ion: librerie, strumenti e documentazione supportati dal team Ion.
- Implementazione PartiQL: l'implementazione, le specifiche e i tutorial per PartiQL.

AWSpost e articoli di blog

- Come Earnin ha creato il proprio servizio di contabilità utilizzando Amazon QLDB (16 febbraio 2023) — Descrive come Earnin.com ha utilizzato QLDB per creare un servizio di contabilità per fornire ai propri utenti un'applicazione finanziaria mobile moderna e completa di funzionalità.
- Esporta e analizza i dati del diario di Amazon QLDB utilizzandoAWS Glue Amazon Athena (19 dicembre 2022) — Spiega come puoi utilizzare la funzione di esportazione di QLDB conAWS Glue e Athena per fornire funzionalità di reporting e analisi alle tue architetture basate su registri.
- Come Shinsegae International migliora l'esperienza del cliente e previene la contraffazione con Amazon QLDB (3 agosto 2022) — Descrive come Shinsegae International ha creato un servizio di verifica dell'autenticità digitale utilizzando Amazon QLDB per informare i clienti sull'autenticità dei beni di lusso e fornire la cronologia di acquisto e distribuzione dei prodotti.
- BungkusIT utilizza la tecnologia ISV di Amazon QLDB e VeriDoc Global per migliorare l'esperienza dei clienti e degli agenti di consegna (29 aprile 2022) — Mostra come una società di logistica, BungKusit, ha utilizzato QLDB per implementare una piattaforma centralizzata e trasparente per la comunicazione intersettoriale con un registro delle transazioni immutabile e verificabile crittograficamente.
- Usa Amazon QLDB come archivio chiave-valore immutabile con un'API REST e JSON (14 febbraio 2022): introduce un modo per lavorare con QLDB come archivio chiave-valore immutabile e utilizzare le sue funzionalità di controllo tramite un'API REST.

- Creazione di un sistema bancario principale con Amazon QLDB (21 gennaio 2022) — Mostra come utilizzare QLDB per creare un sistema di contabilità bancaria principale. Mostra anche perché QLDB è un buon fit-for-purpose database adatto alle esigenze del settore dei servizi finanziari.
- Come Specright utilizza Amazon QLDB per creare una rete di catena di fornitura tracciabile (21 gennaio 2022) — Mostra come Specright utilizza QLDB per creare una rete di catena di fornitura tracciabile che consente a marchi all'ingrosso, rivenditori e produttori di condividere dati critici sulla catena di fornitura e dati sulle specifiche degli imballaggi.
- Come FeMR fornisce dati medici crittograficamente sicuri e verificabili con Amazon QLDB (23 dicembre 2021) — Esplora come Team FeMR ha utilizzato QLDB e altri serviziAWS gestiti per consentire le proprie operazioni di soccorso.
- Monitora i modelli di accesso alle query di Amazon QLDB (8 novembre 2021): mostra come associare le transazioni QLDB e le istruzioni PartiQL eseguite nelle transazioni alle richieste API ricevute tramite Amazon API Gateway.
- Crea una semplice operazione CRUD e un flusso di dati su Amazon QLDB utilizzandoAWS Lambda (28 settembre 2021) — Mostra come eseguire operazioni CRUD (creazione, lettura, aggiornamento ed eliminazione) su QLDB utilizzandoAWS Lambda le funzioni.
- Verifica crittografica nel mondo reale con Amazon QLDB (26 agosto 2021): illustra il valore della verifica crittografica in un registro QLDB nel contesto di un caso d'uso realistico.
- Verifica le condizioni di consegna con Accord Project e Amazon QLDB: parte 1, parte 2 (28 giugno 2021) — Spiega come applicare la tecnologia dei contratti legali intelligenti per verificare le condizioni di consegna con Accord Project e QLDB open source.
- Streaming di dati Amazon QLDB tramiteAWS CDK (7 giugno 2021): mostra come utilizzarloAWS Cloud Development Kit (AWS CDK) per configurare QLDB, compilare i dati QLDB utilizzandoAWS Lambda le funzioni e configurare lo streaming QLDB per fornire la resilienza dei dati contabili.
- Dimostrazione dettagliata del controllo degli accessi in QLDB (1 giugno 2021) — Mostra come iniziare a utilizzare le autorizzazioni granulariAWS Identity and Access Management (IAM) per un registro QLDB.
- Elaborazione di stream con stream DynamoDB e flussi QLDB (23 novembre 2020): fornisce un breve confronto tra stream DynamoDB e stream QLDB e suggerimenti su come iniziare a utilizzarli.
- Streaming di dati da Amazon QLDB a OpenSearch (19 agosto 2020): descrive come trasmettere dati da QLDB ad Amazon OpenSearch Service per supportare la ricerca di testo RTF e l'analisi a valle, come l'aggregazione o le metriche tra i record.

- Come ho trasmesso i dati da Amazon QLDB a DynamoDB utilizzando Nodejs in tempo quasi reale (7 luglio 2020) — Descrive come trasmettere dati da QLDB a DynamoDB per supportare latenza a una cifra e richieste di valori chiave-chiave scalabili all'infinito.
- Creazione di un'interfaccia GraphQL per Amazon QLDB conAWS AppSync: Parte 1 , Parte 2 (4 maggio 2020) — Spiega come integrare QLDB eAWS AppSync fornire un'API versatile basata su GraphQL su un registro QLDB.

Media

Video AWS

- AWSTutorial e demo: da QLDB a Aurora Streaming (17 marzo 2023; 21 minuti): questo video spiega i concetti fondamentali per implementare la funzionalità di streaming QLDB e dimostra come configurare lo streaming di dati da QLDB a un database Amazon Aurora MySQL downstream.
- ArcBlock: Sfruttare Amazon QLDB per creare una soluzione di identità decentralizzata (31 maggio 2022; 4 minuti): ArcBlock illustra la soluzione di identità decentralizzata che hanno creato utilizzando QLDB.
- AWSre:Invent 2020: utilizzo di Amazon QLDB come system-of-trust database per le app aziendali principali (5 febbraio 2021; 30 minuti) — Scopri come i primi utenti di Amazon QLDB hanno applicato le proprietà uniche del database contabile per la provenienza dei dati e la verificabilità crittografica per implementare sistemi di registrazione con integrità dei dati integrata.
- AWSre:Invent 2020: creazione di un'applicazione serverless con Amazon QLDB (5 febbraio 2021; 28 minuti) — Scopri come creare, testare e ottimizzare un'applicazione serverless completamente funzionale combinando Amazon QLDB con servizi comeAWS Lambda Amazon Kinesis e Amazon DynamoDB.
- AWSre:Invent 2020: creazione di app basate su audit che mantengono l'integrità dei dati con Amazon QLDB (5 febbraio 2021; 18 minuti) — Questa sessione approfondisce i problemi che Amazon QLDB può risolvere, risponde alle tue domande su quando e perché dovresti usare un database contabile e condivide i casi d'uso di clienti come Osano.
- Come archiviare centralmente i log immutabili utilizzando Amazon QLDB con applicazioni.NET (7 dicembre 2020; 10 minuti): una dimostrazione di come utilizzare QLDB con applicazioni.NET per archiviare centralmente i log immutabili.
- Workshop virtuale: Creazione di sistemi di registrazione basati su registri con QLDB e Java -AWS Online Tech Talks (29 luglio 2020; 87 minuti) - Un seminario con istruttore che illustra il laboratorio

Working With Ion Immersion Day per creare un programma di caricamento dati utilizzando la libreria Amazon Ion e il driver QLDB per Java.

- Workshop per sviluppatori: utilizzoAWS dell'Immutable Ledger Database QLDB sul nodo ABT (22 giugno 2020; 34 minuti) - Una step-by-step guida su come configurare e configurare QLDB sul nodo ABT. Spiega anche come installare e configurare ArcBlock i blocchi Blockchain Explorer e Boarding Gate per connettersi a QLDB.
- AWSTech Talk: Il punto di vista di un cliente sulla creazione di un'applicazione di sistema di registrazione innescata da eventi con Amazon QLDB (31 marzo 2020; 29 minuti) — Un discorso tecnico di Matt Lewis,AWS Data Hero e Chief Architect della Driver and Vehicle Licensing Agency (DVLA) nel Regno Unito, che spiega il caso d'uso di DVLA per QLDB e l'architettura basata sugli eventi della loro applicazione.
- AWSre:Invent 2019: Perché avete bisogno di un database contabile: BMW, DVLA e Sage discutono dei casi d'uso (5 dicembre 2019; 47 minuti) — Una presentazione dei clienti BMW, l'organizzazione governativa britannica DVLA e Sage che illustra i motivi per utilizzare un database di registri e ne condivide i casi d'uso per QLDB.
- AWSre:Invent 2019: Amazon QLDB: un'analisi approfondita di un ingegnere sul perché questo è un punto di svolta (5 dicembre 2019; 50 minuti) — Una presentazione di Andrew Certain (AWSDistinguished Engineer) che illustra l'architettura unica di QLDB, basata sulle pubblicazioni giornalistiche, insieme alle sue varie innovazioni. Include l'hashing crittografico, gli alberi di Merkle, la replica di più zone di disponibilità e il supporto di PartiQL.
- Creazione di applicazioni con Amazon QLDB, un database di registri unico nel suo genere -AWS Online Tech Talks (19 novembre 2019; 51 minuti) — Una conferenza tecnica approfondita che descrive le caratteristiche uniche di QLDB e specifica come utilizzare le funzionalità di base. Include l'interrogazione della cronologia completa dei dati, la verifica crittografica dei documenti e la progettazione di un modello di dati.

Podcast

- Perché i clienti scelgono Amazon QLDB? (5 luglio 2020; 33 minuti) — Una discussione che spiega la definizione di un database contabile, in che modo è diverso da una blockchain e come i clienti lo utilizzano oggi.

Risorse AWS generali

- Corsi e workshop: collegamenti a corsi basati su ruoli e di specializzazione nonché a corsi gestiti dall'utente per affinare le proprie competenze su AWS e acquisire esperienza pratica.
- Centro sviluppatori AWS: esplora i tutorial, scarica gli strumenti e scopri gli eventi destinati agli sviluppatori AWS.
- Strumenti per sviluppatori AWS: collegamenti a strumenti per sviluppatori, SDK, kit di strumenti IDE e strumenti a riga di comando per lo sviluppo e la gestione delle applicazioni AWS.
- Centro risorse per le nozioni di base: scopri come configurare il tuo Account AWS, unisciti alla community AWS e lancia la tua prima applicazione.
- Esercitazioni: step-by-step scopri le esercitazioni suAWS.
- Whitepaper AWS: collegamenti a un elenco completo di whitepaper tecnici AWS, relativi ad argomenti come architettura, sicurezza ed economia, creati da AWS Solutions Architect o da altri esperti tecnici.
- AWS SupportCentro : il centro in cui creare e gestire i tuoi casi AWS Support. Include inoltre link ad altre risorse utili, quali forum, domande frequenti di tipo tecnico, stato d'integrità del servizio e AWS Trusted Advisor.
- AWS Support: pagina Web principale che include le informazioni suAWS Support one-on-one, un canale di assistenza rapida che aiuta a creare ed eseguire applicazioni nel cloud.
- Contatti - Un punto di contatto centrale per richieste relative a fatturazione, account, eventi, uso illecito e altre questioni relative ad AWS.
- AWS Termini di utilizzo del sito: informazioni dettagliate sul copyright e i marchi, l'account, la licenza, l'accesso al sito e altri argomenti.

Cronologia delle versioni per Amazon QLDB

La tabella seguente descrive le modifiche importanti apportate in ogni versione di Amazon QLDB e gli aggiornamenti corrispondenti nella Guida per gli sviluppatori di Amazon QLDB. Per ricevere notifiche sugli aggiornamenti della documentazione, puoi sottoscrivere il feed RSS.

- Versione API: 2019-01-02
- Ultimo aggiornamento della documentazione: 3 gennaio 2023

Modifica	Descrizione	Data
Guida IAM aggiornata	Guida aggiornata per allinearsi alle best practice IAM. Per ulteriori informazioni, consulta la sezione Best practice per la sicurezza in IAM	3 gennaio 2023
Aggiornamento alle politiche AWS gestite	Amazon QLDB ha aggiornato le politiche AWS gestite esistenti `AmazonQLDBFullAccess` e `AmazonQLDBConsoleFullAccess` . Queste politiche dispongono di una nuova autorizzazione per consentire ai responsabili di redigere le revisioni dei documenti utilizzando una procedura memorizzata PartiQL. Per ulteriori informazioni, consulta le policyAWS gestite per Amazon QLDB.	4 novembre 2022
Redazione dei dati	Amazon QLDB ora supporta la procedura di archiviazione`REDACT_REVISION`	3 novembre 2022

	PartiQL per i registri creati a partire dal 22 luglio 2021. Utilizzando questa procedura memorizzata, è possibile eliminare definitivamente le revisioni inattive dei documenti nella cronologia e mantenere comunque l'integrità complessiva dei dati del libro contabile. Per ulteriori informazioni, consulta Redazione delle revisioni dei documenti.	
Driver Node.js v3	Il driver Amazon QLDB per Node.js versione 3.0 è ora disponibile a livello generale. Questa versione introduce il supporto per laAWS SDK for JavaScript versione 3. Per le note di rilascio, consulta il GitHub repository awslabs/amazon-qldb-driver-nodejs.	26 settembre 2022
Go driver v3	Il driver Amazon QLDB per Go versione 3.0 è ora disponibile a livello generale. Questa versione introduce il supporto per laAWS SDK for Go versione 2. Per le note di rilascio, consulta il GitHub repository awslabs/amazon-qldb-driver-go.	11 agosto 2022

Nuovo editor di query PartiQL	Un nuovo editor di query PartiQLDB è ora disponibile a livello generale. Il nuovo editor QLDB PartiQL fornisce un'interfaccia migliorata per la creazione di query, il debug delle transazioni e l'esplorazione dei risultati. Per informazioni sull'apertura e l'utilizzo dell'editor, consulta Accesso ad Amazon QLDB tramite la console.	22 giugno 2022
Mappatore di oggetti Ion per il driver.NET	Il driver Amazon QLDB per la versione 1.3 .NET introduce il supporto per il mappatore di oggetti Ion. Questa funzionalità consente di aggirare completamente la necessità di eseguire la conversione manuale tra i tipi Amazon Ion e i tipi C# nativi. Per la cronologia completa delle modifiche del mappatore di oggetti Ion, vedere il file ChangeLog.md nel GitHub repository `amzn/ion-object-mapper-dotnet` .	19 gennaio 2022

Formato di esportazione del giornale JSON	Amazon QLDB ora supporta il formato di output JSON Lines per le esportazioni di registrazioni. Per ulteriori informazioni, consulta Esportazione dei dati del giornale da Amazon QLDB.	21 dicembre 2021
Risoluzione dei problemi di Amazon QLDB	È stato aggiunto un nuovo argomento sulla risoluzione dei problemi che fornisce indicazioni per un elenco aggregato di errori comuni che potresti riscontrare durante l'utilizzo di Amazon QLDB.	8 dicembre 2021
Lancio di una nuova regione	Amazon QLDB è ora disponibile nella regione Canada (centrale). Per un elenco completo delle regioni disponibili, consulta gli endpoint e le quote di Amazon QLDB nel Riferimenti generali di Amazon Web Services.	11 novembre 2021
Prevenzione del problema «confused deputy» tra servizi	Amazon QLDB ora supporta l'utilizzo delle `aws:SourceArn` chiavi di contesto delle condizioni `aws:SourceAccount` globali nelle policy delle risorse IAM per prevenire il problema del «confuso vice». Per ulteriori informazioni consulta la pagina relativa alla prevenzione del problema "confused deputy" tra servizi.	8 novembre 2021

Shell Amazon QLDB versione 2	La versione 2.0 della shell Amazon QLDB, scritta in Rust, è ora disponibile a livello generale. Per le note di rilascio, consulta il GitHub repository awslabs/amazon-qldb-shell.	14 ottobre 2021
Aggiornamento alle politiche AWS gestite	Amazon QLDB ha aggiornato le politiche AWS gestite esistenti `AmazonQLDBFullAccess` e `AmazonQLDBConsoleFullAccess` . Queste politiche hanno nuove autorizzazioni aggiunte per consentire ai responsabili di trasferire qualsiasi risorsa di ruolo IAM nel tuo account al servizio QLDB. Questa operazione è necessaria per tutte le richieste di flusso e flusso journal. Per ulteriori informazioni, consulta le policyAWS gestite per Amazon QLDB.	2 settembre 2021
AWS KMSChiavi gestite dal cliente	Amazon QLDB ora supporta la crittografia inattiva utilizzando chiavi gestite dal cliente inAWS Key Management Service (AWS KMS) per nuove risorse contabili. Per ulteriori informazioni, consulta Crittografia dei dati inattivi in Amazon QLDB.	22 luglio 2021

Aggiornamento alla politicaAWS gestita	Amazon QLDB ha aggiornato la policyAWS gestita esistente `AmazonQLDBReadOnly` per rimuovere un'`qldb:GetBlock` azione duplicata precedentemente elencata due volte. Per ulteriori informazioni, consulta le policyAWS gestite per Amazon QLDB.	1° luglio 2021
Lancio di una nuova regione	Amazon QLDB è ora disponibile nella regione Europa (Londra). Per un elenco completo delle regioni disponibili, consulta gli endpoint e le quote di Amazon QLDB nel Riferimenti generali di Amazon Web Services.	24 giugno 2021

Aggiornamento alle politiche AWS gestite	Amazon QLDB ha aggiornato le politiche AWS gestite esistenti `AmazonQLDBFullAccess` e `AmazonQLDBConsoleFullAccess` . Queste politiche hanno nuove autorizzazioni aggiunte per consentire ai responsabili di aggiornare la modalità delle autorizzazioni in tutti i registri e di eseguire tutti i comandi PartiQL in tutti i registri delle `STANDARD` autorizzazioni. Per ulteriori informazioni, consulta le policy AWS gestite per Amazon QLDB.	27 maggio 2021
Modalità autorizzazioni standard	Amazon QLDB ora supporta una modalità di `STANDARD` autorizzazione per le risorse contabili. Con la modalità di autorizzazione standard, è possibile controllare l'accesso con una granularità più fine per libri mastri, tabelle e comandi PartiQL. Per ulteriori informazioni, consulta Nozioni di base su modalità di autorizzazione standard.	27 maggio 2021

Statistiche sulle PartiQL	La funzionalità di statistica delle dichiarazioni PartiQL è ora disponibile nell'editor di query sulla console Amazon QLDB. Per ulteriori informazioni, consulta l'argomento relativo alle informazioni.	24 maggio 2021
Tutorial: verifica dei dati tramite unAWS SDK	È stato aggiunto un step-by-step tutorial con esempi di codice che dimostrano come verificare un hash di revisione e un hash di blocco in Amazon QLDB utilizzando l'API QLDB tramite unAWS SDK. Per ulteriori informazioni, consulta Tutorial: Verifica dei dati utilizzando unAWS SDK.	6 maggio 2021
Driver.NET versione 1.2	Il driver Amazon QLDB per la versione .NET 1.2 è ora disponibile a livello generale. Questa versione introduce le API asincrone. Per le note di rilascio, consulta il GitHub repository awslabs/amazon-qldb-driver-dotnet.	1 aprile 2021
`DROP INDEX`Dichiarazione PartiQL	PartiQL in Amazon QLDB ora supporta l'istruzione DROP INDEX. Per ulteriori informazioni, consulta Eliminazione degli indici.	3 marzo 2021

Statistiche sulle PartiQL	La funzione di statistica delle dichiarazioni PartiQL è ora disponibile nell'ultima versione del driver Amazon QLDB per tutte le lingue supportate, inclusi .NET, Go e Python. Per ulteriori informazioni, consulta l'argomento relativo alle informazioni.	25 febbraio 2021
TypeScript tutorial di avvio rapido	Sono stati aggiunti esempi di TypeScript codice nel tutorial di avvio rapido per il driver Amazon QLDB per Node.js.	28 dicembre 2020
Statistiche sulle PartiQL	L'ultima versione del driver Amazon QLDB per Java e Node.js ora fornisce statistiche di esecuzione delle istruzioni che possono aiutarti a eseguire istruzioni PartiQL più efficienti. Per ulteriori informazioni, consulta l'argomento relativo alle informazioni.	22 dicembre 2020
Riferimenti ai libri di cucina per autisti e tutorial di avvio rapido	Sono stati aggiunti riferimenti ai libri di cucina per i driver Go e Node.js, tutorial di avvio rapido per i driver Java e Python e una guida per lavorare con Amazon Ion in QLDB. Per ulteriori informazioni, consulta Nozioni di base su il driver Amazon QLDB.	24 novembre 2020

Shell Amazon QLDB versione 1.1	La versione 1.1 della shell Amazon QLDB è ora disponibile a livello generale. Per le note di rilascio, consulta il GitHub repository awslabs/amazon-qldb-shell.	26 ottobre 2020
Driver Amazon QLDB per Go	Il driver di Amazon QLDB per Go è ora disponibile a livello generale. Questo driver è open source GitHub e consente di utilizzarloAWS SDK for Go per interagire con l'API dei dati transazionali di QLDB. Per le note di rilascio, consulta il GitHub repository awslabs/amazon-qldb-driver-go.	20 ottobre 2020
Suggerimenti per la configurazione del driver Node.js	È stata aggiunta una nuova sezione che fornisce consigli di configurazione per il driver Amazon QLDB per Node.js, incluso come ridurre la latenza riutilizzando le connessioni con keep-alive.	16 ottobre 2020
Creazione di indici su tabelle non vuote	Amazon QLDB ora supporta la creazione di indici su tabelle non vuote. Per ulteriori informazioni, consulta Gestione degli indici.	30 settembre 2020

Ottimizzazione delle prestazioni delle query	È stata aggiunta una nuova sezione che descrive i vincoli delle query in Amazon QLDB e fornisce indicazioni per l'ottimizzazione delle prestazioni delle query in base a questi vincoli.	18 settembre 2020
Riferimento al ricettario per il driver Java	È stato aggiunto un riferimento al Cookbook che mostra esempi di codice di casi d'uso comuni per il driver Amazon QLDB per Java.	3 settembre 2020
Gestione delle sessioni con il conducente	È stata aggiunta una nuova sezione che fornisce una panoramica della gestione delle sessioni con il driver in Amazon QLDB e descrive come il driver QLDB gestisce le sessioni durante l'esecuzione di transazioni di dati.	1 settembre 2020
Driver Java v2.0	Il driver Amazon QLDB per Java versione 2.0 è ora disponibile a livello generale. Per le note di rilascio, consulta il GitHub repository awslabs/amazon-qldb-driver-java.	28 agosto 2020
Driver Node.js v2.0	Il driver Amazon QLDB per Node.js versione 2.0 è ora disponibile a livello generale. Per le note di rilascio, consulta il GitHub repository awslabs/amazon-qldb-driver-nodejs.	27 agosto 2020

Informazioni correlate	È stato aggiunto un nuovo argomento che contiene collegamenti a informazioni correlate e risorse aggiuntive per aiutarti a comprendere e lavorare con Amazon QLDB.	24 agosto 2020
Driver Python v3.0	Il driver Amazon QLDB per Python versione 3.0 è ora disponibile a livello generale. Per le note di rilascio, consulta il GitHub repository awslabs/amazon-qldb-driver-python.	20 agosto 2020
Driver Amazon QLDB per Go	È ora disponibile una versione di anteprima del driver Amazon QLDB per Go. Questo driver consente di utilizzare l'APIAWS SDK for Go per interagire con i dati transazionali di QLDB. Per ulteriori informazioni, consulta l'argomento QLDB di base per Go (Anteprima).	6 agosto 2020
Riferimento al libro di cucina per il driver Python	È stato aggiunto un riferimento al Cookbook che mostra esempi di codice di casi d'uso comuni per il driver Amazon QLDB per Python.	24 luglio 2020
ID univoci in Amazon QLDB	È stata aggiunta una nuova sezione che descrive le proprietà e le linee guida per l'uso degli ID univoci in Amazon QLDB.	9 luglio 2020

AWS CloudFormationrisorsa per flussi di riviste	Amazon QLDB ora supporta una risorsa per la creazione di flussi di diari utilizzando unAWS CloudFormation modello. Per ulteriori informazioni, consulta la AWS::QLDB::Streamrisorsa nella Guida per l'AWS CloudFormationutente.	9 luglio 2020
Riferimento al ricettario per il driver.NET	È stato aggiunto un riferimento al Cookbook che mostra esempi di codice di casi d'uso comuni per il driver Amazon QLDB per .NET.	1 luglio 2020
Driver Amazon QLDB per .NET	Il driver Amazon QLDB per .NET è ora disponibile a livello generale. Questo driver è open source GitHub e consente di utilizzarloAWS SDK for .NET per interagire con l'API dei dati transazionali di QLDB. Per le note di rilascio, consulta il GitHub repository awslabs/amazon-qldb-driver-dotnet.	26 giugno 2020

Driver Amazon QLDB per Node.js	Il driver Amazon QLDB per Node.js è ora disponibile a livello generale. Questo driver è open source GitHub e consente di utilizzare l'AWSSDK per JavaScript in Node.js per interagire con l'API dei dati transazionali di QLDB. Per le note di rilascio, consulta ilamazon-qldb-driver-nodejs GitHub repository awslabs/.	5 giugno 2020
Stream di diari Amazon QLDB	Amazon QLDB ora ti consente di creare uno stream che acquisisce ogni revisione del documento inserita nel tuo diario e li invia ad Amazon Kinesis Data Streams quasi in tempo reale. Per ulteriori informazioni, consulta Streaming dei dati del diario da Amazon QLDB.	19 maggio 2020
Esempi di codice Amazon Ion	Sono stati aggiunti esempi di codice che interrogano ed elaborano i dati Amazon Ion in un registro Amazon QLDB utilizzando il driver QLDB per Java, Node.js e Python. Per ulteriori informazioni, consulta gli esempi di codice Ion in Amazon QLDB.	12 maggio 2020

Modello di concorrenza e contenuto del diario	È stato ampliato l'argomento del modello di concorrenza Amazon QLDB per aggiungere informazioni sull'uso degli indici per limitare i conflitti OCC (Optimistic Concurrency Control). È stata aggiunta una nuova sezione che descrive i contenuti del diario in Amazon QLDB.	4 maggio 2020
Guida di avvio rapido per il driver Node.js	È stata aggiunta una guida rapida per il driver Amazon QLDB per Node.js.	1 maggio 2020
Shell Amazon QLDB	La shell di Amazon QLDB è ora disponibile a livello generale. Questa shell è open source GitHub e fornisce un'interfaccia a riga di comando che consente di eseguire istruzioni PartiQL sui dati contabili. Per le note di rilascio, consulta ilamazon-qldb-shell GitHub repository awslabs/.	20 aprile 2020
Certificazioni di conformità	Amazon QLDB è ora certificato per i programmi diAWS conformità, tra cui HIPAA e ISO. Per ulteriori informazioni, consulta Convalida della conformità per Amazon QLDB.	3 aprile 2020

Consigli estesi per i conducenti	È stata ampliata la sezione dei consigli sui driver Amazon QLDB per applicarla a tutti i linguaggi di programmazione supportati.	2 aprile 2020
Shell Amazon QLDB	È ora disponibile una versione di anteprima della shell Amazon QLDB. Questa shell fornisce un'interfaccia a riga di comando che consente di eseguire istruzioni PartiQL sui dati contabili. Per ulteriori informazioni, consulta Accesso ad Amazon QLDB tramite la shell QLDB (solo API dati) (anteprima).	23 marzo 2020
Driver Java v1.1	Il driver Amazon QLDB per Java versione 1.1 è ora disponibile a livello generale. Per le note di rilascio, consulta ilamazon-qldb-driver-java GitHub repository awslabs/.	20 marzo 2020
Driver Amazon QLDB per .NET	Amazon QLDB ora fornisce una versione di anteprima del driver.NET. Questo driver consente di utilizzare l'APIAWS SDK for .NET per interagire con i dati transazionali di QLDB. Per ulteriori informazioni, consulta il driver Amazon QLDB per .NET (Anteprima).	13 marzo 2020

Driver Amazon QLDB per Python	Il driver di Amazon QLDB per Python è ora disponibile a livello generale. Questo driver è open source GitHub e consente di utilizzarloAWS SDK for Python (Boto3) per interagire con l'API dei dati transazionali di QLDB. Per le note di rilascio, consulta ilamazon-qldb-driver-python GitHub repository awslabs/.	11 marzo 2020
`UNDROP TABLE`Dichiarazione PartiQL e codice HTML annidato	PartiQL in Amazon QLDB ora supporta le istruzioni DML (Data Manipulation Language) in cui le raccolte annidate sono specificate come fonti nella`FROM` clausola. Per ulteriori informazioni, vedere l'istruzione FROM nel riferimento to PartiQL. QLDB PartiQL supporta anche l'istruzione UNDROP TABLE. Per ulteriori informazioni, consulta Riduzione delle informazioni.	26 febbraio 2020
Argomento introduttivo ampliato	Ampliata la parte introduttiva Cos'è Amazon QLDB? argomento che includerà le nuove sezioni Panoramica di Amazon QLDB e Da relazionale a registro.	24 gennaio 2020

Riferimento PartiQL le funzioni supportate	È stato ampliato il riferimento Amazon QLDB PartiQL per includere informazioni dettagliate sull'utilizzo delle funzioni SQL supportate. Per ulteriori informazioni, consulta PartiQL.	2 gennaio 2020
Consigli per i conducenti ed errori comuni	Sono state aggiunte nuove sezioni che descrivono i consigli relativi ai driver per il driver Amazon QLDB per Java e gli errori comuni per tutte le lingue dei driver.	2 gennaio 2020
Lancio di nuove regioni	Amazon QLDB è ora disponibile nelle Regioni Asia Pacifico (Seoul), Asia Pacifico (Tokyo), Asia Pacifico (Tokyo), Asia Pacifico (Tokyo), Asia Pacifico (Francoforte). Per un elenco completo delle regioni disponibili, consulta gli endpoint e le quote di Amazon QLDB nel Riferimenti generali di Amazon Web Services.	19 novembre 2019
Driver Amazon QLDB per Node.js	Amazon QLDB ora fornisce una versione di anteprima del driver Node.js. Questo driver consente di utilizzare l'AWSSDK per JavaScript in Node.js per interagire con l'API dei dati transazionali di QLDB. Per ulteriori informazioni, consulta il driver Amazon QLDB (Anteprima). Node.js	13 novembre 2019

Driver Amazon QLDB per Python	Amazon QLDB ora fornisce una versione di anteprima del driver Python. Questo driver consente di utilizzar e l'APIAWS SDK for Python (Boto3) per interagire con i dati transazionali di QLDB. Per ulteriori informazioni, consulta il driver Amazon QLDB (Anteprima).	29 ottobre 2019
Rilascio pubblico	Questa è la versione pubblica iniziale di Amazon QLDB. Questa versione include la Developer Guide e il riferimento all'API di gestione dei registri integrati.	10 settembre 2019

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.