aws

Panduan Developer

# Amazon Quantum Ledger Database (Amazon QLDB)

# Amazon Quantum Ledger Database (Amazon QLDB): Panduan Developer

# Table of Contents

# Apa itu Amazon QLDB?

Amazon Quantum Ledger Database (Amazon QLDB) adalah basis data buku besar terkelola penuh yang menyediakan log transaksi transparan, berubah-ubah, dan secara kriptografi terkelola penuh yang dimiliki oleh otoritas tepercaya pusat. Anda dapat menggunakan Amazon QLDB untuk melacak semua perubahan data aplikasi, dan mempertahankan riwayat perubahan lengkap dan dapat diverifikasi dari waktu ke waktu. Untuk mempelajari lebih lanjut tentang berbagai opsi database yang tersedia di Amazon Web Services, lihat Memilih database yang tepat untuk organisasi AndaAWS.

Buku besar biasanya digunakan untuk mencatat sejarah kegiatan ekonomi dan keuangan dalam suatu organisasi. Banyak organisasi membangun aplikasi dengan fungsionalitas mirip buku besar karena mereka ingin mempertahankan riwayat data aplikasi mereka yang akurat. Misalnya, mereka mungkin ingin melacak riwayat kredit dan debit dalam transaksi perbankan, memverifikasi garis keturunan data klaim asuransi, atau melacak pergerakan item dalam jaringan rantai pasokan. Aplikasi buku besar sering diimplementasikan menggunakan tabel audit khusus atau jalur audit yang dibuat dalam database relasional.

Amazon QLDB adalah kelas database baru yang membantu menghilangkan kebutuhan untuk terlibat dalam upaya pengembangan kompleks dalam membangun aplikasi mirip buku besar Anda sendiri. Dengan QLDB, riwayat perubahan data Anda tidak dapat diubah — tidak dapat ditimpa atau diubah pada tempatnya. Dan menggunakan kriptografi, Anda dapat memverifikasi bahwa tidak ada perubahan yang tidak diinginkan pada data aplikasi Anda. QLDB menggunakan log transaksional berubah, yang dikenal sebagai jurnal. Jurnal ini hanya ditambahkan dan terdiri dari sekumpulan blok yang diurutkan dan dirantai hash yang berisi data komitmen Anda.

## Video Amazon QLDB

Untuk ikhtisar Amazon QLDB dan bagaimana hal itu dapat bermanfaat bagi Anda, tonton video ikhtisar QLDB ini YouTube.

## Harga Amazon QLDB

Dengan Amazon QLDB, Anda hanya membayar untuk apa yang Anda gunakan tanpa biaya minimum atau penggunaan layanan wajib. Anda hanya membayar untuk sumber daya yang dikonsumsi basis data buku besar Anda, dan Anda tidak perlu menyediakannya terlebih dahulu.

Untuk informasi selengkapnya, lihat Harga Amazon QLDB.

# Memulai dengan QLDB

Kami menyarankan Anda untuk memulai dengan membaca topik berikut:

- Gambaran umum Amazon QLDB- Untuk mendapatkan gambaran umum tingkat tinggi tentang QLDB.
- Konsep inti dan terminologi di Amazon QLDB— Untuk mempelajari konsep dan terminologi QLDB mendasar.
- Mengakses Amazon QLDB- Untuk mempelajari cara mengakses QLDB menggunakanAWS Management Console, API, atauAWS Command Line Interface (AWS CLI).
- Bagaimana Amazon QLDB bekerja dengan IAM- Untuk mempelajari cara mengontrol akses ke QLDB menggunakanAWS Identity and Access Management (IAM).

Untuk memulai dengan cepat menggunakan konsol QLDB, lihatMemulai dengan konsol Amazon QLDB.

Untuk mempelajari tentang mengembangkan dengan QLDB menggunakan driver yangAWS disediakan, lihatMemulai dengan driver Amazon QLDB.

# Gambaran umum Amazon QLDB

Bagian berikut memberikan gambaran umum umum tingkat tinggi tentang komponen layanan Amazon QLDB dan bagaimana mereka berinteraksi.

Topik

- Jurnal pertama
- Tetap
- Dapat diverifikasi secara kriptografi
- SQL-seperti dan dokumen yang fleksibel
- Alat pengembang sumber terbuka
- Tanpa server dan sangat tersedia
- Kelas perusahaan

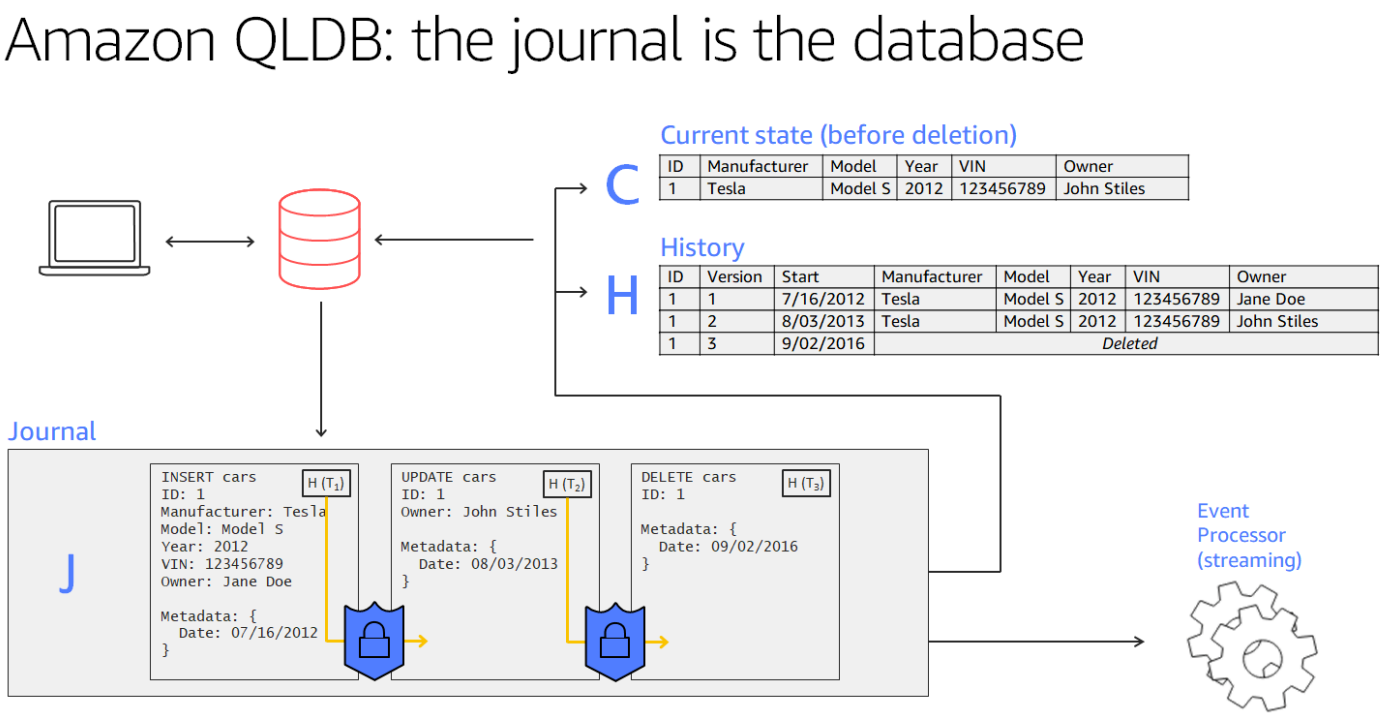
# Jurnal pertama

Dalam arsitektur database tradisional, Anda biasanya menulis data dalam tabel sebagai bagian dari transaksi. Log transaksi — biasanya implementasi internal — mencatat semua transaksi dan modifikasi database yang mereka buat. Log transaksi adalah komponen penting dari database. Anda memerlukan log untuk memutar ulang transaksi jika terjadi kegagalan sistem, pemulihan bencana, atau replikasi data. Namun, log transaksi database tidak berubah dan tidak dirancang untuk memberikan akses langsung dan mudah ke pengguna.

Di Amazon QLDB, jurnal adalah inti dari database. Secara struktural mirip dengan log transaksi, jurnal adalah struktur data yang tidak dapat diubah dan hanya menambahkan yang menyimpan data aplikasi Anda bersama dengan metadata terkait. Semua transaksi tulis, termasuk pembaruan dan penghapusan, berkomitmen untuk jurnal terlebih dahulu.

QLDB menggunakan jurnal untuk menentukan keadaan saat data buku besar Anda dengan mewujudkan itu ke queryable, tabel yang ditetapkan pengguna. Tabel ini juga menyediakan riwayat yang dapat diakses dari semua data transaksi, termasuk revisi dokumen dan metadata. Selain itu, jurnal menangani concurrency, sequencing, verifikasi kriptografi, dan ketersediaan data buku besar.

Diagram berikut mengilustrasikan arsitektur jurnal QLDB.



- Dalam contoh ini, aplikasi terhubung ke buku besar dan menjalankan transaksi yang menyisipkan, memperbarui, dan menghapus dokumen ke dalam tabel bernama`cars`.
- Data pertama kali ditulis ke jurnal dalam urutan berurutan.
- Kemudian data tersebut terwujud ke dalam tabel dengan tampilan bawaan. Tampilan ini memungkinkan Anda menanyakan status saat ini dan riwayat lengkap mobil, dengan setiap revisi menetapkan nomor versi.
- Anda juga dapat mengekspor atau mengalirkan data langsung dari jurnal.

## Tetap

Karena jurnal QLDB hanya ditambahkan, itu menyimpan catatan penuh dari semua perubahan pada data Anda yang tidak dapat dimodifikasi atau ditimpa. Tidak ada API atau metode lain untuk mengubah data berkomitmen apa pun di tempatnya. Struktur jurnal ini memungkinkan Anda mengakses dan menanyakan riwayat lengkap buku besar Anda.

> **Note**
>
> Satu-satunya pengecualian untuk kekekalan yang didukung QLDB adalah redaksi data. Dengan fitur ini, Anda dapat mematuhi undang-undang peraturan seperti Peraturan Perlindungan Data Umum (GDPR) di Uni Eropa dan Undang-Undang Privasi Konsumen California (CCPA).
> QLDB menyediakan operasi redaksi yang memungkinkan Anda menghapus revisi dokumen tidak aktif secara permanen dalam sejarah tabel. Operasi ini hanya menghapus data pengguna dalam revisi yang ditentukan, dan membiarkan urutan jurnal dan metadata dokumen tidak berubah. Ini menjaga integritas data keseluruhan buku besar Anda. Untuk informasi selengkapnya, lihat Menyunting revisi dokumen.

QLDB menulis satu blok ke jurnal dalam transaksi. Setiap blok berisi objek entri yang mewakili dokumen yang Anda masukkan, perbarui, dan hapus, bersama dengan pernyataan yang Anda jalankan untuk melakukan komit. Blok-blok ini diurutkan dan dirantai hash untuk menjamin integritas data.

Diagram berikut mengilustrasikan struktur jurnal ini.

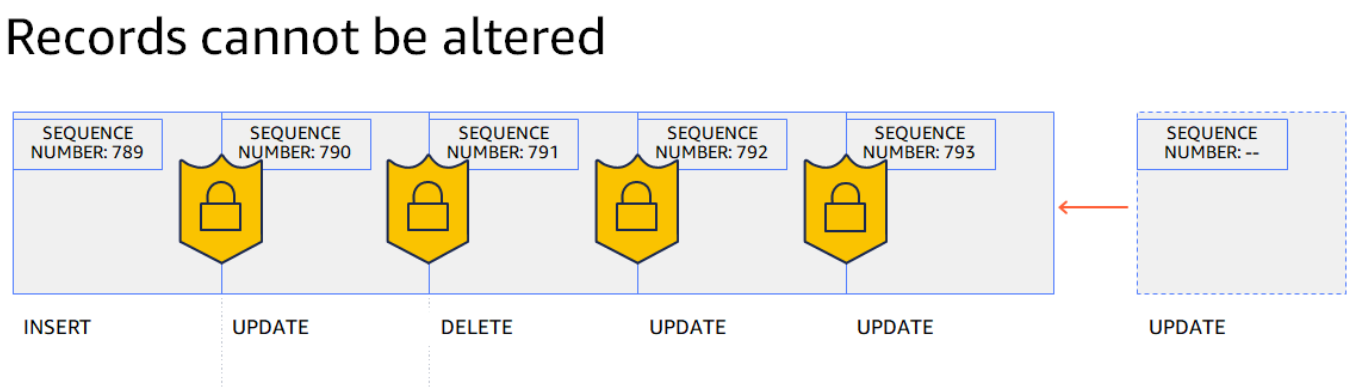
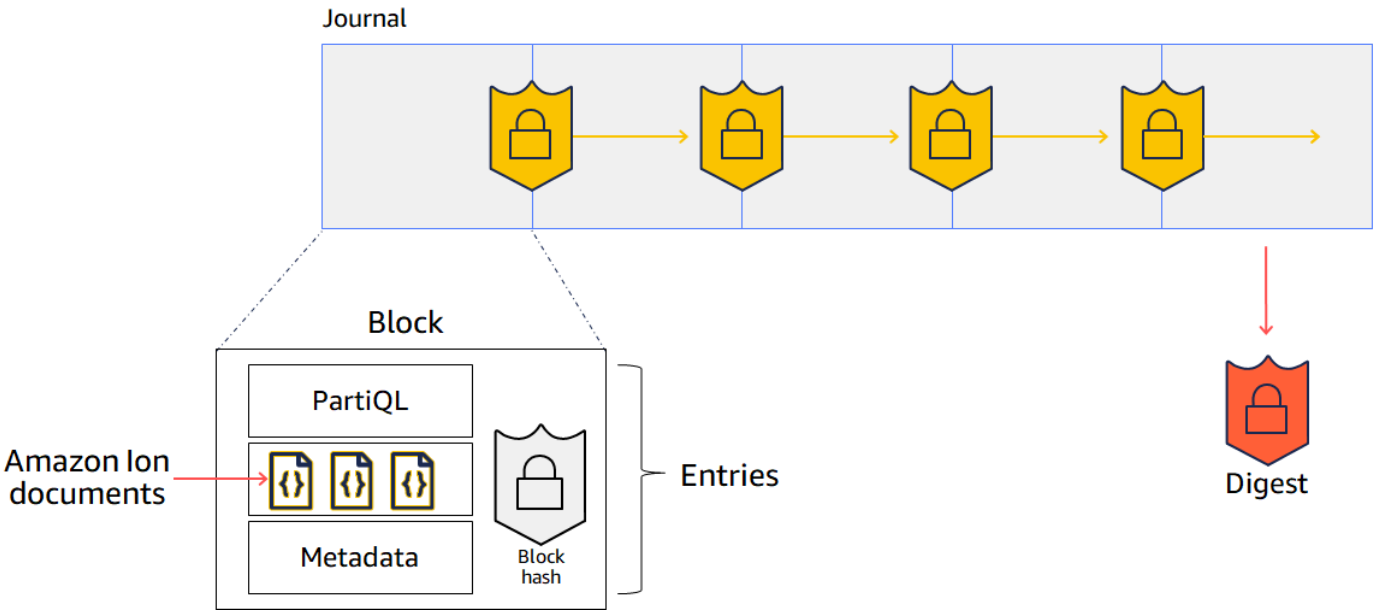



Diagram menunjukkan bahwa transaksi berkomitmen pada jurnal sebagai blok yang dirantai hash untuk verifikasi. Setiap blok memiliki nomor urut untuk menentukan alamatnya.

## Dapat diverifikasi secara kriptografi

Blok jurnal diurutkan dan dirantai bersama dengan teknik hashing kriptografi, mirip dengan blockchain. QLDB menggunakan rantai hash jurnal untuk memberikan integritas data transaksional menggunakan metode verifikasi kriptografi. Menggunakan digest (nilai hash yang mewakili rantai hash penuh jurnal pada titik waktu) dan bukti audit Merkle (mekanisme yang membuktikan validitas node apa pun dalam pohon hash biner), Anda dapat memverifikasi bahwa tidak ada perubahan yang tidak diinginkan pada data Anda kapan saja.

Diagram berikut menunjukkan intisari yang mencakup rantai hash penuh jurnal pada suatu titik waktu.




Dalam diagram ini, blok jurnal di-hash menggunakan fungsi hash kriptografi SHA-256 dan dirantai secara berurutan ke blok berikutnya. Setiap blok berisi entri yang menyertakan dokumen data Anda, metadata, dan pernyataan PartiQL yang berjalan dalam transaksi.

Untuk informasi selengkapnya, lihat Verifikasi data di Amazon QLDB.

## SQL-seperti dan dokumen yang fleksibel

QLDB menggunakan PartiQL sebagai bahasa kueri dan Amazon Ion sebagai model data berorientasi dokumen. PartiQL adalah open-source, bahasa query SQL-kompatibel yang telah diperluas untuk bekerja dengan Ion. Dengan PartiQL, Anda dapat memasukkan, query, dan mengelola data Anda dengan operator SQL akrab. Ketika Anda query dokumen datar, sintaks adalah sama dengan menggunakan SQL untuk query tabel relasional. Untuk mempelajari lebih lanjut tentang implementasi QLDB dari PartiQL, lihatReferensi PartiQLDB QLDB.

Amazon Ion adalah superset dari JSON. Ion adalah open-source, format data berbasis dokumen yang memberi Anda fleksibilitas menyimpan dan memproses data terstruktur, semistruktural, dan bersarang. Untuk mempelajari lebih lanjut tentang Ion dalam QLDB, lihatReferensi format data Amazon Ion di Amazon QLDB.

Untuk perbandingan tingkat tinggi dari komponen inti dan fitur dalam database relasional tradisional versus QLDB, lihatDari relasional ke buku besar.

## Alat pengembang sumber terbuka

Untuk menyederhanakan pengembangan aplikasi, QLDB menyediakan driver open-source dalam berbagai bahasa pemrograman. Anda dapat menggunakan driver ini untuk berinteraksi dengan API data transaksional dengan menjalankan pernyataan PartiQL pada buku besar dan memproses hasil pernyataan tersebut. Untuk informasi dan tutorial tentang bahasa driver yang saat ini didukung, lihatMemulai dengan driver Amazon QLDB.

Amazon Ion juga menyediakan pustaka klien yang memproses data Ion untuk Anda. Untuk panduan pengembang dan contoh kode pemrosesan data Ion, lihat dokumentasi Amazon Ion aktif GitHub.

## Tanpa server dan sangat tersedia

QLDB dikelola sepenuhnya, tanpa server, dan sangat tersedia. Layanan ini secara otomatis menskalakan untuk mendukung permintaan aplikasi Anda, dan Anda tidak perlu menyediakan instans atau kapasitas. Beberapa salinan data Anda direplikasi dalam Availability Zone dan di Availability Zone diWilayah AWS.

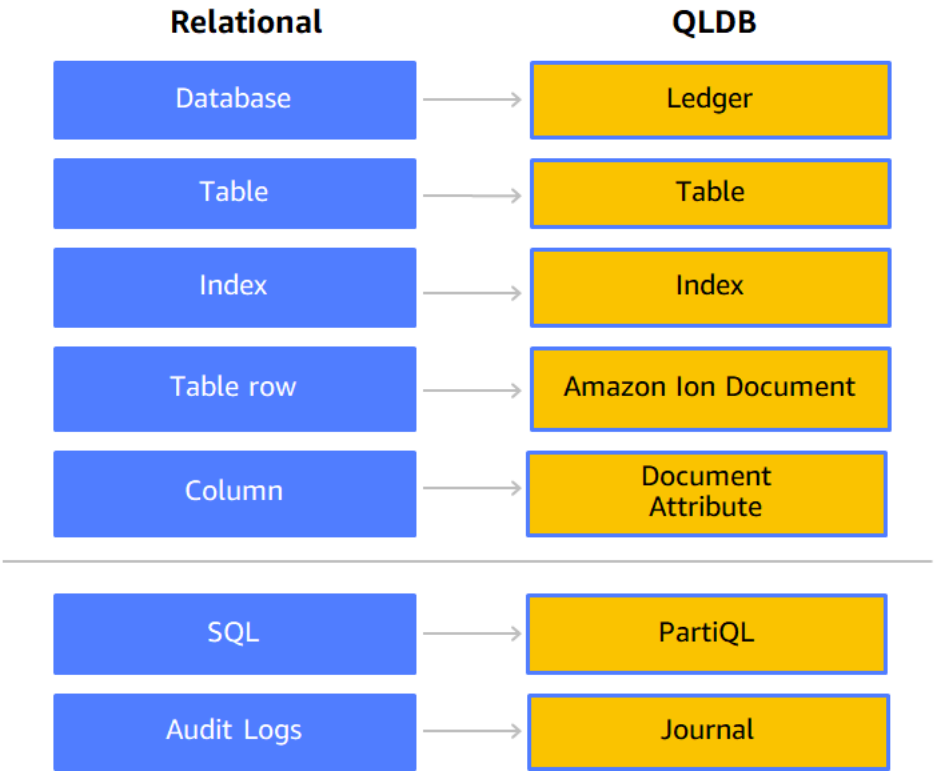
## Kelas perusahaan

Transactions QLDB sepenuhnya sesuai dengan sifat atomisitas, konsistensi, isolasi, daya tahan (ACID). QLDB menggunakan kontrol konkurensi optimis (OCC), dan transaksi beroperasi dengan serialisasi penuh - tingkat isolasi tertinggi. Ini berarti bahwa tidak ada risiko melihat pembacaan hantu, membaca kotor, menulis miring, atau masalah konkurensi serupa lainnya. Untuk informasi selengkapnya, lihat Model Konkurensi Amazon QLDB.

# Dari relasional ke buku besar

Jika Anda seorang developer aplikasi, Anda mungkin memiliki beberapa pengalaman menggunakan sistem manajemen database relasional (RDBMS) dan Structured Query Language (SQL). Ketika Anda mulai bekerja dengan Amazon QLDB, Anda akan menemukan banyak kesamaan. Ketika Anda maju ke topik yang lebih maju, Anda juga akan menemukan fitur baru yang kuat yang QLDB telah dibangun di atas dasar RDBMS. Bagian ini menjelaskan komponen database umum dan operasi, membandingkan dan mengontraskan mereka dengan setara mereka dalam QLDB.

Diagram berikut menunjukkan konstruksi pemetaan komponen inti antara RDBMS tradisional dan Amazon QLDB.




Tabel berikut menunjukkan persamaan tingkat tinggi primer dan perbedaan fitur operasional built-in antara RDBMS tradisional dan QLDB.

| Operasi | RDBMS | QLDB |
| --- | --- | --- |
| Membuat Tabel | `CREATE TABLE`pernyataan yang mendefinisikan semua nama kolom dan tipe data | `CREATE TABLE`pernyataan yang tidak mendefinisikan atribut tabel atau tipe data untuk memungkinkan schemaless dan konten terbuka |
| Membuat indeks | Pernyataan `CREATE INDEX` | `CREATE INDEX`pernyataan untuk setiap bidang tingkat atas di atas meja |
| Memasukkan data | `INSERT`pernyataan yang menentukan nilai-nilai dalam baris baru atau tuple yang | `INSERT`pernyataan yang menentukan nilai dalam dokumen baru dalam format |

| Operasi | RDBMS | QLDB |
| --- | --- | --- |
| | menganut skema seperti yang didefinisikan oleh tabel | Amazon Ion yang valid terlepas dari dokumen yang ada dalam tabel |
| Kueri data kueri | Pernyataan `SELECT-FROM-WHERE` | `SELECT-FROM-WHERE`pernyataan dalam sintaks yang sama seperti SQL ketika query dokumen datar |
| Memperbarui data | Pernyataan `UPDATE-SET-WHERE` | `UPDATE-SET-WHERE`pernyataan dalam sintaks yang sama seperti SQL saat memperbarui dokumen datar |
| Menghapus data | Pernyataan `DELETE-FROM-WHERE` | `DELETE-FROM-WHERE`pernyataan dalam sintaks yang sama seperti SQL saat menghapus dokumen datar |
| Data bersarang dan semistruk turisasi | Baris datar atau tupel saja | Dokumen yang dapat memiliki data terstruktur, semistruktural, atau bersarang sebagaimana didukung oleh format data Amazon Ion dan bahasa kueri PartiQL |
| Kueri metadata kueri | Tidak ada metadata bawaan | `SELECT`pernyataan bahwa query dari built-in pandangan berkomitmen dari tabel |
| Mengkueri riwayat revisi | Tidak ada riwayat data bawaan | `SELECT`pernyataan bahwa query dari fungsi sejarah built-in |

| Operasi | RDBMS | QLDB |
| --- | --- | --- |
| Verifikasi kriptografi | Tidak ada kriptografi bawaan atau kekekalan | API yang mengembalikan intisari jurnal dan bukti yang memverifikasi integritas revisi dokumen apa pun relatif terhadap intisari itu |

Untuk gambaran dari konsep inti dan terminologi di QLDB, lihatKonsep inti.

Untuk informasi terperinci tentang proses pembuatan, kueri, dan pengelolaan data Anda dalam buku besar, lihatBekerja dengan data dan sejarah.

# Konsep inti dan terminologi di Amazon QLDB

Bagian ini memberikan ikhtisar konsep inti dan terminologi di Amazon QLDB, termasuk struktur buku besar dan bagaimana buku besar mengelola data. Sebagai database buku besar, QLDB berbeda dari database berorientasi dokumen lainnya ketika datang ke konsep-konsep kunci berikut.

Topik



## Model objek data QLDB

Model objek data fundamental di Amazon QLDB dijelaskan sebagai berikut:

1. Buku Besar

   Langkah pertama Anda adalah membuat buku besar, yang merupakan jenisAWS sumber daya utama di QLDB. Untuk mempelajari cara membuat buku besar, lihatLangkah 1: Membuat buku besar baru di Memulai konsol, atauOperasi dasar untuk buku besar Amazon QLDB.

Untuk mode`STANDARD` perizinan`ALLOW_ALL` dan buku besar, Anda membuat kebijakanAWS Identity and Access Management (IAM) yang memberikan izin untuk menjalankan operasi API pada sumber daya buku besar ini.

Format ARN buku besar:

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

2. Jurnal dan tabel

   Untuk mulai menulis data dalam buku besar QLDB, Anda pertama kali membuat tabel dengan[CREATE TABLE](#) pernyataan dasar. Data buku besar terdiri dari revisi dokumen yang berkomitmen pada jurnal buku besar. Anda melakukan revisi dokumen ke buku besar dalam konteks tabel yang ditentukan pengguna. Dalam QLDB, tabel mewakili pandangan terwujud dari koleksi revisi dokumen dari jurnal.

   Dalam mode`STANDARD` perizinan buku besar, Anda harus membuat kebijakan IAM yang memberikan izin untuk menjalankan pernyataan PartiQL pada sumber daya tabel ini. Dengan izin pada sumber daya tabel, Anda dapat menjalankan pernyataan yang mengakses keadaan tabel saat ini. Anda juga dapat menanyakan riwayat revisi tabel dengan menggunakan`history()` fungsi bawaan.

   Format ARN tabel:

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

   Untuk informasi selengkapnya tentang pemberian izin pada buku besar dan sumber daya terkait, lihat[Bagaimana Amazon QLDB bekerja dengan IAM](#).

3. Dokumen

   Tabel terdiri dari revisi[Dokumen QLDB B B B B B B B](#), yang merupakan kumpulan data dalam`struct` format [Amazon Ion](#). Revisi dokumen mewakili satu versi dari urutan dokumen yang diidentifikasi oleh ID dokumen unik.

   QLDB menyimpan riwayat perubahan lengkap dokumen yang Anda lakukan. Sebuah tabel memungkinkan Anda query keadaan dokumen saat ini, sementara`history()` fungsi memungkinkan Anda query seluruh sejarah revisi dokumen tabel. Untuk detail tentang query dan menulis revisi, lihat[Bekerja dengan data dan sejarah](#).

4. Katalog sistem

   Setiap buku besar juga menyediakan sumber daya katalog yang ditentukan sistem yang dapat Anda kueri untuk mencantumkan semua tabel dan indeks dalam buku besar. Dalam mode`STANDARD` izin buku besar, Anda memerlukan`qldb:PartiQLSelect` izin pada sumber daya katalog ini untuk melakukan hal berikut:

   - Jalankan`SELECT` pernyataan pada tabel katalog sistem information_schema.user_tables.
   - Lihat tabel dan indeks informasi pada halaman rincian buku besar pada konsol QLDB.
   - Lihat daftar tabel dan indeks di editor PartiQL pada konsol QLDB.

   Format ARN Katalog:

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

## Transaksi jurnal-pertama

Ketika aplikasi membaca atau menulis data dalam buku besar QLDB, ia melakukannya dalam transaksi database. Semua transaksi tunduk pada batas sebagaimana didefinisikan dalamKuota dan batasan di Amazon QLDB. Dalam transaksi, QLDB melakukan langkah-langkah berikut:

1. Membaca status data saat ini dari buku besar.
2. Lakukan pernyataan yang disediakan dalam transaksi, dan kemudian memeriksa setiap konflik menggunakan kontrol konkurensi optimis (OCC) untuk memastikan isolasi sepenuhnya serializable.
3. Jika tidak ditemukan konflik OCC, kembalikan hasil transaksi sebagai berikut:
   - Untuk membaca, mengembalikan hasil set dan komit`SELECT` pernyataan ke jurnal dengan cara menambahkan saja.
   - Untuk penulisan, lakukan pembaruan, hapus, atau data yang baru disisipkan ke jurnal dengan cara khusus tambahan.

Jurnal ini mewakili riwayat yang lengkap dan tidak berubah dari semua perubahan pada data Anda. QLDB menulis satu blok dirantai ke jurnal dalam transaksi. Setiap blok berisi objek entri yang mewakili revisi dokumen yang Anda masukkan, update, dan hapus, bersama dengan pernyataan PartiQL yang melakukan mereka.

Diagram berikut menggambarkan struktur jurnal ini.

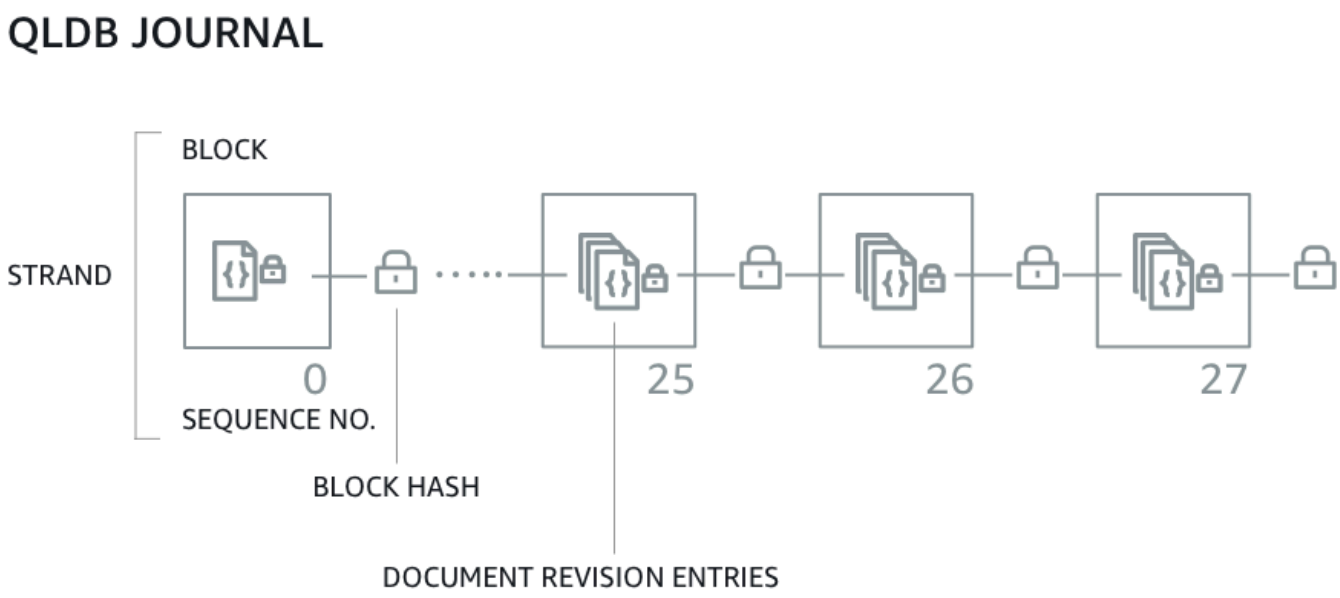



Diagram menunjukkan bahwa transaksi berkomitmen untuk jurnal sebagai blok yang berisi entri revisi dokumen. Setiap blok di-hash dan dirantai ke blok berikutnya untuk verifikasi. Setiap blok memiliki nomor urut untuk menentukan alamatnya di dalam untai.

> **Note**
>
> Di Amazon QLDB, untai adalah partisi jurnal buku besar Anda. QLDB saat ini mendukung jurnal dengan untai tunggal saja.

Untuk informasi tentang isi data dalam blok, lihatIsi jurnal di Amazon QLDB.

## Mengkueri data Anda

QLDB dimaksudkan untuk memenuhi kebutuhan beban kerja pemrosesan transaksi online (Online Transaction Processing atau OLTP). Buku besar menyediakan tampilan tabel queryable data Anda berdasarkan informasi transaksi yang berkomitmen untuk jurnal. Tampilan tabel di QLDB adalah bagian dari data dalam tabel. Tampilan dipertahankan secara real time, sehingga selalu tersedia untuk permintaan aplikasi.

Anda dapat query tampilan sistem-didefinisikan berikut menggunakan`SELECT` pernyataan PartiQL:

- Pengguna - Revisi aktif terbaru hanya data yang Anda tulis di tabel (yaitu, keadaan data pengguna Anda saat ini). Ini adalah tampilan default di QLDB.
- Committed - Revisi aktif terbaru dari data pengguna Anda dan metadata yang dihasilkan sistem. Ini adalah tabel penuh sistem didefinisikan yang sesuai langsung ke tabel pengguna Anda.

Selain tampilan kueri ini, Anda dapat query sejarah revisi data Anda dengan menggunakan built-inFungsi Riwayat. Fungsi history mengembalikan data pengguna Anda dan metadata terkait dalam skema yang sama dengan tampilan berkomitmen.

## Penyimpanan data

Ada dua tipe penyimpanan data di QLDB:

- Penyimpanan jurnal - Ruang disk yang digunakan oleh jurnal buku besar. Jurnal hanya ditambahkan dan berisi riwayat lengkap, tidak dapat diubah, dan dapat diverifikasi dari semua perubahan pada data Anda.
- Penyimpanan terindeks - Ruang disk yang digunakan oleh tabel buku besar, indeks, dan riwayat terindeks. Penyimpanan terindeks terdiri dari data buku besar yang dioptimalkan untuk kueri berkinerja tinggi.

Setelah data Anda berkomitmen untuk jurnal, itu terwujud ke dalam tabel yang Anda tentukan. Tabel ini dioptimalkan untuk kueri yang lebih cepat dan lebih efisien. Saat aplikasi menggunakan API data transaksional untuk membaca data, aplikasi mengakses tabel dan indeks yang disimpan di penyimpanan terindeks Anda.

## Model API QLDB

QLDB menyediakan dua jenis API yang dapat berinteraksi dengan kode aplikasi Anda:

- Amazon QLDB — API manajemen sumber daya QLDB (juga dikenal sebagai bidang kontrol). API ini hanya digunakan untuk mengelola sumber daya buku besar dan untuk operasi data non-transaksional. Anda dapat menggunakan operasi ini untuk membuat, menghapus, menggambarkan, membuat daftar, dan memperbarui buku besar. Anda juga dapat memverifikasi data secara kriptografis, dan mengekspor atau melakukan streaming blok jurnal.
- Sesi QLDB Amazon — API data transaksional QLDB. Anda dapat menggunakan API ini untuk menjalankan transaksi data pada buku besar dengan pernyataan PartiQL.

> **Important**
>
> Alih-alih berinteraksi langsung dengan API Sesi QLDB, sebaiknya gunakan driver QLDB atau shell QLDB untuk menjalankan transaksi data pada buku besar.
>
> - Jika Anda bekerja denganAWS SDK, gunakan driver QLDB. Driver menyediakan lapisan abstraksi tingkat tinggi di atas API data Sesi QLDB dan mengelola`SendCommand` operasi untuk Anda. Untuk informasi dan daftar bahasa pemrograman yang didukung, lihatMemulai dengan driver.
> - Jika Anda bekerja denganAWS CLI, gunakan shell QLDB. Shell adalah antarmuka baris perintah yang menggunakan driver QLDB untuk berinteraksi dengan buku besar. Untuk informasi, lihat Menggunakan shell Amazon QLDB (hanya API data).

Untuk informasi selengkapnya tentang operasi API ini, lihat bagianReferensi API Amazon QLDB.

## Langkah selanjutnya

Untuk mempelajari cara menggunakan buku besar dengan data Anda, lihatBekerja dengan data dan riwayat di Amazon QLDB dan ikuti contoh yang menjelaskan proses pembuatan tabel, memasukkan data, dan menjalankan kueri dasar. Panduan ini menjelaskan bagaimana konsep-konsep ini bekerja secara mendalam, menggunakan data sampel dan contoh kueri untuk konteks.

Untuk memulai dengan cepat dengan tutorial aplikasi contoh menggunakan konsol QLDB, lihatMemulai dengan konsol Amazon QLDB.

Untuk daftar istilah dan definisi utama yang dijelaskan di bagian ini, lihatGlosarium Amazon QLDB.

# Isi jurnal di Amazon QLDB

Di Amazon QLDB, jurnal adalah log transaksional abadi yang menyimpan riwayat lengkap dan dapat diverifikasi dari semua perubahan pada data Anda. Jurnal ini hanya ditambahkan dan terdiri dari sekumpulan blok yang diurutkan dan dirantai hash yang berisi data berkomitmen dan metadata sistem lainnya. QLDB menulis satu blok dirantai ke jurnal dalam transaksi.

Bagian ini memberikan contoh blok jurnal dengan data sampel dan menjelaskan isi blok.

Topik

- Contoh blok

- Memblokir konten blok blok blok blok
- Revisi yang disunting
- Aplikasi sampel
- Lihat juga

## Contoh blok

Sebuah blok jurnal berisi metadata transaksi bersama dengan entri yang mewakili revisi dokumen yang dilakukan dalam transaksi dan pernyataan PartiQL yang berkomitmen mereka.

Berikut ini adalah contoh blok dengan data sampel berikut.

> **Note**
>
> Contoh blok ini disediakan untuk tujuan informasi saja. Hash yang ditampilkan bukan nilai hash yang dihitung secara nyata.

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

Di`revisions` lapangan, beberapa objek revisi mungkin hanya berisi`hash` nilai dan tidak ada atribut lainnya. Ini adalah revisi sistem internal saja yang tidak berisi data pengguna. Hash dari revisi ini adalah bagian dari rantai hash penuh jurnal, yang diperlukan untuk verifikasi kriptografi.

## Memblokir konten blok blok blok blok

Blok jurnal blok memiliki bidang berikut:

**`blockAddress`**

Lokasi blok dalam jurnal. Alamat adalah struktur Amazon Ion yang memiliki dua bidang:`strandId` dan`sequenceNo`.

Misalnya: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`

**`transactionId`**

ID unik dari transaksi yang dilakukan blok.

**`blockTimestamp`**

Stempel waktu saat blok berkomitmen pada jurnal.

**`blockHash`**

Nilai hash 256-bit yang secara unik mewakili blok. Ini adalah hash dari rangkaian`entriesHash` dan`previousBlockHash`.

**`entriesHash`**

Hash yang mewakili semua entri dalam blok, termasuk entri sistem internal saja. Ini adalah akar hash dari pohon Merkle di mana node daun terdiri dari semua hash di`entriesHashList`.

**`previousBlockHash`**

Hash dari blok dirantai sebelumnya dalam jurnal.

**`entriesHashList`**

Daftar hash yang mewakili setiap entri dalam blok. Daftar ini dapat mencakup hash entri berikut:

- Ion hash yang mewakili`transactionInfo`. Nilai ini dihitung dengan mengambil hash Ion dari seluruh`transactionInfo` struktur.
- Akar hash dari pohon Merkle di mana simpul daun terdiri dari semua hash di`revisions`.
- Ion hash yang mewakili`redactionInfo`. Hash ini hanya ada di blok yang dilakukan oleh transaksi redaksi. Nilainya dihitung dengan mengambil hash Ion dari seluruh`redactionInfo` struktur.
- Hash yang mewakili metadata sistem internal saja. Hash ini mungkin tidak ada di semua blok.

**`transactionInfo`**

Struktur Amazon Ion yang berisi informasi tentang pernyataan dalam transaksi yang melakukan blok. Struktur ini memiliki bidang berikut:

- `statements`- Daftar pernyataan PartiQL dan`startTime` ketika mereka mulai berjalan. Setiap pernyataan memiliki`statementDigest` hash, yang diperlukan untuk menghitung hash`transactionInfo` struktur.
- `documents`- ID dokumen yang diperbarui oleh pernyataan. Setiap dokumen termasuk`tableName` dan`tableId` bahwa itu milik, dan indeks setiap pernyataan yang diperbarui itu.

**`revisions`**

Daftar revisi dokumen yang dilakukan di blok. Setiap struktur revisi berisi semua bidang dari pandangan revisi yang dilakukan.

Ini juga dapat mencakup hash yang mewakili revisi sistem internal saja yang merupakan bagian dari rantai hash penuh jurnal.

## Revisi yang disunting

Di Amazon QLDB,`DELETE` pernyataan hanya secara logis menghapus dokumen dengan membuat revisi baru yang menandainya sebagai dihapus. QLDB juga mendukung operasi redaksi data yang memungkinkan Anda menghapus revisi dokumen yang tidak aktif secara permanen dalam sejarah tabel.

Operasi redaksi hanya menghapus data pengguna dalam revisi yang ditentukan, dan membiarkan urutan jurnal dan metadata dokumen tidak berubah. Ini menjaga integritas data keseluruhan buku besar Anda. Untuk informasi lebih lanjut dan contoh operasi redaksi, lihatMenyunting revisi dokumen.

### Contoh revisi isi yang disunting

Pertimbangkan contoh blok sebelumnya. Di blok ini, anggaplah Anda menyunting revisi yang memiliki ID dokumen`HwVFkn8IMRa0xjze5xcgga` dan nomor versi`0`.

Setelah redaksi selesai, data pengguna dalam revisi (diwakili oleh`data` struktur) diganti dengan`dataHash` bidang baru. Nilai bidang ini adalah hash Ion dari`data` struktur yang dihapus. Akibatnya, buku besar mempertahankan integritas data secara keseluruhan dan tetap dapat diverifikasi secara kriptografis melalui operasi API verifikasi yang ada.

Contoh revisi berikut menunjukkan hasil redaksi ini, dengan`dataHash` bidang baru disorot dalam *huruf miring merah*.

> **Note**
>
> Contoh revisi ini disediakan untuk tujuan informasi saja. Hash yang ditampilkan bukan nilai hash yang dihitung secara nyata.

```
...
{
  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 juga menambahkan blok baru ke jurnal untuk permintaan redaksi selesai. Blok ini mencakup`redactionInfo` entri tambahan yang berisi daftar revisi yang disunting dalam transaksi, seperti yang ditunjukkan pada contoh berikut.

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

```
...
```

## Aplikasi sampel

Untuk contoh kode Java yang memvalidasi rantai hash jurnal menggunakan data yang diekspor, lihat GitHub repositori aws-samples/amazon-qldb-dmv-sample -java. Aplikasi contoh ini mencakup file kelas berikut:

- ValidateQldbHashChain.java - Berisi kode tutorial yang mengekspor blok jurnal dari buku besar dan menggunakan data yang diekspor untuk memvalidasi rantai hash antara blok.
- JournalBlock.java - Berisi metode bernama`verifyBlockHash()` yang menunjukkan bagaimana menghitung setiap komponen hash individu dalam blok. Metode ini disebut dengan kode tutorial di`ValidateQldbHashChain.java`.

Untuk petunjuk tentang cara untuk men-download dan menginstal aplikasi sampel lengkap ini, lihatMenginstal aplikasi sampel Amazon QLDB Java. Sebelum Anda menjalankan kode tutorial, pastikan Anda mengikuti Langkah 1-3 ditutorial java untuk menyiapkan buku besar sampel dan memuatnya dengan data sampel.

## Lihat juga

Untuk informasi lebih lanjut tentang jurnal di QLDB, lihat topik berikut:

- Mengekspor data jurnal dari Amazon QLDB— Untuk mempelajari cara mengekspor data jurnal ke Amazon Simple Storage Service (Amazon S3).
- Streaming data jurnal dari Amazon QLDB— Untuk mempelajari cara mengalirkan data jurnal ke Amazon Kinesis Data Streams.
- Verifikasi data di Amazon QLDB— Untuk mempelajari tentang verifikasi kriptografi data jurnal.

# Glosarium Amazon QLDB

Berikut ini adalah definisi untuk istilah utama yang mungkin Anda serakan saat Anda bekerja dengan Amazon QLDB.

blok | mencerna | dokumen | ID dokumen | revisi dokumen | masuk | bidang | indeks | penyimpanan terindeks | jurnal | blok jurnal | penyimpanan jurnal | untai jurnal | tip jurnal | buku besar | bukti | revisi | sesi | untai | tabel | tampilan tabel | view

blok

Sebuah objek yang berkomitmen untuk jurnal dalam transaksi. Satu transaksi menulis satu blok dalam jurnal, sehingga blok hanya dapat dikaitkan dengan satu transaksi. Sebuah blok berisi entri yang mewakili revisi dokumen yang dilakukan dalam transaksi, bersama dengan pernyataan PartiQL yang melakukan mereka.

Setiap blok juga memiliki nilai hash untuk verifikasi. Sebuah blok hash dihitung dari hash entri dalam blok yang dikombinasikan dengan hash dari blok dirantai sebelumnya.

mencerna

Nilai hash 256-bit yang secara unik mewakili seluruh riwayat revisi dokumen buku besar Anda sebagai titik waktu. Sebuah hash digest dihitung dari rantai hash penuh jurnal Anda sebagai blok berkomitmen terbaru dalam jurnal pada saat itu.

QLDB memungkinkan Anda menghasilkan digest sebagai file output yang aman. Kemudian, Anda dapat menggunakan file output itu untuk memverifikasi integritas revisi dokumen Anda relatif terhadap hash itu.

dokumen

Kumpulan data dalam`struct` format Amazon Ion yang dapat disisipkan, diperbarui, dan dihapus dalam tabel. Sebuah dokumen QLDB dapat memiliki terstruktur, semistructured, bersarang, dan skema-kurang data.

ID dokumen

Pengenal unik universal (UUID) yang diberikan QLDB untuk setiap dokumen yang Anda masukkan ke dalam tabel. ID ini adalah angka 128-bit yang diwakili dalam string alfanumerik yang dikodekan Base62 dengan panjang tetap 22 karakter.

revisi dokumen

Struktur Ion yang mewakili satu versi dari urutan dokumen yang diidentifikasi oleh ID dokumen unik. Revisi mencakup data pengguna Anda (yaitu, data yang Anda tulis dalam tabel) dan metadata yang dihasilkan sistem. Setiap revisi dikaitkan dengan tabel, dan diidentifikasi secara unik dengan kombinasi ID dokumen dan nomor versi berbasis nol.

masuk

Objek yang dimuat dalam blok. Entri mewakili revisi dokumen yang dimasukkan, diperbarui, dan dihapus dalam transaksi, bersama dengan pernyataan PartiQL yang berkomitmen mereka.

Setiap entri juga memiliki nilai hash untuk verifikasi. Sebuah hash entri dihitung dari hash revisi atau hash pernyataan dalam entri itu.

bidang

Pasangan nama-nilai yang membentuk setiap atribut dari dokumen QLDB. Nama adalah token simbol, dan nilainya tidak dibatasi.

indeks

Struktur data yang dapat Anda buat di atas meja untuk mengoptimalkan kinerja operasi pengambilan data. Untuk informasi tentang indeks dalam QLDB, lihatCREATE INDEX di referensi Amazon QLDB PartiQL.

penyimpanan terindeks

Ruang disk yang digunakan oleh tabel buku besar, indeks, dan sejarah diindeks. Penyimpanan terindeks terdiri dari data buku besar yang dioptimalkan untuk kueri berkinerja tinggi.

jurnal

Kumpulan hash-chained dari semua blok yang berkomitmen dalam buku besar Anda. Jurnal ini hanya ditambahkan dan mewakili riwayat lengkap dan tidak berubah dari semua perubahan pada data buku besar Anda.

blok jurnal

Lihat blok.

penyimpanan jurnal

Ruang disk yang digunakan oleh jurnal buku besar.

untai jurnal

Lihat untai.

tip jurnal

Blok berkomitmen terbaru dalam jurnal pada suatu titik waktu.

buku besar

Instance dari sumber daya database buku besar Amazon QLDB. Ini adalah jenisAWS sumber daya utama di QLDB. Buku besar terdiri dari penyimpanan jurnal dan penyimpanan terindeks. Setelah data buku besar berkomitmen untuk jurnal, itu tersedia untuk query dalam tabel revisi dokumen Amazon Ion.

bukti

Daftar memerintahkan 256-bit nilai hash yang QLDB kembali untuk mencerna diberikan dan revisi dokumen. Ini terdiri dari hash yang dibutuhkan oleh model pohon Merkle untuk merantai hash revisi yang diberikan ke hash digest. Anda menggunakan bukti untuk memverifikasi integritas revisi Anda relatif terhadap intisari. Untuk informasi selengkapnya, lihat Verifikasi data di Amazon QLDB.

revisi

Lihat revisi dokumen.

sesi

Objek yang mengelola informasi tentang permintaan transaksi data Anda dan tanggapan ke dan dari buku besar. Sesi aktif (yang secara aktif menjalankan transaksi) mewakili koneksi tunggal ke buku besar. QLDB mendukung satu transaksi yang berjalan secara aktif per sesi.

untai

Partisi jurnal. QLDB saat ini mendukung jurnal dengan untai tunggal saja.

tabel

Sebuah pandangan terwujud dari koleksi unordered revisi dokumen yang berkomitmen dalam jurnal buku besar.

tampilan tabel

Sebuah subset queryable dari data dalam tabel, berdasarkan transaksi berkomitmen untuk jurnal. Dalam pernyataan PartiQL, pandangan dilambangkan dengan kualifikasi awalan (dimulai dengan`_ql_`) untuk nama tabel.

Anda dapat query tampilan sistem-didefinisikan berikut menggunakan`SELECT` pernyataan:

- Pengguna - Revisi aktif terbaru hanya data yang Anda tulis di tabel (yaitu, keadaan data pengguna Anda saat ini). Ini adalah tampilan default di QLDB.
- Committed - Revisi aktif terbaru dari data pengguna Anda dan metadata yang dihasilkan sistem. Ini adalah tabel penuh sistem didefinisikan yang sesuai langsung ke tabel pengguna Anda. Misalnya: `_ql_committed_`*`TableName`*.

view

Lihat tampilan tabel.

# Mengakses Amazon QLDB

Anda dapat mengakses Amazon QLDB menggunakan AWS Management Console,AWS CLI(), AWS Command Line Interface atau QLDB API. Bagian berikut menjelaskan cara menggunakan opsi ini dan prasyarat untuk menggunakannya.

## Prasyarat

Sebelum Anda dapat mengakses QLDB, Anda harus mengatur Akun AWS jika Anda belum melakukannya.

Topik

- Mendaftar untuk Akun AWS
- Buat pengguna dengan akses administratif
- Kelola izin QLDB di IAM
- Berikan akses terprogram (opsional)

### Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimkan email konfirmasi setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

## Buat pengguna dengan akses administratif

Setelah Anda mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

   Untuk bantuan masuk dengan menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.
2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, lihat Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root (konsol) Anda di Panduan Pengguna IAM.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .
2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna.

   Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal AWS akses di Panduan AWS Sign-In Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

   Untuk petunjuknya, lihat Membuat set izin di Panduan AWS IAM Identity Center Pengguna.
2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di Panduan AWS IAM Identity Center Pengguna.

## Kelola izin QLDB di IAM

Untuk informasi tentang penggunaan AWS Identity and Access Management (IAM) untuk mengelola izin QLDB bagi pengguna, lihat. Bagaimana Amazon QLDB bekerja dengan IAM

## Berikan akses terprogram (opsional)

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar. AWS Management Console Cara untuk memberikan akses terprogram tergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses programatis, pilih salah satu opsi berikut.

| Pengguna mana yang membutuhkan akses programatis? | Untuk | Oleh |
| --- | --- | --- |
| Identitas tenaga kerja<br><br>(Pengguna yang dikelola di Pusat Identitas IAM) | Gunakan kredenal sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS | Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.<br><br>• Untuk AWS CLI, lihat Mengkonfigurasi yang akan AWS CLI digunakan AWS |

| Pengguna mana yang membutuhkan akses programatis? | Untuk | Oleh |
| --- | --- | --- |
| | | IAM Identity Center dalam Panduan AWS Command Line Interface Pengguna.<br>• Untuk AWS SDK, alat, dan AWS API, lihat autentikasi Pusat Identitas IAM di Panduan Referensi AWS SDK dan Alat. |
| IAM | Gunakan kredenal sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS | Mengikuti petunjuk dalam Menggunakan kredensil sementara dengan AWS sumber daya di Panduan Pengguna IAM. |

| Pengguna mana yang membutuhkan akses programatis? | Untuk | Oleh |
| --- | --- | --- |
| IAM | (Tidak direkomendasikan) Gunakan kredensil jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS | Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.<br>• Untuk mengetahui AWS CLI, lihat Mengautentikasi menggunakan kredensil pengguna IAM di Panduan Pengguna.AWS Command Line Interface<br>• Untuk AWS SDK dan alat bantu, lihat Mengautentikasi menggunakan kredensil jangka panjang di Panduan Referensi AWS SDK dan Alat.<br>• Untuk AWS API, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM. |

## Cara mengakses Amazon QLDB

Setelah menyelesaikan prasyarat untuk menyiapkan Akun AWS, lihat topik berikut untuk mempelajari lebih lanjut tentang cara mengakses QLDB:

- Menggunakan konsol
- Menggunakan AWS CLI (hanya API manajemen)
- Menggunakan shell Amazon QLDB (hanya API data)
- Menggunakan API

# Mengakses Amazon QLDB menggunakan konsol

Anda dapat mengakses AWS Management Console untuk Amazon QLDB di https://console.aws.amazon.com/qldb.

Anda dapat menggunakan konsol untuk melakukan hal berikut di QLDB:

- Buat, hapus, jelaskan, dan buat daftar buku besar.
- Jalankan pernyataan PartiQL dengan menggunakan editor PartiQL.
- Kelola tag untuk sumber daya QLDB.
- Verifikasi data jurnal secara kriptografis.
- Ekspor atau streaming blok jurnal.

Untuk mempelajari cara membuat buku besar QLDB Amazon dan mengaturnya dengan contoh data aplikasi, lihat. Memulai dengan konsol Amazon QLDB

## Referensi cepat editor PartiQL

Amazon QLDB mendukung subset PartiQL sebagai bahasa kueri dan Amazon Ion sebagai format data berorientasi dokumen. Untuk panduan lengkap dan informasi lebih rinci tentang implementasi QLDB dari PartiQL, lihat. Referensi PartiQLDB QLDB

Topik berikut memberikan gambaran referensi singkat tentang cara menggunakan PartiQL di QLDB.

Topik

- Tips cepat PartiQL di QLDB
- Commands
- Tampilan yang ditentukan sistem
- Aturan sintaks dasar
- Pintasan keyboard editor PartiQL

### Tips cepat PartiQL di QLDB

Berikut ini adalah ringkasan singkat tips dan praktik terbaik untuk bekerja dengan PartiQL di QLDB:

- Memahami batas konkurensi dan transaksi — Semua pernyataan, termasuk `SELECT` kueri, tunduk pada konflik dan batas transaksi kontrol konkurensi optimis (OCC), termasuk batas waktu transaksi 30 detik.
- Gunakan indeks — Gunakan indeks kardinalitas tinggi dan jalankan kueri yang ditargetkan untuk mengoptimalkan pernyataan Anda dan menghindari pemindaian tabel lengkap. Untuk mempelajari selengkapnya, lihat Mengoptimalkan kinerja kueri.
- Gunakan predikat kesetaraan - Pencarian yang diindeks memerlukan operator kesetaraan (atau). `= IN` Operator ketidaksetaraan (<,>,`LIKE`,`BETWEEN`) tidak memenuhi syarat untuk pencarian yang diindeks dan menghasilkan pemindaian tabel lengkap.
- Gunakan sambungan batin saja — QLDB hanya mendukung sambungan batin. Sebagai praktik terbaik, bergabunglah di bidang yang diindeks untuk setiap tabel yang Anda ikuti. Pilih indeks kardinalitas tinggi untuk kriteria gabungan dan predikat kesetaraan.

## Commands

QLDB mendukung perintah PartiQL berikut.

Bahasa definisi data (DDL)

| Perintah | Deskripsi |
| --- | --- |
| CREATE INDEX | Membuat indeks untuk bidang dokumen tingkat atas di atas meja. |
| CREATE TABLE | Membuat tabel. |
| DROP INDEX | Menghapus indeks dari tabel. |
| MEJA DROP | Menonaktifkan tabel yang ada. |
| TABEL UNDROP | Mengaktifkan kembali tabel yang tidak aktif. |

Bahasa manipulasi data (DML)

| Perintah | Deskripsi |
| --- | --- |
| HAPUS | Menandai dokumen aktif sebagai dihapus dengan membuat revisi akhir dokumen yang baru. |
| DARI (INSERT, HAPUS, atau SET) | Secara semantik sama seperti. UPDATE |
| SISIPKAN | Menambahkan satu atau lebih dokumen ke tabel. |
| Pilih | Mengambil data dari satu atau lebih tabel. |
| PERBARUI | Memperbarui, menyisipkan, atau menghapus elemen tertentu dalam dokumen. |

Contoh pernyataan DML

SISIPKAN

```
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
}
```

PEMBARUAN-SISIPKAN

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

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

### UPDATE-HAPUS

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

### PILIH - Subquery berkorelasi

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

### PILIH - Gabung batin

```
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')
```

### PILIH - Dapatkan ID dokumen menggunakan klausa BY

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

## Tampilan yang ditentukan sistem

QLDB mendukung tampilan tabel yang ditentukan sistem berikut.

| Tayang | Deskripsi |
| --- | --- |
| *table_name* | Tampilan pengguna default dari tabel yang menyertakan status data pengguna Anda saat ini saja. |
| _ql_committed_ *table_name* | Tampilan komit lengkap yang ditentukan sistem dari tabel yang menyertakan status saat ini dari data pengguna dan metadata yang dihasilkan sistem, seperti ID dokumen. |

| Tayang | Deskripsi |
| --- | --- |
| `history(table_name )` | Fungsi riwayat bawaan yang mengembalikan riwayat revisi lengkap tabel. |

## Aturan sintaks dasar

QLDB mendukung aturan sintaks dasar berikut untuk PartiQL.

| Karakter | Deskripsi |
| --- | --- |
| ' | Tanda kutip tunggal menunjukkan nilai string, atau nama bidang dalam struktur Amazon Ion. |
| " | Tanda kutip ganda menunjukkan pengidentifikasi yang dikutip, seperti kata cadangan yang digunakan sebagai nama tabel. |
| ` | Backticks menunjukkan nilai literal Ion. |
| . | Notasi titik mengakses nama bidang struktur induk. |
| [ ] | Tanda kurung siku mendefinisikan Ion`list`, atau menunjukkan nomor urut berbasis nol untuk daftar yang ada. |
| { } | Kurung kurawal mendefinisikan Ion. `struct` |
| << >> | Kurung sudut ganda menentukan tas PartiQL, yang merupakan koleksi yang tidak berurutan. Anda menggunakan tas untuk memasukkan beberapa dokumen ke dalam tabel. |
| Sensitivitas kasus | Semua nama objek sistem QLDB — termasuk nama bidang dan nama tabel — peka huruf besar/kecil. |

## Pintasan keyboard editor PartiQL

Editor PartiQL di konsol QLDB mendukung pintasan keyboard berikut.

| Tindakan | macOS | Windows |
| --- | --- | --- |
| Jalankan . | Cmd+Return | Ctrl+Enter |
| Komentar | Cmd+/ | Ctrl+/ |
| Jelas | Cmd+Shift+Delete | Ctrl+Shift+Delete |

# Mengakses Amazon QLDB menggunakan (hanya API manajemen AWS CLI )

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk mengontrol beberapa Layanan AWS dari baris perintah dan mengotomatiskannya melalui skrip. Anda dapat menggunakan AWS CLI untuk operasi satu kali sesuai kebutuhan. Anda juga dapat menggunakannya untuk menyematkan operasi QLDB Amazon dalam skrip utilitas.

Untuk akses CLI, Anda memerlukan ID kunci akses dan kunci akses rahasia. Gunakan kredensi sementara alih-alih kunci akses jangka panjang jika memungkinkan. Kredensi sementara mencakup ID kunci akses, kunci akses rahasia, dan token keamanan yang menunjukkan kapan kredensialnya kedaluwarsa. Untuk informasi selengkapnya, lihat Menggunakan kredensil sementara dengan AWS sumber daya di Panduan Pengguna IAM.

Untuk daftar lengkap dan contoh penggunaan semua perintah yang tersedia untuk QLDB di, lihat AWS CLI Referensi AWS CLI Perintah.

> **Note**
>
> AWS CLI Satu-satunya mendukung operasi API `qldb` manajemen yang tercantum dalamReferensi API Amazon QLDB. API ini hanya digunakan untuk mengelola sumber daya buku besar dan untuk operasi data non-transaksional.
> Untuk menjalankan transaksi data dengan `qldb-session` API menggunakan antarmuka baris perintah, lihatMengakses Amazon QLDB menggunakan shell QLDB (hanya API data).

Topik

- Menginstal dan mengkonfigurasi AWS CLI

- Menggunakan AWS CLI dengan QLDB

## Menginstal dan mengkonfigurasi AWS CLI

AWS CLI Berjalan di Linux, macOS, atau Windows. Untuk menginstal dan mengonfigurasinya, lihat petunjuk berikut di Panduan AWS Command Line Interface Pengguna:

1. Menginstal atau memperbarui versi terbaru dari AWS CLI
2. Pengaturan cepat

## Menggunakan AWS CLI dengan QLDB

Format baris perintah terdiri dari nama operasi Amazon QLDB, diikuti oleh parameter untuk operasi itu. AWS CLI Mendukung sintaks singkatan untuk nilai parameter, selain JSON.

Gunakan `help` untuk mencantumkan semua perintah yang tersedia di QLDB:

```
aws qldb help
```

Anda juga dapat menggunakan `help` untuk mendeskripsikan perintah tertentu dan mempelajari lebih lanjut tentang penggunaannya:

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

Misalnya, untuk membuat buku besar:

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

# Mengakses Amazon QLDB menggunakan shell QLDB (hanya API data)

Amazon QLDB menyediakan shell baris perintah untuk interaksi dengan API data transaksional. Dengan shell QLDB, Anda dapat menjalankan pernyataan PartiQL pada data buku besar.

Versi terbaru dari shell ini ditulis dalam Rust dan open source di awslabs/ GitHub repositoriamazon-qldb-shell pada`main` cabang default. Versi Python (v1) juga masih tersedia untuk digunakan dalam repositori yang sama di`master` cabang.

> **Note**
>
> Shell Amazon QLDB hanya mendukung API data`qldb-session` transaksional. API ini hanya digunakan untuk menjalankan pernyataan PartiQL pada buku besar QLDB.
> Untuk berinteraksi dengan operasi API`qldb` manajemen menggunakan antarmuka baris perintah, lihatMengakses Amazon QLDB menggunakan (hanya API manajemen AWS CLI ).

Alat ini tidak dimaksudkan untuk dimasukkan ke dalam aplikasi atau diadopsi untuk tujuan produksi. Tujuan dari alat ini adalah untuk membiarkan Anda cepat bereksperimen dengan QLDB dan PartiQL.

Bagian berikut menjelaskan cara memulai shell QLDB.

Topik



## Prasyarat

Sebelum memulai shell QLDB, Anda harus melakukan hal berikut:

1. Ikuti petunjukAWS pengaturan diMengakses Amazon QLDB. Hal ini termasuk skenario berikut:
    1. Daftar ke AWS.
    2. Buat pengguna dengan izin QLDB yang sesuai.
    3. Memberikan akses terprogram untuk pengembangan.
2. SiapkanAWS kredensi dan default AndaWilayah AWS. Untuk petunjuknya, lihat Dasar-dasar konfigurasi di PanduanAWS Command Line Interface Pengguna.

Untuk daftar lengkap Wilayah yang tersedia, lihat titik akhir dan kuota Amazon QLDB di bagian Referensi Umum AWS.

3. Untuk buku besar apa pun dalam mode`STANDARD` izin, buat kebijakan IAM yang memberi Anda izin untuk menjalankan pernyataan PartiQL pada tabel yang sesuai. Untuk mempelajari cara membuat kebijakan ini, lihatMemulai dengan mode izin standar di Amazon QLDB.

## Memasang shell

Untuk menginstal versi terbaru dari shell QLDB, lihat file README.md aktif GitHub. QLDB menyediakan file biner bawaan untuk Linux, macOS, dan Windows di bagian Rilis GitHub repositori.

Untuk macOS, shell terintegrasi dengan keran`aws/tap` Homebrew. Untuk menginstal shell di macOS menggunakan Homebrew, jalankan perintah berikut.

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

### Konfigurasi

Setelah instalasi, shell memuat file konfigurasi default yang terletak di`$XDG_CONFIG_HOME/qldbshell/config.ion` selama inisialisasi. Di Linux dan macOS, file ini biasanya berada di`~/.config/qldbshell/config.ion`. Jika file seperti itu tidak ada, shell berjalan dengan pengaturan default.

Anda dapat membuat`config.ion` file secara manual setelah instalasi. File konfigurasi ini menggunakan format data Amazon Ion. Berikut ini adalah contoh`config.ion` file minimal.

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

Jika`default_ledger` tidak diatur dalam file konfigurasi Anda,`--ledger` parameter diperlukan saat Anda memanggil shell. Untuk daftar lengkap opsi konfigurasi, lihat file README.md aktif GitHub.

## Memanggil shell

Untuk memanggil shell QLDB pada terminal baris perintah Anda untuk buku besar tertentu, jalankan perintah berikut. Ganti *my-example-ledger*dengan nama buku besar Anda.

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

Perintah ini terhubung ke default AndaWilayah AWS. Untuk secara eksplisit menentukan Region, Anda dapat menjalankan perintah dengan`--qldb-session-endpoint` parameter`--region` or, seperti yang dijelaskan di bagian berikut.

Setelah memohon sesi`qldb` shell, Anda dapat memasukkan jenis berikut masukan:

- Perintah shell
- Pernyataan PartiQL tunggal dalam transaksi terpisah
- Beberapa pernyataan PartiQL dalam transaksi

## Parameter shell

Untuk daftar lengkap bendera dan opsi yang tersedia untuk memohon shell, jalankan`qldb` perintah dengan`--help` bendera, sebagai berikut.

```
$ qldb --help
```

Berikut ini adalah beberapa bendera kunci dan opsi untuk`qldb` perintah. Anda dapat menambahkan parameter opsional ini untuk menggantiWilayah AWS, profil kredensyal, titik akhir, format hasil, dan opsi konfigurasi lainnya.

Pemakaian

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

BENDERA

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

Mencetak informasi bantuan.

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

Mengkonfigurasi verbositas logging. Secara default, shell log kesalahan saja. Untuk meningkatkan tingkat verbositas, ulangi argumen ini (misalnya,`-vv`). Tingkat tertinggi adalah`-vvv` yang sesuai dengan`trace` verbositas.

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

Mencetak informasi versi.

OPSI

**`-l`**,**`--ledger`** ***`LEDGER_NAMA`***

Nama buku besar untuk terhubung. Ini adalah parameter shell yang diperlukan jika`default_ledger` tidak diatur dalam`config.ion` file Anda. Dalam file ini, Anda dapat mengatur opsi tambahan, seperti Wilayah.

**`-c`**,**`--config`** ***`CONFIG_FILE`***

File tempat Anda dapat menentukan opsi konfigurasi shell apa pun. Untuk memformat detail dan daftar lengkap opsi konfigurasi, lihat file README.md aktif GitHub.

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

Format output dari hasil kueri Anda. Defaultnya adalah `ion`.

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

Lokasi profilAWS kredensial Anda untuk autentikasi.

Jika tidak disediakan, shell menggunakanAWS profil default Anda, yang terletak di`~/.aws/credentials`.

**`-r`**,**`--region`** ***`KODE WILAYAH`***

Wilayah AWSKode buku besar QLDB untuk terhubung ke. Misalnya: `us-east-1`.

Jika tidak disediakan, shell terhubung ke default AndaWilayah AWS seperti yang ditentukan dalamAWS profil Anda.

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

Endpoint`qldb-session` API untuk terhubung ke.

Untuk daftar lengkap Wilayah QLDB dan titik akhir yang tersedia, lihat titik akhir dan kuota Amazon QLDB di bagian Referensi Umum AWS.

## Referensi perintah

Setelah Anda memanggil`qldb` sesi, shell mendukung kunci berikut dan perintah database:

Tombol shell

| Key | Deskripsi fungsi |
| --- | --- |
| Enter | Menjalankan pernyataan. |
| Escape+Enter (MacOS, Linux)<br><br>Shift+Enter (Jendela) | Mulai baris baru untuk memasukkan pernyataan yang mencakup beberapa baris. Anda juga dapat menyalin teks input dengan beberapa baris dan menempelkannya ke dalam shell.<br><br>Untuk petunjuk tentang pengaturan,Option bukanEscape sebagai kunci Meta di macOS, lihat situs OS X Daily. |
| Ctrl+C | Membatalkan perintah saat ini. |
| Ctrl+D | Sinyal akhir file (EOF) dan keluar tingkat shell. Jika tidak dalam transaksi aktif, keluar shell. Dalam transaksi aktif, membatalkan transaksi. |

Perintah basis data shell

| Perintah | Deskripsi fungsi |
| --- | --- |
| `help` | Menampilkan informasi bantuan. |
| `begin` | Memulai transaksi. |
| `start transaction` | |

| Perintah | Deskripsi fungsi |
| --- | --- |
| `commit` | Melakukan transaksi Anda ke jurnal buku besar. |
| `abort` | Menghentikan transaksi Anda dan menolak setiap perubahan yang Anda buat. |
| `exit` | Keluar shell. |
| `quit` | |

> **Note**
>
> Semua perintah shell QLDB tidak peka huruf besar/kecil.

## Menjalankan pernyataan individual

Kecuali untuk perintah database dan perintah meta shell yang tercantum dalam README.md, shell menafsirkan setiap perintah yang Anda masukkan sebagai pernyataan PartiQL terpisah. Secara default, shell mengaktifkan`auto-commit` mode. Mode ini dapat dikonfigurasi.

Dalam`auto-commit` mode, shell secara implisit menjalankan setiap pernyataan dalam transaksinya sendiri dan secara otomatis melakukan transaksi jika tidak ada kesalahan yang ditemukan. Ini berarti bahwa Anda tidak perlu menjalankan`start transaction` (atau`begin`) dan`commit` secara manual setiap kali Anda menjalankan pernyataan.

## Mengelola transaksi

Atau, shell QLDB memungkinkan Anda mengontrol transaksi secara manual. Anda dapat menjalankan beberapa pernyataan dalam transaksi secara interaktif, atau non-interaktif dengan mengelompokkan perintah dan pernyataan secara berurutan.

### Transaksi interaktif

Untuk menjalankan transaksi interaktif, lakukan langkah-langkah berikut.

1. Untuk memulai transaksi, masukkan`begin` perintah.

```
qldb> begin
```

Setelah Anda memulai transaksi, shell menampilkan prompt perintah berikut.

```
qldb *>
```

2. Kemudian, setiap pernyataan yang Anda masukkan berjalan dalam transaksi yang sama.
   - Misalnya, Anda dapat menjalankan pernyataan tunggal sebagai berikut.

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

     Setelah Anda menekanEnter, shell menampilkan hasil pernyataan.
   - Anda juga dapat memasukkan beberapa pernyataan atau perintah dipisahkan oleh titik koma (`;`) pembatas sebagai berikut.

     ```
     qldb *> SELECT * FROM Vehicle WHERE VIN = '1N4AL11D75C109151'; commit
     ```
3. Untuk mengakhiri transaksi, masukkan salah satu perintah berikut.
   - Masukkan`commit` perintah untuk melakukan transaksi Anda ke jurnal buku besar.

     ```
     qldb *> commit
     ```
   - Masukkan`abort` perintah untuk menghentikan transaksi Anda dan menolak perubahan yang Anda buat.

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

Batas batas waktu transaksi

Transaksi interaktif mematuhi batas waktu transaksi QLDB. Jika Anda tidak melakukan transaksi dalam waktu 30 detik sejak memulainya, QLDB secara otomatis akan kedaluwarsa transaksi dan menolak setiap perubahan yang dilakukan selama transaksi.

Kemudian, alih-alih menampilkan hasil pernyataan, shell menampilkan pesan kesalahan kedaluwarsa dan kembali ke prompt perintah normal. Untuk mencoba lagi, Anda harus memasukkan`begin` perintah lagi untuk memulai transaksi baru.

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

### Transaksi non-interaktif

Anda dapat menjalankan transaksi lengkap dengan beberapa pernyataan dengan mengelompokkan perintah dan pernyataan secara berurutan sebagai berikut.

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

Anda harus memisahkan setiap perintah dan pernyataan dengan titik koma (`;`) pembatas. Jika ada pernyataan dalam transaksi yang tidak valid, shell secara otomatis menolak transaksi. Shell tidak melanjutkan dengan pernyataan berikutnya yang Anda masukkan.

Anda juga dapat mengatur beberapa transaksi.

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

Mirip dengan contoh sebelumnya, jika transaksi gagal, shell tidak melanjutkan transaksi atau pernyataan berikutnya yang Anda masukkan.

Jika Anda tidak mengakhiri transaksi, shell beralih ke mode interaktif dan meminta Anda untuk perintah atau pernyataan berikutnya.

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

## Keluar shell

Untuk keluar dari sesi`qldb` shell saat ini, masukkan`quit` perintah`exit` atau, atau gunakan pintasan keyboardCtrl +D saat shell tidak dalam transaksi.

```
qldb> exit
$
```

```
qldb> quit
$
```

### Contoh

Untuk informasi tentang menulis pernyataan PartiQL di QLDB, lihatReferensi PartiQLDB QLDB.

Example

Contoh berikut menunjukkan urutan perintah dasar.

> **Note**
>
> Shell QLDB menjalankan setiap pernyataan PartiQL dalam contoh ini dalam transaksi sendiri. Contoh ini mengasumsikan bahwa buku besar`test-ledger` sudah ada dan aktif.

```
$ 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
```

## Mengakses Amazon QLDB menggunakan API

Anda dapat menggunakan AWS Management Console dan AWS Command Line Interface (AWS CLI) untuk bekerja secara interaktif dengan Amazon QLDB. Namun, untuk mendapatkan hasil maksimal dari QLDB, Anda dapat menulis kode aplikasi dengan driver QLDB atau SDK untuk berinteraksi dengan buku besar Anda AWS menggunakan API.

Driver memungkinkan aplikasi Anda berinteraksi dengan QLDB menggunakan API data transaksional. AWS SDK mendukung interaksi dengan API manajemen sumber daya QLDB. Untuk informasi selengkapnya tentang API ini, lihatReferensi API Amazon QLDB.

Driver menyediakan dukungan untuk QLDB di Java, .NET, Go, Node.js, dan Python. Untuk segera memulai dengan bahasa ini, lihat Memulai dengan driver Amazon QLDB.

Sebelum Anda dapat menggunakan driver QLDB atau SDK dalam aplikasi Anda, Anda harus AWS memberikan akses terprogram. Untuk informasi selengkapnya, lihat Memberikan akses programatis.

# Memulai dengan konsol Amazon QLDB

Tutorial ini memandu Anda melalui langkah-langkah untuk membuat buku besar Amazon QLDB pertama Anda dan mengisinya dengan tabel dan data sampel. Buku besar sampel yang Anda buat dalam skenario ini adalah database untuk aplikasi departemen kendaraan bermotor (DMV) yang melacak informasi historis lengkap tentang pendaftaran kendaraan.

Sejarah aset adalah kasus penggunaan umum untuk QLDB karena melibatkan berbagai skenario dan operasi yang menyoroti kegunaan database buku besar. Dengan QLDB Anda dapat langsung mengakses, query, dan memverifikasi sejarah lengkap perubahan data Anda dalam database berorientasi dokumen yang mendukung kemampuan query SQL-seperti.

Saat Anda mengerjakan tutorial ini, topik berikut menjelaskan cara menambahkan registrasi kendaraan, memodifikasinya, dan melihat riwayat perubahan pada pendaftaran tersebut. Panduan ini juga menunjukkan kepada Anda cara memverifikasi dokumen pendaftaran secara kriptografis, dan menyimpulkan dengan membersihkan sumber daya dan menghapus buku besar sampel.

Setiap langkah dalam tutorial memiliki instruksi untuk menggunakanAWS Management Console.

Topik



## Prasyarat dan pertimbangan tutorial

Sebelum memulai tutorial Amazon QLDB ini, pastikan Anda menyelesaikan prasyarat berikut ini:

1. Ikuti petunjukAWS pengaturan diMengakses Amazon QLDB, jika Anda belum melakukannya. Langkah-langkah ini termasuk mendaftarAWS dan membuat pengguna administratif.

2. Ikuti petunjukMengatur izin untuk menyiapkan izin IAM untuk sumber daya QLDB. Untuk menyelesaikan semua langkah di tutorial ini, Anda perlu akses administratif penuh ke sumber daya buku besar Anda melaluiAWS Management Console.

   > **Note**
   >
   > Jika sudah masuk sebagai pengguna dengan izinAWS administratif penuh, Anda dapat melewati langkah ini.

3. (Opsional) QLDB mengenkripsi data saat istirahat menggunakan kunci inAWS Key Management Service (AWS KMS). Anda dapat memilih salah satu jenis berikutAWS KMS keys:

   - AWSdimiliki kunci KMS - Gunakan kunci KMS yang dimiliki dan dikelola olehAWS atas nama Anda. Ini adalah pilihan default dan tidak memerlukan pengaturan tambahan.
   - Kunci KMS yang dikelola pelanggan — Gunakan kunci KMS enkripsi simetris di akun yang Anda buat, miliki, dan kelola. QLDB tidak mendukung kunci asimetris.

     Opsi ini mengharuskan Anda untuk membuat kunci KMS atau menggunakan kunci yang ada di akun Anda. Untuk petunjuk tentang cara membuat kunci yang dikelola pelanggan, lihat Membuat kunci KMS enkripsi simetris di PanduanAWS Key Management Service Pengembang.

     Anda dapat menentukan kunci KMS yang dikelola pelanggan dengan menggunakan ID, alias, atau Amazon Resource Name (ARN). Untuk mempelajari lebih lanjut, lihat Pengidentifikasi kunci (KeyId) di PanduanAWS Key Management Service Developer.

     > **Note**
     >
     > Kunci lintas wilayah tidak didukung. Kunci KMS yang ditentukan harusWilayah AWS sama dengan buku besar Anda.

## Mengatur izin

Pada langkah ini, Anda mengatur izin akses penuh melalui konsol untuk semua sumber daya QLDB diAkun AWS. Untuk memberikan izin ini dengan cepat, gunakan kebijakanAWS terkelola AmazonQLDBConsoleFullAccess.

Untuk menyediakan akses, tambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup diAWS IAM Identity Center:

  Buat set izin. Ikuti petunjuk di Buat set izin di PanduanAWS IAM Identity Center Pengguna.

- Pengguna yang dikelola dalam IAM melalui penyedia identitas:

  Membuat peran untuk federasi identitas. Ikuti petunjuk dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) di Panduan Pengguna IAM.

- Pengguna IAM:
  - Buat peran yang dapat diasumsikan pengguna Anda. Ikuti petunjuk dalam Membuat peran untuk pengguna IAM di Panduan Pengguna IAM.
  - (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk dalam Menambahkan izin ke pengguna (konsol) di Panduan Pengguna IAM.

> **Important**
>
> Untuk tujuan tutorial ini, Anda memberikan akses administratif penuh ke semua sumber daya QLDB. Untuk kasus penggunaan produksi, ikuti praktik terbaik keamanan untuk memberikan hak istimewa paling rendah, atau hanya memberikan izin yang diperlukan untuk melakukan tugas. Sebagai contoh, lihat Contoh kebijakan berbasis identitas untuk Amazon QLDB.

Untuk membuat buku besar bernama`vehicle-registration`, lanjutkan keLangkah 1: Membuat buku besar baru.

## Langkah 1: Membuat buku besar baru

Pada langkah ini, Anda membuat buku besar Amazon QLDB baru bernama`vehicle-registration`. Kemudian, Anda mengkonfirmasi bahwa status buku besar adalah Aktif. Anda juga dapat memverifikasi tag apa pun yang Anda tambahkan ke buku besar.

Saat Anda membuat buku besar, perlindungan penghapusan diaktifkan secara default. Perlindungan penghapusan adalah fitur di QLDB yang mencegah buku besar dihapus oleh pengguna manapun. Anda dapat menonaktifkan perlindungan penghapusan ketika Anda membuat buku besar menggunakan QLDB API atauAWS Command Line Interface (AWS CLI).

Untuk membuat buku besar baru

1. Masuk keAWS Management Console, dan buka konsol Amazon QLDB di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih Memulai.
3. Pada Buat kartu buku besar pertama Anda, pilih Buat Buku Besar.
4. Pada halaman Create Ledger, lakukan hal berikut:
   - Informasi buku besar - Nama Ledger harus diisi sebelumnya**`vehicle-registration`**.
   - Mode izin — Mode izin untuk ditetapkan ke buku besar. Pilih salah satu opsi berikut:
     - Izinkan semua — Mode izin warisan yang memungkinkan kontrol akses dengan rincian tingkat API untuk buku besar.

       Mode ini memungkinkan pengguna yang memiliki izin API `SendCommand` untuk buku besar ini untuk menjalankan semua perintah PartiQL (maka, `ALLOW_ALL`) pada setiap tabel dalam buku besar yang ditentukan. Mode ini mengabaikan setiap kebijakan izin IAM tingkat tabel atau tingkat perintah yang Anda buat untuk buku besar.
     - Standar — (Direkomendasikan) Mode perizinan yang memungkinkan kontrol akses dengan rincian yang lebih halus untuk buku besar, tabel, dan perintah PartiQL. Kami sangat merekomendasikan untuk menggunakan mode izin ini untuk memaksimalkan keamanan data buku besar Anda.

       Secara default, mode ini menyangkal semua permintaan untuk menjalankan perintah PartiQL pada setiap tabel dalam buku besar ini. Untuk mengizinkan perintah PartiQL, Anda harus membuat kebijakan izin IAM untuk sumber daya tabel tertentu dan tindakan PartiQL, selain izin`SendCommand` API untuk buku besar. Untuk informasi, lihat Memulai dengan mode izin standar di Amazon QLDB.
   - Enkripsi data saat istirahat - Kunci diAWS Key Management Service (AWS KMS) yang digunakan untuk enkripsi data saat istirahat. Pilih salah satu opsi berikut:
     - Gunakan kunci KMS yangAWS dimiliki - Gunakan kunci KMS yang dimiliki dan dikelola olehAWS atas nama Anda. Ini adalah pilihan default dan tidak memerlukan pengaturan tambahan.
     - PilihAWS KMS kunci yang berbeda — Gunakan kunci KMS enkripsi simetris di akun yang Anda buat, miliki, dan kelola.

Untuk membuat kunci baru dengan menggunakanAWS KMS konsol, pilih BuatAWS KMS kunci. Untuk informasi selengkapnya, lihat Membuat kunci KMS enkripsi simetris di PanduanAWS Key Management Service Pengembang.

Untuk menggunakan kunci KMS yang ada, pilih salah satu dari daftar dropdown atau tentukan ARN kunci KMS.

- Tag — (Opsional) Tambahkan metadata ke buku besar dengan melampirkan tag sebagai pasangan nilai kunci. Anda dapat menambahkan tag ke buku besar untuk membantu mengatur dan mengidentifikasi buku besar tersebut. Untuk informasi selengkapnya, lihat Pemberian tag pada sumber daya Amazon QLDB.

  Pilih Tambahkan tag, lalu masukkan pasangan kunci-nilai apa pun yang sesuai.

5. Jika pengaturan sesuai keinginan Anda, pilih Buat buku besar.

> **Note**
>
> Anda dapat mengakses buku besar QLDB Anda ketika statusnya menjadi Aktif. Ini dapat memakan waktu beberapa menit.

6. Dalam daftar Buku Besar, cari`vehicle-registration` dan konfirmasikan bahwa status buku besar adalah Aktif.
7. (Opsional) Pilih nama`vehicle-registration` buku besar. Pada halaman detail buku besar registrasi kendaraan, konfirmasikan bahwa tag apa pun yang Anda tambahkan ke buku besar muncul di kartu Tag. Anda juga dapat mengedit tag buku besar Anda menggunakan halaman konsol ini.

Untuk membuat tabel di`vehicle-registration` buku besar, lanjutkan keLangkah 2: Buat tabel, indeks, dan data sampel dalam buku besar.

## Langkah 2: Buat tabel, indeks, dan data sampel dalam buku besar

Saat buku besar Amazon QLDB Anda aktif, Anda dapat mulai membuat tabel untuk data tentang kendaraan, pemiliknya, dan informasi pendaftarannya. Setelah membuat tabel dan indeks, Anda dapat memuatnya dengan data.

Pada langkah ini, Anda membuat empat tabel di`vehicle-registration` buku besar:

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

Anda juga membuat indeks berikut.

| Nama tabel | Bidang |
|---|---|
| `VehicleRegistration` | `VIN` |
| `VehicleRegistration` | `LicensePlateNumber` |
| `Vehicle` | `VIN` |
| `Person` | `GovId` |
| `DriversLicense` | `LicensePlateNumber` |
| `DriversLicense` | `PersonId` |

Anda dapat menggunakan konsol QLDB untuk secara otomatis membuat tabel ini dengan indeks dan memuatnya dengan data sampel. Atau, Anda dapat menggunakan editor PartiQL di konsol untuk menjalankan setiap pernyataan PartiQL secara manual step-by-step.

## Opsi otomatis

Untuk membuat tabel, indeks, dan data sampel

1. Buka konsol Amazon QLDB di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih Memulai.
3. Di bawah opsi Otomatis pada kartu data aplikasi Sampel, pilih`vehicle-registration` dalam daftar buku besar.
4. Pilih Muat data sampel.

   Jika operasi selesai dengan sukses, konsol akan menampilkan pesan Sampel data dimuat.

Script ini menjalankan semua pernyataan dalam satu transaksi. Jika ada bagian dari transaksi gagal, setiap pernyataan digulung kembali, dan pesan kesalahan yang sesuai ditampilkan. Anda dapat mencoba kembali operasi setelah mengatasi masalah apa pun.

> **Note**
>
> - Salah satu kemungkinan penyebab kegagalan transaksi adalah mencoba membuat tabel duplikat. Permintaan Anda untuk memuat data sampel akan gagal jika salah satu nama tabel berikut sudah ada di buku besar Anda:`VehicleRegistration`,`Vehicle`,`Person`, dan`DriversLicense`.
>
>   Sebagai gantinya, coba muat data sampel ini dalam buku besar kosong.
> - Script ini menjalankan`INSERT` pernyataan parameterisasi. Jadi, pernyataan PartiQL ini dicatat dalam blok jurnal Anda dengan parameter mengikat bukan data literal. Misalnya, Anda mungkin melihat pernyataan berikut di blok jurnal, di mana tanda tanya (?) adalah pengganti variabel untuk isi dokumen.
>
>   ```
>   INSERT INTO Vehicle ?
>   ```

## Opsi manual

Anda memasukkan dokumen ke dalam`VehicleRegistration` dengan`PrimaryOwner` bidang kosong, dan ke`DriversLicense` dengan`PersonId` bidang kosong. Kemudian, Anda mengisi bidang ini dengan dokumen yang ditugaskan sistem`id` dari`Person` tabel.

> **Tip**
>
> Sebagai praktik terbaik, gunakan bidang`id` metadata dokumen ini sebagai kunci asing. Untuk informasi selengkapnya, lihat Melakukan Kueri Metadata Dokumen.

Untuk membuat tabel, indeks, dan data sampel

1. Buka konsol Amazon QLDB di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih editor PartiQL.

3. Pilihvehicle-registration buku besar.

4. Mulailah dengan membuat empat tabel. QLDB mendukung konten terbuka dan tidak menegakkan skema, sehingga Anda tidak menentukan atribut atau tipe data.

   Di jendela editor kueri, masukkan pernyataan berikut ini, lalu pilih Run (Jalankan). Untuk menjalankan pernyataan, Anda juga dapat menggunakan pintasanCtrlEnter keyboard+untuk Windows, atauCmd +Return untuk macOS. Untuk pintasan keyboard lainnya, lihatPintasan keyboard editor PartiQL.

   ```
   CREATE TABLE VehicleRegistration
   ```

   Ulangi langkah ini untuk masing-masing hal berikut ini.

   ```
   CREATE TABLE Vehicle
   ```

   ```
   CREATE TABLE Person
   ```

   ```
   CREATE TABLE DriversLicense
   ```

5. Selanjutnya, buat indeks yang mengoptimalkan kinerja kueri untuk setiap tabel.

   > **Important**
   >
   > QLDB membutuhkan indeks untuk secara efisien mencari dokumen. Tanpa indeks, QLDB perlu melakukan pemindaian meja penuh saat membaca dokumen. Hal ini dapat menyebabkan masalah kinerja pada tabel besar, termasuk konflik konkurensi dan batas waktu transaksi.
   > Untuk menghindari pemindaian tabel, Anda harus menjalankan pernyataan dengan klausaWHERE predikat menggunakan operator kesetaraan (=orIN) pada bidang yang diindeks atau ID dokumen. Untuk informasi selengkapnya, lihat Mengoptimalkan kinerja kueri.

   Di jendela editor kueri, masukkan pernyataan berikut ini, lalu pilih Run (Jalankan).

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

Ulangi langkah ini untuk hal berikut ini.

```
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. Setelah membuat indeks Anda, Anda dapat mulai memuat data ke tabel Anda. Pada langkah ini, masukkan dokumen ke dalam`Person` tabel dengan informasi pribadi tentang pemilik kendaraan yang dilacak buku besar.

   Di jendela editor kueri, masukkan pernyataan berikut ini, lalu pilih Run (Jalankan).

```
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. Kemudian, isi`DriversLicense` tabel dengan dokumen yang menyertakan informasi SIM untuk setiap pemilik kendaraan.

   Di jendela editor kueri, masukkan pernyataan berikut ini, lalu pilih Run (Jalankan).

```
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. Sekarang, isi`VehicleRegistration` meja dengan dokumen registrasi kendaraan. Dokumen-dokumen ini termasuk`Owners` struktur bersarang yang menyimpan pemilik primer dan sekunder.

   Di jendela editor kueri, masukkan pernyataan berikut ini, lalu pilih Run (Jalankan).

```
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. Terakhir, isi`Vehicle` tabel dengan dokumen yang menggambarkan kendaraan yang terdaftar di buku besar Anda.

   Di jendela editor kueri, masukkan pernyataan berikut ini, lalu pilih Run (Jalankan).

```
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'
} >>
```

Selanjutnya, Anda dapat menggunakan`SELECT` pernyataan untuk membaca data dari tabel di`vehicle-registration` buku besar. Lanjut ke Langkah 3: Membuat Kueri tabel di buku besar.

## Langkah 3: Membuat Kueri tabel di buku besar

Setelah membuat tabel di buku besar Amazon QLDB dan memuatnya dengan data, Anda dapat menjalankan kueri untuk meninjau data registrasi kendaraan yang baru saja Anda masukkan. QLDB menggunakan PartiQL sebagai bahasa kueri dan Amazon Ion sebagai model data berorientasi dokumen.

PartiQL adalah open-source, bahasa query SQL-kompatibel yang telah diperluas untuk bekerja dengan Ion. Dengan PartiQL, Anda dapat memasukkan, query, dan mengelola data Anda dengan operator SQL akrab. Amazon Ion adalah superset dari JSON. Ion adalah open-source, format data berbasis dokumen yang memberi Anda fleksibilitas menyimpan dan memproses data terstruktur, semistruktural, dan bersarang.

Pada langkah ini, Anda menggunakan`SELECT` pernyataan untuk membaca data dari tabel di`vehicle-registration` buku besar.

> **Warning**
>
> Ketika Anda menjalankan query di QLDB tanpa pencarian diindeks, memanggil scan tabel penuh. PartiQL mendukung query tersebut karena SQL kompatibel. Namun, jangan jalankan pemindaian tabel untuk kasus penggunaan produksi di QLDB. Pemindaian tabel dapat menyebabkan masalah kinerja pada tabel besar, termasuk konflik konkurensi dan batas waktu transaksi.
>
> Untuk menghindari pemindaian tabel, Anda harus menjalankan pernyataan dengan klausa`WHERE` predikat menggunakan operator kesetaraan pada bidang yang diindeks atau

ID dokumen; misalnya,`WHERE indexedField = 123` atau`WHERE indexedField IN (456, 789)`. Untuk informasi selengkapnya, lihat Mengoptimalkan kinerja kueri.

Untuk query tabel

1. Buka konsol Amazon QLDB di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih editor PartiQL.
3. Pilih`vehicle-registration` buku besar.
4. Di jendela editor kueri, masukkan pernyataan berikut untuk menanyakan`Vehicle` tabel nomor identifikasi kendaraan tertentu (VIN) yang Anda tambahkan ke buku besar, lalu pilih Jalankan.

   Untuk menjalankan pernyataan, Anda juga dapat menggunakan pintasanCtrlEnter keyboard +untuk Windows, atauCmd +Return untuk macOS. Untuk pintasan keyboard lainnya, lihatPintasan keyboard editor PartiQL.

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

5. Anda dapat menulis kueri gabungan batin. Contoh query ini bergabung`Vehicle` dengan`VehicleRegistration` dan mengembalikan informasi pendaftaran bersama dengan atribut dari kendaraan terdaftar untuk ditentukan`VIN`.

   Masukkan pernyataan berikut, dan kemudian pilih Jalankan.

   ```
   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
   ```

   Anda juga dapat bergabung dengan`Person` dan`DriversLicense` tabel untuk melihat atribut yang terkait dengan driver yang ditambahkan ke buku besar.

   Ulangi langkah ini untuk hal berikut ini.

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

Untuk mempelajari tentang memodifikasi dokumen dalam tabel di`vehicle-registration` buku besar, lihatLangkah 4: Pemodifikasian dokumen di buku besar.

# Langkah 4: Pemodifikasian dokumen di buku besar

Sekarang setelah Anda memiliki data untuk digunakan, Anda dapat mulai membuat perubahan pada dokumen dalam`vehicle-registration` buku besar di Amazon QLDB. Misalnya, pertimbangkan Audi A5 dengan `VIN1N4AL11D75C109151`. Mobil ini awalnya dimiliki oleh sopir bernama Raul Lewis di Seattle, WA.

Misalkan Raul menjual mobil ke penduduk di Everett, WA bernama Brent Logan. Kemudian, Brent dan Alexis Pena memutuskan untuk menikah. Brent ingin menambahkan Alexis sebagai pemilik sekunder pada pendaftaran. Pada langkah ini, pernyataan bahasa manipulasi data (DMS) berikut menunjukkan bagaimana membuat perubahan yang sesuai dalam buku besar Anda untuk mencerminkan peristiwa ini.

> **Tip**
>
> Sebagai praktik terbaik, gunakan sistem dokumen yang ditugaskan`id` sebagai kunci asing. Meskipun Anda dapat menentukan bidang yang dimaksudkan untuk menjadi pengidentifikasi unik (misalnya, VIN kendaraan), pengenal unik sebenarnya dari dokumen adalah`id`. Bidang ini termasuk dalam metadata dokumen, yang dapat Anda kueri dalam tampilan berkomitmen (tampilan tabel yang ditentukan sistem).
> Untuk informasi lebih lanjut tentang tampilan di QLDB, lihatKonsep inti. Untuk mempelajari selengkapnya tentang metadata, lihatMelakukan Kueri Metadata Dokumen.

Untuk mengubah dokumen

1. Buka konsol Amazon QLDB di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih editor PartiQL.
3. Pilih`vehicle-registration` buku besar.

> **Note**
>
> Jika Anda menyiapkan buku besar menggunakan fitur Load sample data otomatis konsol, lanjutkan ke langkah 6.

4. Jika Anda menjalankan`INSERT` pernyataan secara manual untuk memuat data sampel, lanjutkan dengan langkah-langkah ini.

   Untuk awalnya mendaftarkan Raul sebagai pemilik kendaraan ini, mulailah dengan menemukan dokumen yang ditugaskan sistem`id` di`Person` tabel. Bidang ini termasuk dalam metadata dokumen, yang dapat Anda kueri dalam tampilan tabel yang ditentukan sistem, yang disebut tampilan berkomitmen.

   Di jendela editor kueri, masukkan pernyataan berikut ini, lalu pilih Run (Jalankan).

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

   Awalan`_ql_committed_` adalah awalan reserved menandakan bahwa Anda ingin query pandangan berkomitmen dari`Person` tabel. Dalam tampilan ini, data Anda bersarang di`data` bidang, dan metadata bersarang di`metadata` bidang.

5. Sekarang, gunakan ini`id` dalam sebuah`UPDATE` pernyataan untuk memodifikasi dokumen yang sesuai dalam`VehicleRegistration` tabel. Masukkan pernyataan berikut, dan kemudian pilih Jalankan.

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

   Konfirmasikan bahwa Anda memodifikasi`Owners` bidang dengan mengeluarkan pernyataan ini.

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

6. Untuk mentransfer kepemilikan kendaraan ke Brent di kota Everett, pertama-tama temukan miliknya`id` dari`Person` tabel dengan pernyataan berikut.

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

   Selanjutnya, gunakan ini`id` untuk memperbarui`PrimaryOwner` dan`City` di`VehicleRegistration` tabel.

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

Konfirmasikan bahwa Anda memodifikasi`PrimaryOwner` dan`City` bidang dengan mengeluarkan pernyataan ini.

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

7. Untuk menambahkan Alexis sebagai pemilik sekunder mobil, temukan dia`Person id`.

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

Kemudian, masukkan ini`id` ke dalam`SecondaryOwners` daftar dengan pernyataan DML-INSERT berikut.

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

Konfirmasikan bahwa Anda diubah`SecondaryOwners` dengan mengeluarkan pernyataan ini.

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

Untuk meninjau perubahan ini dalam`vehicle-registration` buku besar, lihatLangkah 5: Melihat riwayat revisi untuk sebuah dokumen.

# Langkah 5: Melihat riwayat revisi untuk sebuah dokumen

Setelah memodifikasi data pendaftaran untuk mobil dengan VIN `1N4AL11D75C109151`, Anda dapat menanyakan riwayat semua pemilik terdaftar dan bidang terbaru lainnya. Anda dapat melihat semua revisi dokumen yang Anda masukkan, diperbarui, dan dihapus dengan query built-inFungsi Riwayat.

Fungsi histori mengembalikan revisi dari tampilan berkomitmen tabel Anda, yang mencakup data aplikasi dan metadata terkait. Metadata menunjukkan dengan tepat kapan setiap revisi dibuat, dalam urutan apa, dan transaksi mana yang dilakukan.

Pada langkah ini, Anda meminta riwayat revisi dokumen dalam`VehicleRegistration` tabel di`vehicle-registration` buku besar.

Untuk melihat riwayat revisi

1. Buka konsol Amazon QLDB di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih editor PartiQL.
3. Pilih`vehicle-registration` buku besar.
4. Untuk menanyakan riwayat dokumen, mulailah dengan menemukan keunikannya`id`. Selain query tampilan berkomitmen, cara lain untuk mendapatkan dokumen`id` adalah dengan menggunakanBY kata kunci dalam tampilan pengguna default tabel. Untuk mempelajari selengkapnya, lihat Menggunakan klausa BY untuk query ID dokumen.

   Di jendela editor kueri, masukkan pernyataan berikut, lalu pilih Run (Jalankan).

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

5. Selanjutnya, Anda dapat menggunakan`id` nilai ini untuk menanyakan fungsi riwayat. Masukkan pernyataan berikut, dan kemudian pilih Jalankan. Pastikan untuk menggant`id` nilai dengan ID dokumen Anda sendiri yang sesuai.

   ```
   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**
>
> Untuk tujuan tutorial ini, permintaan sejarah ini mengembalikan semua revisi ID dokumen`ADR2LQq48kB9neZDupQrMm`. Namun, sebagai praktik terbaik, kualifikasi kueri riwayat dengan ID dokumen dan rentang tanggal (waktu mulai dan waktu akhir). Dalam QLDB, setiap`SELECT` permintaan diproses dalam transaksi dan tunduk pada batas batas waktu transaksi. Kueri riwayat yang menyertakan waktu mulai dan waktu akhir mendapatkan manfaat kualifikasi rentang tanggal. Untuk informasi selengkapnya, lihat Fungsi Riwayat.

Fungsi sejarah mengembalikan dokumen dalam skema yang sama dengan tampilan berkomitmen. Contoh ini memproyeksikan data registrasi kendaraan Anda yang dimodifikasi. Outputnya akan terlihat serupa dengan yang berikut ini:

| VIN | Kota | Pemilik |
|---|---|---|
| `"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**
>
> Kueri sejarah mungkin tidak selalu mengembalikan revisi dokumen dalam urutan berurutan.

Tinjau output dan konfirmasikan bahwa perubahan mencerminkan apa yang Anda lakukanLangkah 4: Pemodifikasian dokumen di buku besar.

6. Kemudian, Anda dapat memeriksa metadata dokumen dari setiap revisi. Masukkan pernyataan berikut, dan kemudian pilih Jalankan. Sekali lagi, pastikan untuk menggantiid nilai dengan ID dokumen Anda sendiri yang sesuai.

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

Outputnya akan terlihat serupa dengan yang berikut ini:

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

Bidang metadata ini memberikan detail tentang kapan setiap item diubah, dan transaksi mana. Dari data ini, Anda dapat menyimpulkan hal berikut:

- Dokumen ini diidentifikasi secara unik oleh sistem-ditugaskan`id`:`ADR2LQq48kB9neZDupQrMm`. Ini adalah identifier universal unik (UUID) yang diwakili dalam string Base62 dikodekan.
- `txTime`Menunjukkan bahwa revisi awal dokumen (versi`0`) dibuat di`2019-05-23T19:20:360d-3Z`.
- Setiap transaksi berikutnya menciptakan revisi baru dengan dokumen yang sama`id`, nomor versi bertambah, dan diperbarui`txId` dan`txTime`.

Untuk memverifikasi revisi dokumen secara kriptografis di`vehicle-registration` buku besar, lanjutkan keLangkah 6: Verifikasi dokumen di buku besar.

# Langkah 6: Verifikasi dokumen di buku besar

Dengan Amazon QLDB, Anda dapat secara efisien memverifikasi integritas dokumen dalam jurnal buku besar Anda dengan menggunakan hashing kriptografi dengan SHA-256. Dalam contoh ini, Alexis dan Brent memutuskan untuk meningkatkan ke model baru dengan berdagang kendaraan dengan VIN`1N4AL11D75C109151` di dealer mobil. Dealer memulai proses dengan memverifikasi kepemilikan kendaraan dengan kantor pendaftaran.

Untuk mempelajari lebih lanjut tentang cara kerja verifikasi dan hashing kriptografi di QLDB, lihatVerifikasi data di Amazon QLDB.

Pada langkah ini, Anda memverifikasi revisi dokumen di`vehicle-registration` buku besar. Pertama, Anda meminta digest, yang dikembalikan sebagai file output dan bertindak sebagai tanda tangan dari seluruh riwayat perubahan buku besar Anda. Kemudian, Anda meminta bukti untuk revisi relatif terhadap intisari itu. Dengan menggunakan bukti ini, integritas revisi Anda diverifikasi jika semua pemeriksaan validasi lulus.

## Untuk meminta intisari

1. Buka konsol Amazon QLDB di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih Buku besar.
3. Dalam daftar buku besar, pilih`vehicle-registration`.
4. Pilih Dapatkan digest. Kotak dialog Get digest menampilkan rincian mencerna berikut:
   - Digest - Nilai hash SHA-256 dari intisari yang Anda minta.

- Alamat tip mencerna - Lokasi blok terbaru dalam jurnal yang dicakup oleh intisari yang Anda minta. Alamat memiliki dua bidang berikut:
  - `strandId`- ID unik untai jurnal yang berisi blok.
  - `sequenceNo`- Nomor indeks yang menentukan lokasi blok di dalam untai.
- Buku besar - Nama buku besar yang Anda minta mencerna.
- Tanggal - Stempel waktu saat Anda meminta intisari.

5. Tinjau informasi yang dicerna. Lalu, pilih Simpan. Anda dapat menyimpan nama file default, atau memasukkan nama baru.

   Langkah ini menyimpan file teks biasa dengan konten dalam format Amazon Ion. File ini memiliki ekstensi nama file `.ion.txt` dan berisi semua informasi mencerna yang tercantum pada kotak dialog sebelumnya. Berikut ini adalah contoh isi file digest. Urutan bidang dapat bervariasi tergantung pada browser Anda.

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

6. Simpan file ini di mana Anda dapat mengaksesnya nanti. Dalam langkah-langkah berikut, Anda menggunakan file ini untuk memverifikasi revisi dokumen terhadap.

Setelah Anda menyimpan intisari buku besar, Anda dapat memulai proses verifikasi revisi dokumen terhadap intisari itu.

> **Note**
>
> Dalam kasus penggunaan produksi untuk verifikasi, Anda menggunakan digest yang sebelumnya disimpan daripada melakukan dua tugas secara berurutan. Sebagai praktik terbaik, minta dan simpan intisari segera setelah revisi yang ingin Anda verifikasi nanti ditulis ke jurnal.

## Untuk memverifikasi revisi dokumen

1. Pertama, kueri buku besar Anda untuk`id` dan`blockAddress` revisi dokumen yang ingin Anda verifikasi. Bidang ini disertakan dalam metadata dokumen, yang dapat Anda kueri dalam tampilan berkomitmen.

   Dokumen`id` ini adalah string ID unik yang ditugaskan sistem. Itu`blockAddress` adalah struktur Ion yang menentukan lokasi blok tempat revisi dilakukan.

   Di panel navigasi konsol QLDB, pilih editor PartiQL.
2. Pilih`vehicle-registration` buku besar.
3. Di jendela editor kueri, masukkan pernyataan berikut, kemudian pilih Run (Jalankan).

   ```
   SELECT r.metadata.id, r.blockAddress
   FROM _ql_committed_VehicleRegistration AS r
   WHERE r.data.VIN = '1N4AL11D75C109151'
   ```
4. Salin dan simpan`id` dan`blockAddress` nilai yang dikembalikan kueri Anda. Pastikan untuk menghilangkan tanda kutip ganda untuk`id` bidang. Di Amazon Ion, tipe data string dibatasi dengan tanda kutip ganda.
5. Sekarang setelah Anda memiliki revisi dokumen yang dipilih, Anda dapat memulai proses memverifikasinya.

   Di panel navigasi, pilih Verifikasi.
6. Pada formulir Verifikasi dokumen, di bawah Tentukan dokumen yang ingin Anda verifikasi, masukkan parameter masukan berikut:

   - Ledger - Pilih`vehicle-registration`.
   - Blokir alamat -`blockAddress` Nilai yang dikembalikan oleh kueri Anda pada langkah 3.
   - ID Dokumen -`id` Nilai yang dikembalikan oleh kueri Anda pada langkah 3.
7. Di bawah Tentukan intisari yang akan digunakan untuk verifikasi, pilih intisari yang sebelumnya Anda simpan dengan memilih Choose digest. Jika file valid, ini otomatis mengisi semua bidang intisari di konsol Anda. Atau, Anda dapat menyalin dan menempelkan nilai berikut secara manual langsung dari file digest Anda:

   - Digest -`digest` Nilai dari file digest Anda.
   - Alamat tip mencerna -`digestTipAddress` Nilai dari file intisari Anda.

8. Tinjau dokumen Anda dan cerna parameter input, lalu pilih Verifikasi.

   Konsol mengotomatiskan dua langkah untuk Anda:

   a. Minta bukti dari QLDB untuk dokumen yang Anda tentukan.
   b. Gunakan bukti yang dikembalikan oleh QLDB untuk memanggil API sisi klien, yang memverifikasi revisi dokumen Anda terhadap intisari yang disediakan.

   Konsol menampilkan hasil permintaan Anda di kartu hasil Verifikasi. Untuk informasi selengkapnya, lihat Hasil verifikasi.
9. Untuk menguji logika verifikasi, ulangi langkah 6-8 di bawah Untuk memverifikasi revisi dokumen, tetapi ubah satu karakter dalam string input Digest. Ini akan menyebabkan permintaan Verifikasi Anda gagal dengan pesan kesalahan yang sesuai.

Jika Anda tidak perlu lagi menggunakan`vehicle-registration` buku besar, lanjutkan keLangkah 7 (opsional): Bersihkan Sumber Daya.

# Langkah 7 (opsional): Bersihkan Sumber Daya

Anda dapat terus menggunakan`vehicle-registration` buku besar. Namun, jika Anda tidak lagi membutuhkannya, Anda harus menghapusnya.

Jika perlindungan penghapusan diaktifkan untuk buku besar Anda, Anda harus menonaktifkannya terlebih dahulu sebelum dapat menghapus buku besar menggunakan konsol QLDB API,AWS Command Line Interface (AWS CLI), atau QLDB.

Untuk menghapus buku besar

1. Buka konsol Amazon QLDB di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih Buku besar.
3. Jika perlindungan penghapusan diaktifkan untuk buku besar ini, Anda harus mematikannya terlebih dahulu.

   Dalam daftar buku besar, pilih`vehicle-registration`, dan kemudian pilih Edit buku besar.
4. Di halaman Edit buku besar, matikan perlindungan Penghapusan, lalu pilih Konfirmasi perubahan.
5. Dalam daftar buku besar, pilih`vehicle-registration` lagi, lalu pilih Hapus.

6. Konfirmasikan ini dengan memasukkan**`delete vehicle-registration`** di bidang yang disediakan.

Untuk mempelajari selengkapnya tentang bekerja dengan buku besar di QLDB, lihatMemulai dengan Amazon QLDB: Langkah selanjutnya.

# Memulai dengan Amazon QLDB: Langkah selanjutnya

Untuk informasi selengkapnya tentang cara menggunakan Amazon QLDB, lihat topik berikut:

- Bekerja dengan data dan riwayat di Amazon QLDB
- Memulai dengan driver Amazon QLDB
- Model Konkurensi Amazon QLDB
- Mengekspor data jurnal dari Amazon QLDB
- Streaming data jurnal dari Amazon QLDB
- Verifikasi data di Amazon QLDB
- Referensi PartiQLDB QLDB

# Memulai dengan driver Amazon QLDB

Bab ini berisi tutorial langsung untuk membantu Anda belajar tentang pengembangan dengan Amazon QLDB dengan menggunakan driver QLDB. Driver dibangun di atasAWS SDK, yang mendukung interaksi dengan QLDB API.

Abstraksi sesi QLDB

Pengemudi menyediakan lapisan abstraksi tingkat tinggi di atas API data transaksional (Sesi QLDB). Ini merampingkan proses menjalankan pernyataan PartiQL pada data buku besar dengan mengelola panggilan SendCommandAPI. Panggilan API ini memerlukan beberapa parameter yang ditangani pengemudi untuk Anda, termasuk pengelolaan sesi, transaksi, dan kebijakan coba lagi jika terjadi kesalahan. Pengemudi juga memiliki optimasi kinerja dan menerapkan praktik terbaik untuk berinteraksi dengan QLDB.

> **Note**
>
> Untuk berinteraksi dengan operasi API manajemen sumber daya yang tercantum dalam referensi API Amazon QLDB, Anda menggunakanAWS SDK secara langsung, bukan driver. Anda menggunakan API manajemen hanya untuk mengelola sumber daya buku besar dan untuk operasi data non-transaksional, seperti mengekspor, streaming, dan verifikasi data.

Dukungan Amazon Ion

Selain itu, driver menggunakan pustaka Amazon Ion untuk memberikan dukungan untuk menangani data Ion saat menjalankan transaksi. Pustaka ini juga menangani penghitungan hash nilai Ion. QLDB membutuhkan hash Ion ini untuk memeriksa integritas permintaan transaksi data.

Terminologi pengemudi

Alat ini disebut driver karena sebanding dengan driver database lain yang menyediakan antarmuka yang ramah pengembang. Driver ini juga merangkum logika yang mengubah serangkaian perintah dan fungsi standar menjadi panggilan tertentu yang diperlukan oleh API tingkat rendah layanan.

Driver aktif open source GitHub dan tersedia dalam bahasa pemrograman berikut:

- Driver Java
- Driver .NET

- Driver Go
- Sopir Node.js
- Driver Python

Untuk informasi driver umum untuk semua bahasa pemrograman yang didukung, dan tutorial tambahan, lihat topik berikut:

- Manajemen sesi dengan pengemudi
- Rekomendasi pengemudi
- Kebijakan coba lagi pengemudi
- Kesalahan umum
- Aplikasi aplikasi sampel sampel sampel untuk aplikasi sampel sampel
- Bekerja dengan Amazon Ion
- Mendapatkan statistik pernyataan PartiQL

# Driver QLDB Amazon untuk Java

Untuk bekerja dengan data dalam buku besar Anda, Anda dapat terhubung ke Amazon QLDB dari aplikasi Java Anda dengan menggunakan driver yangAWS disediakan. Topik berikut menjelaskan cara untuk memulai dengan driver QLDB untuk Java.

Topik

- Sumber daya driver
- Prasyarat
- MengaturAWS kredensi dan Wilayah default
- Instalasi
- Driver Amazon QLDB untuk Java - Tutorial mulai cepat
- Driver Amazon QLDB untuk Java - referensi Cookbook

## Sumber daya driver

Untuk informasi selengkapnya tentang fungsionalitas yang didukung oleh driver Java, lihat sumber daya berikut:

- Referensi API: 2.x, 1.x
- Kode sumber driver (GitHub)
- Contoh kode sumber aplikasi (GitHub)
- Kerangka loader buku besar (GitHub)
- Contoh kode Amazon Ion

## Prasyarat

Sebelum Anda memulai dengan driver QLDB untuk Java, Anda harus melakukan hal berikut:

1. Ikuti petunjukAWS pengaturan diMengakses Amazon QLDB. Hal ini termasuk skenario berikut:
    1. Daftar ke AWS.
    2. Buat pengguna dengan izin QLDB yang sesuai.
    3. Memberikan akses terprogram untuk pengembangan.
2. Siapkan lingkungan pengembangan Java dengan mengunduh dan menginstal yang berikut ini:
    1. Java SE Development Kit 8, seperti Amazon Corretto 8.
    2. (Opsional) Java integrated development environment (IDE) pilihan Anda, seperti Eclipse atau IntelliJ.
3. Konfigurasikan lingkungan pengembangan Anda untukAWS SDK for Java byMengaturAWS kredensi dan Wilayah default.

Selanjutnya, Anda dapat mengunduh aplikasi contoh tutorial lengkap—atau Anda hanya dapat menginstal driver dalam proyek Java dan menjalankan contoh kode pendek.

- Untuk menginstal driver QLDB danAWS SDK for Java dalam proyek yang ada, lanjutkan keInstalasi.
- Untuk menyiapkan proyek dan menjalankan contoh kode pendek yang menunjukkan transaksi data dasar pada buku besar, lihatTutorial Quick Start.
- Untuk menjalankan contoh yang lebih mendalam dari kedua data dan operasi API manajemen dalam aplikasi sampel tutorial lengkap, lihattutorial java.

## MengaturAWS kredensi dan Wilayah default

Driver QLDB dan yang mendasari AWS SDK for Javamengharuskan Anda memberikanAWS kredensi untuk aplikasi Anda pada saat runtime. Contoh kode dalam panduan ini menganggap bahwa Anda menggunakan fileAWS kredensi, seperti yang dijelaskan dalam Menyiapkan kredensi dan Wilayah default dalam PanduanAWS SDK for Java 2.x Developer.

Sebagai bagian dari langkah-langkah ini, Anda juga harus mengatur default AndaWilayah AWS untuk menentukan endpoint QLDB default Anda. Contoh kode terhubung ke QLDB di default AndaWilayah AWS. Untuk daftar lengkap Wilayah tempat QLDB tersedia, lihat titik akhir dan kuota Amazon QLDB di bagian Referensi Umum AWS.

Berikut ini adalah contoh file kredensial AWS bernama `~/.aws/credentials`, di mana karakter tilde (~) mewakili direktori beranda Anda.

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

Ganti nilaiAWS kredensi Anda sendiri untuk nilai *your_key_key_key_key_key_key_secret_access_key_key_key_key_key_key_key_key_k*

## Instalasi

QLDB mendukung versi driver Java berikut dan dependensiAWS SDK mereka.

| Versi driver | AWS SDK | Status | Tanggal rilis terbaru |
|---|---|---|---|
| 1.x | AWS SDK for Java1.x | Rilis produksi | 20 Maret 2020 |
| 2.x | AWS SDK for Java 2.x | Rilis produksi | 4 Juni 2021 |

Untuk menginstal driver QLDB, sebaiknya gunakan sistem manajemen dependensi, seperti Gradle atau Maven. Misalnya, tambahkan artefak berikut sebagai dependensi dalam proyek Java Anda.

2.x

Gradle

Tambahkan dependensi ini di file`build.gradle` konfigurasi Anda.

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

Maven

Tambahkan dependensi ini di file`pom.xml` konfigurasi Anda.

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

Artefak ini secara otomatis menyertakan modulAWS SDK for Java 2.x inti, pustaka Amazon Ion, dan dependensi lain yang diperlukan.

1.x

Gradle

Tambahkan dependensi ini di file`build.gradle` konfigurasi Anda.

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

Maven

Tambahkan dependensi ini di file`pom.xml` konfigurasi Anda.

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

```
</dependencies>
```

Artefak ini secara otomatis menyertakan modulAWS SDK for Java inti, pustaka Amazon Ion, dan dependensi lain yang diperlukan.

> **Important**
>
> Namespace Amazon Ion — Saat mengimpor kelas Amazon Ion di aplikasi Anda, Anda harus menggunakan paket yang berada di bawah namespace`com.amazon.ion`. AWS SDK for JavaTergantung pada paket Ion lain di bawah namespace`software.amazon.ion`, tetapi ini adalah paket lama yang tidak kompatibel dengan driver QLDB.

Untuk contoh kode singkat tentang cara menjalankan transaksi data dasar pada buku besar, lihatReferensi buku masak.

## Pustaka opsional lainnya

Anda juga dapat menambahkan library berguna berikut dalam proyek Anda. Artefak ini diperlukan dependensi dalam aplikasitutorial java sampel.

1. aws-java-sdk-qldb- Modul QLDB dariAWS SDK for Java. Versi minimum yang didukung QLDB adalah`1.11.785`.

   Gunakan modul ini di aplikasi Anda untuk berinteraksi langsung dengan operasi API manajemen yang tercantum dalam aplikasiReferensi API Amazon QLDB.
2. jackson-dataformat-ion- Modul format data Jackson FasterXML's untuk Ion. Aplikasi sampel membutuhkan versi`2.10.0` atau yang lebih baru.

### Gradle

Tambahkan dependensi ini di file`build.gradle` konfigurasi Anda.

```
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

Tambahkan dependensi ini di file`pom.xml` konfigurasi Anda.

```
<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 untuk Java - Tutorial mulai cepat

Dalam tutorial ini, Anda belajar cara mengatur aplikasi sederhana menggunakan versi terbaru dari driver Amazon QLDB untuk Java. Panduan ini mencakup langkah-langkah untuk menginstal driver dan contoh kode pendek operasi dasar buat, baca, update, and delete (CRUD). Untuk contoh yang lebih mendalam yang menunjukkan operasi ini dalam aplikasi sampel lengkap, lihattutorial java.

Topik



### Prasyarat

Sebelum memulai, pastikan Anda melakukan hal berikut:

1. Lengkapi driver Java, jika Anda belum melakukannya.Prasyarat Ini termasuk mendaftarAWS, memberikan akses terprogram untuk pengembangan, dan menginstal Java integrated development environment (IDE).
2. Buat buku besar bernama`quick-start`.

   Untuk mempelajari cara membuat buku besar, lihatOperasi dasar untuk buku besar Amazon QLDB atauLangkah 1: Membuat buku besar baru di Memulai konsol.

## Langkah 1: Siapkan proyek Anda

Pertama, siapkan proyek Java Anda. Kami merekomendasikan menggunakan sistem manajemen ketergantungan Maven untuk tutorial ini.

> **Note**
>
> Jika Anda menggunakan IDE yang memiliki fitur untuk mengotomatiskan langkah-langkah penyiapan ini, Anda dapat melompat ke depanLangkah 2: Inisialisasi driver.

1. Buat folder untuk aplikasi Anda.

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

2. Masukkan perintah berikut untuk menginisialisasi proyek Anda dari template Maven. Ganti *`project-package`*, *`project-name`*, dan *`maven-template`* dengan nilai-nilai Anda sendiri yang sesuai.

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

   Untuk *`maven-template`*, Anda dapat menggunakan template Maven dasar:`maven-archetype-quickstart`

3. Untuk menambahkan driver QLDB untuk Java sebagai ketergantungan proyek, navigasikan kepom.xml file yang baru dibuat dan tambahkan artefak berikut.

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

Artefak ini secara otomatis menyertakan modul AWS SDK for Java 2.xinti, pustaka Amazon Ion, dan dependensi lain yang diperlukan. `pom.xml`File Anda sekarang akan terlihat seperti berikut.

```
<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. Buka file `App.java`.

   Kemudian, secara bertahap menambahkan contoh kode dalam langkah-langkah berikut untuk mencoba beberapa operasi CRUD dasar. Atau, Anda dapat melewati step-by-step tutorial dan sebagai gantinya menjalankan aplikasi lengkap.

## Langkah 2: Inisialisasi driver

Inisialisasi instance driver yang terhubung ke buku besar bernama`quick-start`. Tambahkan kode berikut ke`App.java` file Anda.

```
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();
    }
}
```

## Langkah 3: Buat tabel dan indeks

Contoh kode berikut menunjukkan cara menjalankan`CREATE TABLE` dan`CREATE INDEX` pernyataan.

Dalam`main` metode ini, tambahkan kode berikut yang membuat tabel bernama`People` dan indeks untuk`lastName` bidang pada tabel itu. Indeks diperlukan untuk mengoptimalkan kinerja kueri dan membantu membatasi pengecualian konflik kontrol konkurensi (OCC) yang optimis.

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

```
});
```

## Langkah 4: Masukkan dokumen

Contoh kode berikut menunjukkan cara menjalankan`INSERT` pernyataan. QLDB mendukung bahasa kueri PartiQL (kompatibel dengan SQL) dan format data Amazon Ion (superset JSON).

Tambahkan kode berikut yang menyisipkan dokumen ke dalam`People` tabel.

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

Contoh ini menggunakan tanda tanya (?) sebagai placeholder variabel untuk meneruskan informasi dokumen untuk pernyataan. Bila Anda menggunakan placeholder, Anda harus lulus nilai jenis`IonValue`.

> **Tip**
>
> Untuk menyisipkan beberapa dokumen dengan menggunakanSISIPKAN pernyataan tunggal, Anda dapat melewati parameter jenis IonList(secara eksplisit dilemparkan sebagai`IonValue`) untuk pernyataan sebagai berikut.
>
> ```
> // people is an IonList explicitly cast as an IonValue
> txn.execute("INSERT INTO People ?", (IonValue) people);
> ```
>
> Anda tidak melampirkan variabel placeholder (?) dalam kurung sudut ganda (`<<...>>`) ketika melewati`IonList`. Dalam pernyataan PartiQL manual, kurung sudut ganda menunjukkan koleksi unordered dikenal sebagai tas.

## Langkah 5: Cari dokumen

Contoh kode berikut menunjukkan cara menjalankan`SELECT` pernyataan.

Tambahkan kode berikut yang meminta dokumen dari`People` tabel.

```
// 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
});
```

## Langkah 6: Perbarui dokumen

Contoh kode berikut menunjukkan cara menjalankan`UPDATE` pernyataan.

1. Tambahkan kode berikut yang memperbarui dokumen dalam`People` tabel dengan memperbarui`age` ke`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. Kueri dokumen lagi untuk melihat nilai yang diperbarui.

   ```
   // 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. Gunakan Maven atau IDE Anda untuk menjalankan`App.java` file.

## Menjalankan aplikasi lengkap

Contoh kode berikut adalah versi lengkap dari`App.java` aplikasi. Alih-alih melakukan langkah-langkah sebelumnya secara individual, Anda juga dapat menyalin dan menjalankan contoh kode ini dari awal hingga akhir. Aplikasi ini menunjukkan beberapa operasi CRUD dasar pada buku besar bernama`quick-start`.

> **Note**
>
> Sebelum Anda menjalankan kode ini, pastikan bahwa Anda belum memiliki tabel aktif bernama`People` dalam`quick-start` buku besar.
> Pada baris pertama, ganti *`proyek-paket`* dengan`groupId` nilai yang Anda gunakan untuk perintah Maven diLangkah 1: Siapkan proyek Anda.

```
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
        });
    }
}
```

Gunakan Maven atau IDE Anda untuk menjalankan`App.java` file.

## Driver Amazon QLDB untuk Java - referensi Cookbook

Panduan referensi ini menunjukkan kasus penggunaan umum driver QLDB Amazon untuk Java. Ini memberikan contoh kode Java yang menunjukkan cara menggunakan driver untuk menjalankan operasi dasar buat, baca, perbarui, dan hapus atau create, read, update, and delete (CRUD). Ini juga mencakup contoh kode untuk memproses data Amazon Ion. Selain itu, panduan ini menyoroti praktik terbaik untuk membuat transaksi idempoten dan menerapkan kendala keunikan.

> **Note**
>
> Jika berlaku, beberapa kasus penggunaan memiliki contoh kode yang berbeda untuk setiap versi utama driver QLDB yang didukung untuk Java.

Daftar Isi

## Mengimpor driver

Contoh kode berikut mengimpor driver, klien sesi QLDB, paket Amazon Ion, dan dependensi terkait lainnya.

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;
```

## Beri contoh driver

Contoh kode berikut membuat contoh driver yang terhubung ke nama buku besar tertentu, dan menggunakan logika coba lagi tertentu dengan batas percobaan ulang kustom.

> **Note**
>
> Contoh ini juga membuat instance objek sistem Amazon Ion (`IonSystem`). Anda memerlukan objek ini untuk memproses data Ion saat menjalankan beberapa operasi data dalam referensi ini. Untuk mempelajari selengkapnya, lihat Bekerja dengan 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();
```

## Operasi CRUD

QLDB berjalan operasi buat, baca, memperbarui, dan menghapus (CRUD) operasi sebagai bagian dari transaksi.

> **Warning**
>
> Sebagai praktik terbaik, buatlah transaksi tulis Anda secara ketat idempoten.

### Melakukan transaksi idempoten

Kami menyarankan Anda melakukan transaksi tulis idempoten untuk menghindari efek samping yang tidak terduga dalam kasus percobaan ulang. Transaksi idempoten jika dapat berjalan beberapa kali dan menghasilkan hasil yang identik setiap kali.

Misalnya, pertimbangkan transaksi yang memasukkan dokumen ke tabel bernama`Person`. Transaksi pertama-tama harus memeriksa apakah dokumen sudah ada dalam tabel atau tidak. Tanpa pemeriksaan ini, tabel mungkin berakhir dengan dokumen duplikat.

Misalkan QLDB berhasil melakukan transaksi di sisi server, tetapi klien kali keluar sambil menunggu respon. Jika transaksi tidak idempoten, dokumen yang sama dapat dimasukkan lebih dari sekali dalam kasus percobaan ulang.

### Menggunakan indeks untuk menghindari pemindaian tabel penuh

Kami juga menyarankan Anda menjalankan pernyataan dengan klausa`WHERE` predikat menggunakan operator kesetaraan pada bidang yang diindeks atau ID dokumen; misalnya,`WHERE indexedField = 123` atau`WHERE indexedField IN (456, 789)`. Tanpa pencarian yang diindeks ini, QLDB perlu melakukan pemindaian tabel, yang dapat menyebabkan batas waktu transaksi atau konflik kontrol konkurensi (OCC) yang optimis.

Untuk informasi lebih lanjut tentang OCC, lihat[Model Konkurensi Amazon QLDB](#).

### Transaksi yang dibuat secara implisit

Metode QldbDriver.execute menerima fungsi lambda yang menerima instance Pelaksana, yang dapat Anda gunakan untuk menjalankan pernyataan. `Executor`Instance membungkus transaksi yang dibuat secara implisit.

Anda dapat menjalankan pernyataan dalam fungsi lambda dengan menggunakan`Executor.execute` metode. Pengemudi secara implisit melakukan transaksi ketika fungsi lambda kembali.

Bagian berikut menunjukkan cara menjalankan operasi CRUD dasar, menentukan logika percobaan ulang kustom, dan menerapkan batasan keunikan.

> **Note**
>
> Jika berlaku, bagian ini memberikan contoh kode untuk memproses data Amazon Ion menggunakan pustaka Ion bawaan dan pustaka pemetaan Jackson Ion. Untuk mempelajari selengkapnya, lihat Bekerja dengan Amazon Ion.

Daftar Isi



## Membuat Tabel

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

## Membuat indeks

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

## Membaca dokumen

```
// 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
});
```

Menggunakan parameter kueri

Contoh kode berikut menggunakan parameter kueri tipe 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
});
```

Contoh kode berikut menggunakan beberapa parameter query.

```
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
});
```

Contoh kode berikut menggunakan daftar parameter query.

```
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
});
```

Menggunakan pemetaan 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();
    }
});
```

Menggunakan parameter kueri

Contoh kode berikut menggunakan parameter kueri tipe 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();
    }
});
```

Contoh kode berikut menggunakan beberapa parameter query.

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

Contoh kode berikut menggunakan daftar parameter query.

```
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**
>
> Ketika Anda menjalankan kueri tanpa pencarian yang diindeks, itu akan memanggil pemindaian tabel penuh. Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, kueri dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

### Memasukkan dokumen

Contoh kode berikut menyisipkan tipe data 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);
    }
});
```

Menggunakan pemetaan Jackson

Contoh kode berikut menyisipkan tipe data 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();
    }
});
```

Transaksi ini menyisipkan dokumen ke dalam`Person` tabel. Sebelum memasukkan, pertama-tama memeriksa apakah dokumen sudah ada di tabel. Cek ini membuat transaksi idempoten di alam. Bahkan jika Anda menjalankan transaksi ini beberapa kali, itu tidak akan menyebabkan efek samping yang tidak diinginkan.

> **Note**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

Memasukkan beberapa dokumen dalam satu pernyataan

Untuk menyisipkan beberapa dokumen dengan menggunakanSISIPKAN pernyataan tunggal, Anda dapat melewati parameter jenis IonList(secara eksplisit dilemparkan sebagai`IonValue`) untuk pernyataan sebagai berikut.

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

Anda tidak melampirkan variabel placeholder (?) dalam kurung sudut ganda (`<<...>>`) ketika melewati`IonList`. Dalam pernyataan PartiQL manual, kurung sudut ganda menunjukkan koleksi unordered dikenal sebagai tas.

Mengapa pemeran eksplisit diperlukan?

TransactionExecutorMetode.execute kelebihan beban. Ia menerima sejumlah variabel`IonValue` argumen (varargs), atau`List<IonValue>` argumen tunggal. Dalam ion-java,`IonList` diimplementasikan sebagai`List<IonValue>`.

Java default untuk implementasi metode yang paling spesifik ketika Anda memanggil metode kelebihan beban. Dalam hal ini, ketika Anda melewati`IonList` parameter, default ke metode yang mengambil`List<IonValue>`. Ketika dipanggil, implementasi metode ini melewati`IonValue` elemen daftar sebagai nilai yang berbeda. Jadi, untuk memanggil metode varargs sebagai gantinya, Anda harus secara eksplisit mentransmisikan`IonList` parameter sebagai sebuah`IonValue`.

Memperbarui dokumen

```
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);
```

```
});
```

### Menggunakan pemetaan 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**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

### Menghapus dokumen

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

### Menggunakan pemetaan Jackson

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

> **Note**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

Menjalankan beberapa pernyataan dalam transaksi

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

Logika coba lagi

`execute`Metode pengemudi memiliki mekanisme coba ulang bawaan yang mencoba ulang transaksi jika terjadi pengecualian yang dapat dicoba ulang (seperti batas waktu atau konflik OCC).

2.x

Jumlah maksimum upaya coba lagi dan strategi backoff dapat dikonfigurasi.

Batas percobaan ulang default adalah4, dan strategi backoff default adalah DefaultQldbTransactionBackoffStrategy. Anda dapat mengatur konfigurasi coba ulang per contoh driver dan juga per transaksi dengan menggunakan instance RetryPolicy.

Contoh kode berikut menentukan logika coba lagi dengan batas percobaan ulang kustom dan strategi cadangan khusus untuk instance 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());
    }
}
```

Contoh kode berikut menentukan logika coba lagi dengan batas percobaan ulang kustom dan strategi cadangan khusus untuk transaksi tertentu. Konfigurasi ini untuk`execute` menimpa logika coba ulang yang diatur untuk instance 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

Jumlah maksimum upaya coba lagi dapat dikonfigurasi. Anda dapat mengkonfigurasi batas coba lagi dengan menetapkan`retryLimit` properti saat menginisialisasi`PooledQldbDriver`.

Batas percobaan ulang default adalah`4`.

Menerapkan kendala keunikan

QLDB tidak mendukung indeks unik, tetapi Anda dapat menerapkan perilaku ini dalam aplikasi Anda.

Misalkan Anda ingin menerapkan kendala keunikan pada`GovId` bidang dalam`Person` tabel. Untuk melakukan ini, Anda dapat menulis transaksi yang melakukan hal berikut:

1. Menegaskan bahwa tabel tidak memiliki dokumen yang ada dengan yang ditentukan`GovId`.
2. Masukkan dokumen jika pernyataan berlalu.

Jika transaksi yang bersaing secara bersamaan melewati pernyataan, hanya satu dari transaksi yang akan berhasil dilakukan. Transaksi lainnya akan gagal dengan pengecualian konflik OCC.

Contoh kode berikut ini menunjukkan cara menerapkan logika kendala keunikan ini.

```
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**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

## Bekerja dengan Amazon Ion

Ada beberapa cara untuk memproses data Amazon Ion di QLDB. Anda dapat menggunakan metode bawaan dari pustaka Ion untuk membuat dan memodifikasi dokumen secara fleksibel sesuai kebutuhan. Atau, Anda dapat menggunakan modul format data Jackson FasterXML's untuk Ion untuk memetakan dokumen Ion ke model objek Java (POJO) lama.

Bagian berikut memberikan contoh kode pengolahan data Ion menggunakan kedua teknik.

Daftar Isi



### Mengimpor paket Ion

Tambahkan artefak ion-java sebagai dependensi dalam proyek Java Anda.

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>
```

Impor paket Ion berikut.

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

Menggunakan pemetaan Jackson

Tambahkan artefak jackson-dataformat-ionsebagai dependensi dalam proyek Java Anda. QLDB membutuhkan versi`2.10.0` atau yang lebih baru.

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>
```

Impor paket Ion berikut.

```
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;
```

Inisialisasi Ion

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

Menggunakan pemetaan Jackson

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

Membuat objek Ion

Contoh kode berikut membuat objek Ion dengan menggunakan`IonStruct` antarmuka dan metode built-in.

```
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
```

Menggunakan pemetaan Jackson

Misalkan Anda memiliki kelas model yang dipetakan JSON bernama`Person`, sebagai berikut.

```
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;
    }
}
```

Contoh kode berikut menciptakan sebuah`IonStruct` objek dari sebuah instance dari`Person`.

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

Membaca objek Ion

Contoh kode berikut mencetak setiap bidang`ionStruct` instance.

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

Menggunakan pemetaan Jackson

Contoh kode berikut membaca sebuah`IonStruct` objek dan peta ke sebuah instance dari`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
```

Untuk informasi selengkapnya tentang bekerja dengan Ion, lihat dokumentasi Amazon Ion GitHub. Untuk contoh kode lebih lanjut bekerja dengan Ion di QLDB, lihatBekerja dengan tipe data Amazon Ion di Amazon QLDB.

# Driver Amazon QLDB for .NET

Untuk bekerja dengan data dalam buku besar Anda, Anda dapat terhubung ke Amazon QLDB dari aplikasi Microsoft .NET Anda dengan menggunakan driver yangAWS disediakan. Pengemudi menargetkan .NET Standard 2.0. Lebih khusus lagi, mendukung .NET Core (LTS) 2.1+ dan .NET Framework 4.5.2+. Untuk informasi tentang kompatibilitas, lihat .NET Standard di situs Microsoft Docs.

Kami sangat merekomendasikan penggunaan pemetaan objek Ion untuk sepenuhnya melewati kebutuhan untuk mengonversi secara manual antara jenis Amazon Ion dan tipe C # asli.

Topik berikut menjelaskan cara untuk memulai dengan driver QLDB for .NET.

Topik



## Sumber daya driver

Untuk informasi selengkapnya tentang fungsi yang didukung oleh driver .NET, lihat sumber daya berikut:

- Referensi API
- Kode sumber driver (GitHub)
- Contoh kode sumber aplikasi (GitHub)
- Buku Masak Amazon Ion
- Ion objek mapper (GitHub)

## Prasyarat

Sebelum memulai dengan driver QLDB for .NET, Anda harus melakukan hal berikut:

1. Ikuti petunjukAWS pengaturan diMengakses Amazon QLDB. Ini termasuk hal berikut:

    1. Daftar ke AWS.
    2. Buat pengguna dengan izin QLDB yang sesuai.
    3. Memberikan akses terprogram untuk pengembangan.
2. Unduh dan instal .NET Core SDK versi 2.1 atau yang lebih baru dari situs unduhan Microsoft .NET.
3. (Opsional) Instal lingkungan pengembangan terintegrasi (IDE) pilihan Anda, seperti Visual Studio, Visual Studio untuk Mac, atau Visual Studio Code. Anda dapat men-download ini dari situs Microsoft Visual Studio.
4. Konfigurasikan lingkungan pengembangan Anda untuk AWS SDK for .NET:

    1. SiapkanAWS kredensi Anda. Kami merekomendasikan membuat file kredensi bersama.

        Untuk petunjuknya, lihat MengonfigurasiAWS kredensi menggunakan file kredensil di PanduanAWS SDK for .NET Pengembang.
    2. Tetapkan default AndaWilayah AWS. Untuk mempelajari caranya, lihat Wilayah AWSseleksi.

        Untuk daftar lengkap Wilayah yang tersedia, lihat titik akhir dan kuota Amazon QLDB di bagian Referensi Umum AWS.

Selanjutnya, Anda dapat mengatur aplikasi sampel dasar dan menjalankan contoh kode singkat—atau Anda dapat menginstal driver dalam proyek .NET yang ada.

- Untuk menginstal driver QLDB danAWS SDK for .NET dalam proyek yang ada, lanjutkan keInstalasi.
- Untuk menyiapkan proyek dan menjalankan contoh kode pendek yang menunjukkan transaksi data dasar pada buku besar, lihatQuick start tutorial.

## Instalasi

Gunakan manajer NuGet paket untuk menginstal driver QLDB untuk .NET. Sebaiknya gunakan Visual Studio atau IDE pilihan Anda untuk menambahkan dependensi proyek. Nama paket driver adalah Amazon.qldb.driver.

Misalnya di Visual Studio, buka NuGet Package Manager Console pada menu Tools. Kemudian, masukkan perintah berikut diPM> perintah.

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

Menginstal driver juga menginstal dependensinya, termasuk paketAWS SDK for .NET dan Amazon Ion.

### Instal pemetaan objek Ion

Versi 1.3.0 driver QLDB untuk .NET memperkenalkan dukungan untuk menerima dan mengembalikan tipe data C # asli tanpa perlu bekerja dengan Amazon Ion. Untuk menggunakan fitur ini, tambahkan paket berikut ke proyek Anda.

- Amazon.qldb.driver.serialization - Sebuah perpustakaan yang dapat memetakan nilai Ion ke C # objek CLR lama polos (POCO), dan sebaliknya. Pemeta objek Ion ini memungkinkan aplikasi Anda berinteraksi langsung dengan tipe data C # asli tanpa perlu bekerja dengan Ion. Untuk panduan singkat tentang cara menggunakan pustaka ini, lihat file Serialization.md di GitHub repositoriawslabs/amazon-qldb-driver-dotnet.

Untuk menginstal paket ini, masukkan perintah berikut.

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

Untuk contoh kode singkat tentang cara menjalankan transaksi data dasar pada buku besar, lihatReferensi Referensi.

## Driver Amazon QLDB untuk .NET - Tutorial mulai cepat

Dalam tutorial ini, Anda belajar cara mengatur aplikasi sederhana menggunakan driver Amazon QLDB untuk .NET. Panduan ini mencakup langkah-langkah untuk menginstal driver dan contoh kode pendek dari operasi pembuatan, pembacaan, pembaruan, dan penghapusan (CRUD).

Topik

- Langkah 5: Cari Dokumen
- Langkah 6: Perbarui Dokumen
- Menjalankan aplikasi lengkap

## Prasyarat

Sebelum memulai, pastikan Anda melakukan hal berikut:

1. Lengkapi driverPrasyarat untuk .NET, jika Anda belum memilhnya. Ini termasuk mendaftarAWS, memberikan akses terprogram untuk pengembangan, dan menginstal .NET Core SDK.
2. Buat buku besar bernamaquick-start.

   Untuk mempelajari cara membuat buku besar, lihatOperasi dasar untuk buku besar Amazon QLDB atauLangkah 1: Membuat buku besar baru di Memulai konsol.

## Langkah 1: Siapkan proyek Anda

Pertama, siapkan proyek .NET Anda.

1. Untuk membuat dan menjalankan aplikasi template, masukkandotnet perintah berikut pada terminal seperti bash, PowerShell, atau Command Prompt.

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

   Template ini membuat folder bernamaAmazon.QLDB.QuickStartGuide. Dalam folder itu, itu menciptakan sebuah proyek dengan nama yang sama dan file bernamaProgram.cs. Program ini berisi kode yang menampilkan outputHello World!.
2. Gunakan manajer NuGet paket untuk menginstal driver QLDB untuk .NET. Sebaiknya gunakan Visual Studio atau IDE pilihan Anda untuk menambahkan dependensi ke proyek Anda. Nama paket driver adalah Amazon.qldb.driver.
   - Misalnya di Visual Studio, buka NuGet Package Manager Console pada menu Tools. Kemudian, masukkan perintah berikut diPM> prompt.

     ```
     PM> Install-Package Amazon.QLDB.Driver
     ```
   - Atau, Anda dapat memasukkan perintah berikut di terminal Anda.

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

Menginstal driver juga menginstal dependensinya, termasuk library AWS SDK for .NETdan Amazon Ion.

3. Instal pustaka serialisasi driver.

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

4. Buka file `Program.cs`.

Kemudian, secara bertahap menambahkan contoh kode dalam langkah-langkah berikut untuk mencoba beberapa operasi CRUD dasar. Atau, Anda dapat melewati step-by-step tutorial dan sebagai gantinya menjalankan aplikasi lengkap.

> **Note**
>
> - Memilih antara API sinkron dan asinkron - Driver menyediakan API sinkron dan asinkron. Untuk aplikasi permintaan tinggi yang menangani beberapa permintaan tanpa memblokir, sebaiknya gunakan API asinkron untuk meningkatkan kinerja. Pengemudi menawarkan API sinkron sebagai kenyamanan tambahan untuk basis kode yang ada yang ditulis secara serempak.
>
>   Tutorial ini mencakup contoh kode sinkron dan asinkron. Untuk informasi lebih lanjut tentang API, lihatAsyncQldbDriver antarmuka IQldbDriver dan I dalam dokumentasi API.
> - Memproses data Amazon Ion - Tutorial ini memberikan contoh kode untuk memproses data Amazon Ion menggunakan pemeta objek Ion secara default. QLDB memperkenalkan pemetaan objek Ion dalam versi 1.3.0 dari driver .NET. Jika berlaku, tutorial ini juga menyediakan contoh kode menggunakan pustaka Ion standar sebagai alternatif. Untuk mempelajari selengkapnya, lihat Bekerja dengan Amazon Ion.

## Langkah 2: Inisialisasi driver

Inisialisasi instance driver yang terhubung ke buku besar bernama`quick-start`. Tambahkan kode berikut ke`Program.cs` file Anda.

### 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();
        }
    }
}
```

### Menggunakan Pustaka 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();
        }
    }
}
```

## Langkah 3: Membuat tabel dan indeks

Untuk sisa tutorial ini melalui Langkah 6, Anda perlu menambahkan contoh kode berikut ke contoh kode sebelumnya.

Pada langkah ini, kode berikut menunjukkan bagaimana menjalankan`CREATE TABLE` dan`CREATE INDEX` pernyataan. Ini menciptakan tabel bernama`Person` dan indeks untuk`firstName` bidang

pada tabel itu. Indeks diperlukan untuk mengoptimalkan kinerja kueri dan membantu membatasi pengecualian konflik kontrol konkurensi (OCC) yang optimis.

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)");
});
```

## Langkah 4: Masukkan dokumen

Contoh kode berikut ini menunjukkan cara menjalankan`INSERT` pernyataan. QLDB mendukung bahasa kueri PartiQL (kompatibel dengan SQL) dan format data Amazon Ion (superset JSON).

Tambahkan kode berikut yang menyisipkan dokumen ke dalam`Person` tabel.

### 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);
});
```

## Menggunakan Pustaka 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**

Untuk menyisipkan beberapa dokumen dengan menggunakanSISIPKAN pernyataan tunggal, Anda dapat melewati parameter jenis daftar Ion untuk pernyataan sebagai berikut.

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

Anda tidak menyertakan variabel placeholder (?) dalam kurung sudut ganda (`<<...>>`) saat meneruskan daftar Ion. Dalam pernyataan PartiQL manual, kurung sudut ganda menunjukkan koleksi unordered dikenal sebagai tas.

## Langkah 5: Cari Dokumen

Contoh kode berikut ini menunjukkan cara menjalankan`SELECT` pernyataan.

Tambahkan kode berikut yang meminta dokumen dari`Person` tabel.

### 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
}
```

### Menggunakan Pustaka 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);
}
```

Contoh ini menggunakan tanda tanya (?) sebagai placeholder variabel untuk meneruskan informasi dokumen untuk pernyataan. Bila Anda menggunakan placeholder, Anda harus lulus nilai jenis`IonValue`.

## Langkah 6: Perbarui Dokumen

Contoh kode berikut ini menunjukkan cara menjalankan`UPDATE` pernyataan.

1. Tambahkan kode berikut yang memperbarui dokumen dalam`Person` tabel dengan memperbarui`age` ke 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. Kueri dokumen lagi untuk melihat nilai yang diperbarui.

   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. Untuk menjalankan aplikasi, masukkan perintah berikut dari direktori induk direktori`Amazon.QLDB.QuickStartGuide` proyek Anda.

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

Menggunakan Pustaka Ion

1. Tambahkan kode berikut yang memperbarui dokumen dalam`Person` tabel dengan memperbarui`age` ke 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. Kueri dokumen lagi untuk melihat nilai yang diperbarui.

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. Untuk menjalankan aplikasi, masukkan perintah berikut dari direktori induk direktori`Amazon.QLDB.QuickStartGuide` proyek Anda.

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

## Menjalankan aplikasi lengkap

Contoh kode berikut adalah versi lengkap dari`Program.cs` aplikasi. Alih-alih melakukan langkah-langkah sebelumnya secara individual, Anda juga dapat menyalin dan menjalankan contoh kode ini dari awal hingga akhir. Aplikasi ini menunjukkan beberapa operasi CRUD dasar pada buku besar bernama`quick-start`.

> **Note**
>
> Sebelum Anda menjalankan kode ini, pastikan bahwa Anda belum memiliki tabel aktif bernama`Person` dalam`quick-start` buku besar.

### 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
            }

        }
    }
}
```

### Menggunakan Pustaka 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);
            }
        }
    }
}
```

Untuk menjalankan aplikasi lengkap, masukkan perintah berikut dari direktori induk direktori`Amazon.QLDB.QuickStartGuide` proyek Anda.

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

## Driver Amazon QLDB untuk .NET - referensi Cookbook

Panduan referensi ini menunjukkan kasus penggunaan umum driver QLDB Amazon untuk .NET. Ini memberikan contoh kode C# yang menunjukkan cara menggunakan driver untuk menjalankan operasi dasar buat, baca, perbarui, dan hapus atau create, read, and delete (CRUD). Ini juga mencakup contoh kode untuk memproses data Amazon Ion. Selain itu, panduan ini menyoroti praktik terbaik untuk membuat transaksi idempoten dan menerapkan kendala keunikan.

> **Note**
>
> Topik ini memberikan contoh kode pemrosesan data Amazon Ion menggunakan pemetaan objek Ion secara default. QLDB memperkenalkan pemetaan objek Ion dalam versi 1.3.0 dari driver .NET. Jika berlaku, topik ini juga menyediakan contoh kode menggunakan pustaka Ion

standar sebagai alternatif. Untuk mempelajari selengkapnya, lihat Bekerja dengan Amazon Ion.

Daftar Isi



## Mengimpor driver

Contoh kode berikut mengimpor driver.

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

Menggunakan Pustaka

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

## Beri contoh driver

Contoh kode berikut membuat sebuah instance dari driver yang menghubungkan ke nama ledger tertentu menggunakan pengaturan default.

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();
```

Menggunakan Pustaka

Async

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

Sync

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

## Operasi CRUD

QLDB menjalankan operasi buat, baca, memperbarui, dan menghapus (CRUD) sebagai bagian dari transaksi.

> **Warning**
>
> Sebagai praktik terbaik, buatlah transaksi tulis Anda dengan ketat idempoten.

### Melakukan transaksi idempoten

Kami menyarankan Anda melakukan transaksi tulis idempoten untuk menghindari efek samping yang tidak terduga dalam kasus percobaan ulang. Transaksi idempoten jika dapat berjalan beberapa kali dan menghasilkan hasil yang identik setiap kali.

Misalnya, pertimbangkan transaksi yang memasukkan dokumen ke tabel bernama`Person`. Transaksi pertama-tama harus memeriksa apakah dokumen sudah ada dalam tabel atau tidak. Tanpa pemeriksaan ini, tabel mungkin berakhir dengan dokumen duplikat.

Misalkan QLDB berhasil melakukan transaksi di sisi server, tetapi klien kali keluar sambil menunggu respon. Jika transaksi tidak idempoten, dokumen yang sama dapat dimasukkan lebih dari sekali dalam kasus percobaan ulang.

### Menggunakan indeks untuk menghindari pemindaian tabel penuh

Kami juga menyarankan Anda menjalankan pernyataan dengan klausa`WHERE` predikat menggunakan operator kesetaraan pada bidang yang diindeks atau ID dokumen; misalnya,`WHERE indexedField = 123` atau`WHERE indexedField IN (456, 789)`. Tanpa pencarian yang diindeks ini, QLDB perlu melakukan pemindaian tabel, yang dapat menyebabkan batas waktu transaksi atau konflik kontrol konkurensi (OCC) yang optimis.

Untuk informasi lebih lanjut tentang OCC, lihat[Model Konkurensi Amazon QLDB](#).

### Transaksi yang dibuat secara implisit

Metode [QldbDriverAmazon.qldb.driver.i .Execute](#) menerima fungsi lambda yang menerima instance [Amazon.qldb.driver. TransactionExecutor](#), yang dapat Anda gunakan untuk menjalankan pernyataan. Contoh`TransactionExecutor` membungkus transaksi yang dibuat secara implisit.

Anda dapat menjalankan pernyataan dalam fungsi lambda dengan menggunakan`Execute` metode pelaksana transaksi. Pengemudi secara implisit melakukan transaksi ketika fungsi lambda kembali.

Bagian berikut menunjukkan cara menjalankan operasi CRUD dasar, menentukan logika percobaan ulang kustom, dan menerapkan batasan keunikan.

Daftar Isi



## Membuat Tabel

### 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 }
}
```

### Menggunakan Pustaka

#### 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"
    // }
}
```

### Membuat indeks

#### 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 }
}
```

## Menggunakan Pustaka

### 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"
    // }
}
```

## Membaca dokumen

```
// 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**
>
> Ketika Anda menjalankan kueri tanpa pencarian yang diindeks, itu akan memanggil pemindaian tabel penuh. Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, kueri dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

### Menggunakan parameter kueri

Contoh kode berikut menggunakan C # jenis parameter query.

```
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.
}
```

Contoh kode berikut menggunakan beberapa C # jenis parameter query.

```
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.
}
```

Contoh kode berikut menggunakan array C # jenis parameter query.

```
// 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.
}
```

Contoh kode berikut menggunakan C # daftar sebagai nilai.

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

### Menggunakan Pustaka

#### 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**
>
> Ketika Anda menjalankan kueri tanpa pencarian yang diindeks, itu akan memanggil pemindaian tabel penuh. Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, kueri dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

Contoh kode berikut menggunakan parameter kueri tipe 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.
}
```

Contoh kode berikut menggunakan beberapa parameter query.

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.
}
```

Contoh kode berikut menggunakan daftar parameter query.

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.
}
```

Contoh kode berikut menggunakan daftar Ion sebagai nilai. Untuk mempelajari selengkapnya tentang menggunakan tipe Ion, lihatBekerja dengan tipe data Amazon Ion di 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"
    //     }
    //     ]
    // }
}
```

### Memasukkan dokumen

Contoh kode berikut menyisipkan tipe data 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));
});
```

## Menggunakan Pustaka

### 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);
});
```

Transaksi ini menyisipkan dokumen ke dalam`Person` tabel. Sebelum memasukkan, pertama-tama memeriksa apakah dokumen sudah ada di tabel. Cek ini membuat transaksi idempoten di alam. Bahkan jika Anda menjalankan transaksi ini beberapa kali, itu tidak akan menyebabkan efek samping yang tidak diinginkan.

> **Note**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

Memasukkan beberapa dokumen dalam satu pernyataan

Untuk menyisipkan beberapa dokumen dengan menggunakanSISIPKAN pernyataan tunggal, Anda dapat melewati C #`List` parameter untuk pernyataan sebagai berikut.

```
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 }
}
```

Menggunakan Pustaka

Untuk menyisipkan beberapa dokumen dengan menggunakanSISIPKAN pernyataan tunggal, Anda dapat melewati parameter tipe daftar Ion untuk pernyataan sebagai berikut.

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"
    // }
}
```

Anda tidak menyertakan variabel placeholder (?) dalam kurung sudut ganda (<<...>>) saat meneruskan daftar Ion. Dalam pernyataan PartiQL manual, kurung sudut ganda menunjukkan koleksi unordered dikenal sebagai tas.

### Memperbarui dokumen

```
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 }
}
```

### Menggunakan Pustaka

#### 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**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

### Menghapus dokumen

```
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 }
}
```

### Menggunakan Pustaka

#### 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**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

### Menjalankan beberapa pernyataan dalam transaksi

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

### Menggunakan Pustaka

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

Logika coba lagi

Untuk informasi tentang logika coba ulang bawaan pengemudi, lihat[Memahami kebijakan coba ulang dengan pengemudi di Amazon QLDB](#).

Mengimplementasikan kendala

QLDB tidak mendukung indeks unik, tetapi Anda dapat menerapkan perilaku ini dalam aplikasi Anda.

Misalkan Anda ingin menerapkan kendala keunikan pada`GovId` bidang dalam`Person` tabel. Untuk melakukan ini, Anda dapat menulis transaksi yang melakukan hal berikut:

1. Menegaskan bahwa tabel tidak memiliki dokumen yang ada dengan yang ditentukan`GovId`.
2. Masukkan dokumen jika pernyataan berlalu.

Jika transaksi yang bersaing secara bersamaan melewati pernyataan, hanya satu dari transaksi yang akan berhasil dilakukan. Transaksi lainnya akan gagal dengan pengecualian konflik OCC.

Contoh kode berikut ini menunjukkan cara menerapkan logika kendala keunikan ini.

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

### Menggunakan Pustaka

#### 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**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

## Bekerja dengan Amazon Ion

Ada beberapa cara untuk memproses data Amazon Ion di QLDB. Anda dapat menggunakan pustaka Ion untuk membuat dan memodifikasi nilai Ion. Atau, Anda dapat menggunakan pemetaan objek Ion untuk memetakan C # objek CLR tua polos (POCO) ke dan dari nilai Ion. Versi 1.3.0 dari driver QLDB untuk .NET memperkenalkan dukungan untuk pemetaan objek Ion.

Bagian berikut memberikan contoh kode pengolahan data Ion menggunakan kedua teknik.

Daftar Isi



### Mengimpor modul Ion

```
using Amazon.IonObjectMapper;
```

### Menggunakan Pustaka

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

### Membuat jenis Ion

Contoh kode berikut menunjukkan cara membuat nilai Ion dari C # objek menggunakan Ion objek mapper.

```
// Assumes that Person class is defined as follows:
// public class Person
// {
```

```
//     public string FirstName { get; set; }
//     public int Age { get; set; }
// }

// Initialize the Ion Object Mapper
IonSerializer ionSerializer = new IonSerializer();

// The C# object to be serialized
Person person = new Person
{
    FirstName = "John",
    Age = 13
};

// Serialize the C# object into stream using the Ion Object Mapper
Stream stream = ionSerializer.Serialize(person);

// Load will take in stream and return a datagram; a top level container of Ion values.
IIonValue ionDatagram = IonLoader.Default.Load(stream);

// To get the Ion value within the datagram, we call GetElementAt(0).
IIonValue ionPerson = ionDatagram.GetElementAt(0);

Console.WriteLine(ionPerson.GetField("firstName").StringValue);
Console.WriteLine(ionPerson.GetField("age").IntValue);
```

### Menggunakan Pustaka

Contoh kode berikut menunjukkan dua cara untuk membuat nilai Ion menggunakan pustaka Ion.

### Menggunakan`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);
```

### Menggunakan`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);
```

### Mendapatkan dump biner 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()));
```

### Menggunakan Pustaka

```
// 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()));
```

Mendapatkan dump teks 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()));
```

Menggunakan Pustaka

```
// ionObject is an Ion struct
StringWriter sw = new StringWriter();
using (var writer = IonTextWriterBuilder.Build(sw))
{
    ionObject.WriteTo(writer);
    writer.Finish();
}

Console.WriteLine(sw.ToString());
```

Untuk informasi lebih lanjut tentang bekerja dengan Ion, lihat dokumentasi Amazon Ion GitHub. Untuk contoh kode lebih lanjut bekerja dengan Ion di QLDB, lihatBekerja dengan tipe data Amazon Ion di Amazon QLDB.

# Driver QLDB Amazon untuk Go

Untuk bekerja dengan data dalam buku besar Anda, Anda dapat terhubung ke Amazon QLDB dari aplikasi Go Anda dengan menggunakan driver yangAWS disediakan. Topik berikut menjelaskan cara untuk memulai driver QLDB untuk Go.

Topik



## Sumber daya driver

Untuk informasi selengkapnya tentang fungsionalitas yang didukung driver Go, lihat sumber daya berikut:

- Referensi API: 3.x, 2.x, 1.x
- Kode sumber driver (GitHub)
- Buku Masak Amazon Ion

## Prasyarat

Sebelum Anda menggunakan driver QLDB untuk Go, Anda harus melakukan hal berikut:

1. Ikuti petunjukAWS pengaturan diMengakses Amazon QLDB. Ini mencakup hal berikut:
    1. Daftar ke AWS.
    2. Buat pengguna dengan izin QLDB yang sesuai.
    3. Memberikan akses terprogram untuk pengembangan.
2. (Opsional) Instal lingkungan pengembangan terintegrasi (IDE) pilihan Anda. Untuk daftar IDE yang umum digunakan untuk Go, lihat Editor plugin dan IDE di situs web Go.
3. Unduh dan instal salah satu versi Go dari situs unduhan Go:
    - 1.15 atau yang lebih baru — driver QLDB untuk Go v3
    - 1.14 - driver QLDB untuk Go v1 atau v2
4. Konfigurasikan lingkungan pengembangan Anda untuk AWS SDK for Go:
    1. SiapkanAWS kredensi Anda. Kami sarankan membuat file kredensi bersama.

        Untuk petunjuknya, lihat Menentukan Kredensyal di PanduanAWS SDK for Go Developer.

2. Tetapkan default AndaWilayah AWS. Untuk mempelajari caranya, lihat MenentukanWilayah AWS.

   Untuk daftar lengkap Wilayah yang tersedia, lihat titik akhir dan kuota Amazon QLDB di bagian Referensi Umum AWS.

Selanjutnya, Anda dapat mengatur aplikasi sampel dasar dan menjalankan contoh kode singkat—atau Anda dapat menginstal driver dalam proyek Go yang ada.

- Untuk menginstal driver QLDB danAWS SDK for Go dalam proyek yang ada, lanjutkan keInstalasi.
- Untuk menyiapkan proyek dan menjalankan contoh kode pendek yang menunjukkan transaksi data dasar pada buku besar, lihatTutorial Quick start.

## Instalasi

Driver QLDB untuk Go adalah open source di awslabs GitHub repositori/amazon-qldb-driver-go. QLDB mendukung versi driver berikut dan dependensi Go mereka.

| Versi driver | Pergi versi | Status | Tanggal rilis terbaru |
|---|---|---|---|
| 1.x | 1.14 atau yang lebih baru | Rilis produksi | 16 Juni 2021 |
| 2.x | 1.14 atau yang lebih baru | Rilis produksi | 21 Juli 2021 |
| 3.x | 1.15 atau yang lebih baru | Rilis produksi | 10 November 2022 |

Untuk menginstal driver

1. Pastikan proyek Anda menggunakan modul Go untuk menginstal dependensi proyek.
2. Di direktori proyek Anda, masukkan`go get` perintah berikut.

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
```

Menginstal driver juga menginstal dependensinya, termasuk AWS SDK for Goatau AWS SDK for Gov2, dan paket Amazon Ion.

Untuk contoh kode singkat tentang cara menjalankan transaksi data dasar pada buku besar, lihatReferensi buku masak.

## Driver Amazon QLDB untuk Go - Tutorial mulai cepat

Dalam tutorial ini, Anda belajar cara mengatur aplikasi sederhana menggunakan versi terbaru dari driver Amazon QLDB untuk Go. Panduan ini mencakup langkah-langkah untuk menginstal driver dan contoh kode pendek operasi dasar buat, baca, perbarui, dan hapus atau create, read, update, and delete (CRUD).

Topik

## Prasyarat

Sebelum memulai, pastikan Anda melakukan hal berikut:

1. Selesaikan driverPrasyarat untuk Go, jika Anda belum memilihnya. Ini termasuk mendaftarAWS, memberikan akses terprogram untuk pengembangan, dan menginstal Go.
2. Buat buku besar bernama`quick-start`.

   Untuk mempelajari cara membuat buku besar, lihatOperasi dasar untuk buku besar Amazon QLDB atauLangkah 1: Membuat buku besar baru di Memulai konsol.

## Langkah 1: Instal driver

Pastikan proyek Anda menggunakan modul Go untuk menginstal dependensi proyek.

Di direktori proyek Anda, masukkan`go get` perintah berikut.

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

Menginstal driver juga menginstal dependensinya, termasuk paket AWS SDK for Gov2 dan Amazon Ion.

## Langkah 2: Impor paket

ImporAWS paket berikut.

```
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"
)
```

## Langkah 3: Menginisialisasi driver

Inisialisasi sebuah instance dari driver yang terhubung ke buku besar bernama`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**
>
> Dalam contoh kode ini, ganti *us-timur-1* denganWilayah AWS tempat Anda membuat buku besar Anda.

## Langkah 4: Buat tabel dan indeks

Contoh kode berikut menunjukkan cara menjalankan`CREATE TABLE` dan`CREATE INDEX` pernyataan.

```
_, 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)
}
```

Kode ini membuat tabel bernama`People`, dan indeks untuk`firstName` dan`age` bidang pada tabel itu. Indeks diperlukan untuk mengoptimalkan kinerja kueri dan membantu membatasi pengecualian konflik kontrol konkurensi (OCC) yang optimis.

## Langkah 5: Masukkan dokumen

Contoh kode berikut menunjukkan cara menjalankan`INSERT` pernyataan. QLDB mendukung bahasa kueri PartiQL (kompatibel dengan SQL) dan format data Amazon Ion (superset JSON).

Menggunakan literal PartiQL

Kode berikut menyisipkan dokumen ke dalam`People` tabel menggunakan string pernyataan PartiQL literal.

```
_, 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)
}
```

Menggunakan tipe data Ion

Mirip dengan paket JSON bawaan Go, Anda dapat melakukan marshal dan unmarshal Go tipe data ke dan dari Ion.

1. Misalkan Anda memiliki struktur Go berikut bernama`Person`.

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

2. Buat instans `Person`.

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

Driver marshals representasi teks ION-dikodekan`person` untuk Anda.

> **Important**
>
> Agar marshal dan unmarshal berfungsi dengan baik, nama bidang struktur data Go harus diekspor (huruf pertama dikapitalisasi).

3. Lulus`person` instance ke`Execute` metode transaksi.

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

Contoh ini menggunakan tanda tanya (?) sebagai placeholder variabel untuk meneruskan informasi dokumen untuk pernyataan. Saat Anda menggunakan placeholder, Anda harus meneruskan nilai teks yang dikodekan ION.

> **Tip**
>
> Untuk menyisipkan beberapa dokumen dengan menggunakanSISIPKAN pernyataan tunggal, Anda dapat melewati parameter daftar jenis untuk pernyataan sebagai berikut.
>
> ```
> // people is a list
> ```

```
txn.Execute("INSERT INTO People ?", people)
```

Anda tidak melampirkan variabel placeholder (?) dalam kurung sudut ganda (<<...>>) ketika melewati daftar. Dalam pernyataan PartiQL manual, kurung sudut ganda menunjukkan koleksi unordered dikenal sebagai tas.

## Langkah 6: Kueri-lah dokumen

Contoh kode berikut menunjukkan cara menjalankanSELECT pernyataan.

```
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")
}
```

Contoh ini query dokumen Anda dari`People` tabel, mengasumsikan bahwa hasil set tidak kosong, dan mengembalikan dokumen Anda dari hasil.

## Langkah 7: Perbarui dokumen

Contoh kode berikut menunjukkan cara menjalankan`UPDATE` pernyataan.

```
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)
}
```

## Langkah 8: Kueri-lah dokumen yang diperbarui

Contoh kode berikut query`People` tabel oleh`firstName` dan mengembalikan semua dokumen dalam hasil set.

```
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")
}
```

## Langkah 9: Jatuhkan tabel

Contoh kode berikut menunjukkan cara menjalankan`DROP TABLE` pernyataan.

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

## Menjalankan aplikasi lengkap

Contoh kode berikut adalah versi lengkap dari aplikasi. Alih-alih melakukan langkah-langkah sebelumnya secara individual, Anda juga dapat menyalin dan menjalankan contoh kode ini dari awal hingga akhir. Aplikasi ini menunjukkan beberapa operasi CRUD dasar pada buku besar bernama`quick-start`.

> **Note**
>
> Sebelum Anda menjalankan kode ini, pastikan bahwa Anda belum memiliki tabel aktif bernama`People` dalam`quick-start` buku besar.

```
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 untuk Go - referensi Cookbook

Panduan referensi ini menunjukkan kasus penggunaan umum driver QLDB Amazon untuk Go. Ini menyediakan contoh kode Go yang menunjukkan cara menggunakan driver untuk menjalankan operasi dasar buat, membaca, memperbarui, dan menghapus (CRUD). Ini juga mencakup contoh kode untuk memproses data Amazon Ion. Selain itu, panduan ini menyoroti praktik terbaik untuk membuat transaksi idempoten dan menerapkan kendala keunikan.

> **Note**
>
> Jika berlaku, beberapa kasus penggunaan memiliki contoh kode yang berbeda untuk setiap versi utama driver QLDB yang didukung untuk Go.

Daftar Isi

- Mengimpor modul Ion
- Membuat jenis Ion
- Mendapatkan biner Ion
- Mendapatkan teks Ion

## Mengimpor driver

Contoh kode berikut mengimpor driver danAWS paket lain yang diperlukan.

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**
>
> Contoh ini juga mengimpor paket Amazon Ion (`amzn/ion-go/ion`). Anda memerlukan paket ini untuk memproses data Ion saat menjalankan beberapa operasi data dalam referensi ini. Untuk mempelajari selengkapnya, lihat Bekerja dengan Amazon Ion.

## Beri contoh driver

Contoh kode berikut menciptakan sebuah instance dari driver yang menghubungkan ke nama ledger tertentu dalam tertentuWilayah 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)
}
```

## Operasi CRUD

QLDB menjalankan operasi buat, membaca, memperbarui, dan menghapus (CRUD) operasi sebagai bagian dari transaksi.

> **Warning**
>
> Untuk praktik terbaik, buatlah transaksi tulis Anda secara ketat idempoten.

### Melakukan transaksi idempoten

Kami menyarankan Anda melakukan transaksi tulis idempoten untuk menghindari efek samping yang tidak terduga dalam kasus percobaan ulang. Transaksi idempoten jika dapat berjalan beberapa kali dan menghasilkan hasil yang identik setiap kali.

Misalnya, pertimbangkan transaksi yang memasukkan dokumen ke tabel bernama`Person`. Transaksi pertama-tama harus memeriksa apakah dokumen sudah ada dalam tabel atau tidak. Tanpa pemeriksaan ini, tabel mungkin berakhir dengan dokumen duplikat.

Misalkan QLDB berhasil melakukan transaksi di sisi server, tetapi klien kali keluar sambil menunggu respon. Jika transaksi tidak idempoten, dokumen yang sama dapat dimasukkan lebih dari sekali dalam kasus percobaan ulang.

### Menggunakan indeks untuk menghindari pemindaian tabel penuh

Kami juga menyarankan Anda menjalankan pernyataan dengan klausa`WHERE` predikat menggunakan operator kesetaraan pada bidang yang diindeks atau ID dokumen; misalnya,`WHERE indexedField = 123` atau`WHERE indexedField IN (456, 789)`. Tanpa pencarian yang diindeks ini, QLDB perlu melakukan pemindaian tabel, yang dapat menyebabkan batas waktu transaksi atau konflik kontrol konkurensi (OCC) yang optimis.

Untuk informasi lebih lanjut tentang OCC, lihatModel Konkurensi Amazon QLDB.

### Transaksi yang dibuat secara implisit

Fungsi QldbDriver.execute menerima fungsi lambda yang menerima instance Transaction, yang dapat Anda gunakan untuk menjalankan pernyataan. Contoh`Transaction` membungkus transaksi yang dibuat secara implisit.

Anda dapat menjalankan pernyataan dalam fungsi lambda dengan menggunakan`Transaction.Execute` fungsi. Pengemudi secara implisit melakukan transaksi ketika fungsi lambda kembali.

Bagian berikut menunjukkan cara menjalankan operasi CRUD dasar, menentukan logika percobaan ulang kustom, dan menerapkan batasan keunikan.

Daftar Isi



### Membuat Tabel

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

### Membuat indeks

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

### Membaca dokumen

```
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)
}
```

Menggunakan parameter kueri

Contoh kode berikut menggunakan parameter jenis query asli.

```
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)
}
```

Contoh kode berikut menggunakan beberapa parameter query.

```
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)
}
```

Contoh kode berikut menggunakan daftar parameter query.

```
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**
>
> Ketika Anda menjalankan kueri tanpa pencarian yang diindeks, itu akan memanggil pemindaian tabel penuh. Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, kueri dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

### Memasukkan dokumen

Contoh kode berikut menyisipkan tipe data asli.

```
_, 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
})
```

Transaksi ini menyisipkan dokumen ke dalam`Person` tabel. Sebelum memasukkan, pertama-tama memeriksa apakah dokumen sudah ada di tabel. Cek ini membuat transaksi idempoten di alam. Bahkan jika Anda menjalankan transaksi ini beberapa kali, itu tidak akan menyebabkan efek samping yang tidak diinginkan.

> **Note**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

Memasukkan beberapa dokumen dalam satu pernyataan

Untuk menyisipkan beberapa dokumen dengan menggunakanSISIPKAN pernyataan tunggal, Anda dapat melewati parameter daftar jenis untuk pernyataan sebagai berikut.

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

Anda tidak melampirkan variabel placeholder (?) dalam kurung sudut ganda (`<<...>>`) ketika melewati daftar. Dalam pernyataan PartiQL manual, kurung sudut ganda menunjukkan koleksi unordered dikenal sebagai tas.

Memperbarui dokumen

Contoh kode berikut menggunakan tipe data asli.

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

> **Note**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

### Menghapus dokumen

Contoh kode berikut menggunakan tipe data asli.

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

> **Note**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

### Menjalankan beberapa pernyataan dalam transaksi

```
// 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
}
```

Logika coba lagi

`Execute`Fungsi pengemudi memiliki mekanisme coba ulang bawaan yang mencoba ulang transaksi jika terjadi pengecualian yang dapat dicoba ulang (seperti batas waktu atau konflik OCC). Jumlah maksimum upaya coba lagi dan strategi backoff dapat dikonfigurasi.

Batas percobaan ulang default adalah`4`, dan strategi backoff default adalah ExponentialBackoffStrategydengan basis`10` milidetik. Anda dapat mengatur kebijakan coba ulang per instance driver dan juga per transaksi dengan menggunakan instance RetryPolicy.

Contoh kode berikut menentukan logika coba lagi dengan batas percobaan ulang kustom dan strategi cadangan khusus untuk instance 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)
  }
}
```

Contoh kode berikut menentukan logika coba lagi dengan batas percobaan ulang kustom dan strategi backoff kustom untuk fungsi anonim tertentu. `SetRetryPolicy`Fungsi menimpa kebijakan coba ulang yang ditetapkan untuk instance 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")
  })
}
```

### Mengimplementasikan kendala keunikan

QLDB tidak mendukung indeks unik, tetapi Anda dapat menerapkan perilaku ini dalam aplikasi Anda.

Misalkan Anda ingin menerapkan kendala keunikan pada`GovId` bidang dalam`Person` tabel. Untuk melakukan ini, Anda dapat menulis transaksi yang melakukan hal berikut:

1. Menegaskan bahwa tabel tidak memiliki dokumen yang ada dengan ditentukan`GovId`.

2. Masukkan dokumen jika pernyataan berlalu.

Jika transaksi yang bersaing secara bersamaan melewati pernyataan, hanya satu dari transaksi yang akan berhasil dilakukan. Transaksi lainnya akan gagal dengan pengecualian konflik OCC.

Contoh kode berikut menunjukkan cara menerapkan logika kendala keunikan ini.

```
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**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

# Bekerja dengan Amazon Ion

Bagian berikut menunjukkan cara menggunakan modul Amazon Ion untuk memproses data Ion.

Daftar Isi



## Mengimpor modul Ion

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

## Membuat jenis Ion

Pustaka Ion untuk Go saat ini tidak mendukung Document Object Model (DOM), jadi Anda tidak dapat membuat tipe data Ion. Tapi Anda dapat marshal dan unmarshal antara Go tipe asli dan Ion biner ketika bekerja dengan QLDB.

## Mendapatkan biner Ion

```
aDict := map[string]interface{}{
  "GovId": "TOYENC486FH",
  "FirstName": "Brent",
}

ionBytes, err := ion.MarshalBinary(aDict)
if err != nil {
  panic(err)
}

fmt.Println(ionBytes) // prints [224 1 0 234 238 151 129 131 222 147 135 190 144 133 71
 111 118 73 100 137 70 105 114 115 116 78 97 109 101 222 148 138 139 84 79 89 69 78 67
 52 56 54 70 72 139 133 66 114 101 110 116]
```

## Mendapatkan teks 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"}
```

Untuk informasi selengkapnya tentang Ion, lihat dokumentasi Amazon Ion di GitHub. Untuk contoh kode lebih lanjut bekerja dengan Ion di QLDB, lihatBekerja dengan tipe data Amazon Ion di Amazon QLDB.

# Driver Amazon QLDB untuk Node.js

Untuk bekerja dengan data dalam buku besar Anda, Anda dapat terhubung ke Amazon QLDB dari aplikasi Node.js Anda dengan menggunakan driver yang disediakan. AWS Topik berikut menjelaskan cara memulai dengan driver QLDB untuk Node.js.

Topik

- Sumber daya pengemudi
- Prasyarat
- Instalasi
- Rekomendasi pengaturan
- Driver Amazon QLDB untuk Node.js - Tutorial mulai cepat
- Driver Amazon QLDB untuk Node.js - referensi Cookbook

## Sumber daya pengemudi

Untuk informasi selengkapnya tentang fungsionalitas yang didukung oleh driver Node.js, lihat sumber daya berikut:

- Referensi API: 3.x, 2.x, 1.x
- Kode sumber driver (GitHub)
- Contoh kode sumber aplikasi (GitHub)

- Contoh kode Amazon Ion
- Membangun operasi CRUD sederhana dan aliran data pada AWS Lambda QLDB menggunakan (Blog) AWS

## Prasyarat

Sebelum Anda memulai dengan driver QLDB untuk Node.js, Anda harus melakukan hal berikut:

1. Ikuti instruksi AWS pengaturan diMengakses Amazon QLDB. Ini termasuk yang berikut:
    1. Daftar ke AWS.
    2. Buat pengguna dengan izin QLDB yang sesuai.
    3. Memberikan akses terprogram untuk pengembangan.
2. Instal Node.js versi 14.x atau yang lebih baru dari situs unduhan Node.js. (Versi driver sebelumnya mendukung Node.js versi 10.x atau yang lebih baru.)
3. Konfigurasikan lingkungan pengembangan Anda untuk AWSSDK untuk JavaScript di Node.js:
    1. Siapkan AWS kredensional Anda. Sebaiknya buat file kredensial bersama.

        Untuk petunjuknya, lihat Memuat kredensi di Node.js dari file kredensial bersama di Panduan Pengembang. AWS SDK for JavaScript
    2. Tetapkan default AndaWilayah AWS. Untuk mempelajari caranya, lihat Mengatur Wilayah AWS.

        Untuk daftar lengkap Wilayah yang tersedia, lihat titik akhir dan kuota Amazon QLDB di bagian. Referensi Umum AWS

Selanjutnya, Anda dapat mengunduh aplikasi contoh tutorial lengkap—atau Anda hanya dapat menginstal driver dalam proyek Node.js dan menjalankan contoh kode pendek.

- Untuk menginstal driver QLDB dan SDK JavaScript untuk AWS di Node.js dalam proyek yang ada, lanjutkan ke. Instalasi
- Untuk menyiapkan proyek dan menjalankan contoh kode pendek yang menunjukkan transaksi data dasar pada buku besar, lihat. Tutorial Quick Start
- Untuk menjalankan contoh yang lebih mendalam dari operasi API data dan manajemen dalam aplikasi contoh tutorial lengkap, lihat. Node.js tutorial

# Instalasi

QLDB mendukung versi driver berikut dan dependensi Node.js mereka.

| Versi Driver | Versi Node.js | Status | Tanggal rilis terbaru |
|---|---|---|---|
| 1.x | 10.x atau yang lebih baru | Rilis produksi | 5 Juni 2020 |
| 2.x | 10.x atau yang lebih baru | Rilis produksi | 6 Mei 2021 |
| 3.x | 14.x atau lebih baru | Rilis produksi | November 10, 2023 |

Untuk menginstal driver QLDB menggunakan npm (manajer paket Node.js), masukkan perintah berikut dari direktori root proyek Anda.

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
```

Driver memiliki dependensi peer pada paket-paket berikut. Anda juga harus menginstal paket-paket ini sebagai dependensi dalam proyek Anda.

3.x

Klien QLDB agregat modular (API manajemen)

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

Klien Sesi QLDB agregat modular (API data)

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

Format data Amazon Ion

```
npm install ion-js
```

JavaScript Implementasi murni `BigInt`

```
npm install jsbi
```

2.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Format data Amazon Ion

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

JavaScript Implementasi murni `BigInt`

```
npm install jsbi@3.1.1
```

1.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Format data Amazon Ion

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

JavaScript Implementasi murni `BigInt`

```
npm install jsbi@3.1.1
```

Menggunakan driver untuk terhubung ke buku besar

Kemudian Anda dapat mengimpor driver dan menggunakannya untuk terhubung ke buku besar. Contoh TypeScript kode berikut menunjukkan cara membuat instance driver untuk nama buku besar tertentu danWilayah 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);
});
```

Untuk contoh kode singkat tentang cara menjalankan transaksi data dasar pada buku besar, lihat. Referensi buku masak

## Rekomendasi pengaturan

### Menggunakan kembali koneksi dengan keep-alive

QLDB Node.js driver v3

Agen HTTP/HTTPS Node.js default membuat koneksi TCP baru untuk setiap permintaan baru. Untuk menghindari biaya membuat koneksi baru, AWS SDK for JavaScript v3 menggunakan kembali koneksi TCP secara default. Untuk informasi selengkapnya dan mempelajari cara menonaktifkan penggunaan kembali koneksi, lihat Menggunakan kembali koneksi dengan keep-alive di Node.js di Panduan Pengembang. AWS SDK for JavaScript

Sebaiknya gunakan pengaturan default untuk menggunakan kembali koneksi di driver QLDB untuk Node.js. Selama inisialisasi driver, atur opsi HTTP klien tingkat rendah `maxSockets` ke nilai yang sama dengan yang Anda tetapkan. `maxConcurrentTransactions`

Misalnya, lihat TypeScript kode JavaScript atau kode berikut.

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);
```

QLDB Node.js driver v2

Agen HTTP/HTTPS Node.js default membuat koneksi TCP baru untuk setiap permintaan baru. Untuk menghindari biaya pembuatan koneksi baru, kami sarankan untuk menggunakan kembali koneksi yang ada.

Untuk menggunakan kembali koneksi di driver QLDB untuk Node.js, gunakan salah satu opsi berikut:

- Selama inisialisasi driver, atur opsi HTTP klien tingkat rendah berikut:
  - `keepAlive` – `true`
  - `maxSockets`— Nilai yang sama yang Anda tetapkan `maxConcurrentTransactions`

  Misalnya, lihat TypeScript kode JavaScript atau kode berikut.

  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);
```

- Atau, Anda dapat mengatur variabel `AWS_NODEJS_CONNECTION_REUSE_ENABLED` lingkungan ke1. Untuk informasi selengkapnya, lihat Menggunakan Kembali Koneksi dengan Keep-Alive di Node.js di Panduan Pengembang. AWS SDK for JavaScript

> **Note**
>
> Jika Anda mengatur variabel lingkungan ini, itu mempengaruhi semua Layanan AWS yang menggunakanAWS SDK for JavaScript.

## Driver Amazon QLDB untuk Node.js - Tutorial mulai cepat

Dalam tutorial ini, Anda belajar cara mengatur aplikasi sederhana menggunakan driver Amazon QLDB untuk Node.js. Panduan ini mencakup langkah-langkah untuk menginstal driver, dan contoh singkat JavaScript dan TypeScript kode dari operasi dasar buat, baca, perbarui, dan hapus atau create, read, update, and delete (CRUD). Untuk contoh yang lebih mendalam yang menunjukkan operasi ini dalam aplikasi sampel lengkap, lihatNode.js tutorial.

> **Note**
>
> Jika berlaku, beberapa langkah memiliki contoh kode yang berbeda untuk setiap versi utama driver QLDB yang didukung untuk Node.js.

Topik

- Prasyarat

## Prasyarat

Sebelum memulai, pastikan Anda melakukan hal berikut:

1. Lengkapi driverPrasyarat untuk Node.js, jika Anda belum melakukannya. Ini termasuk mendaftarAWS, memberikan akses terprogram untuk pengembangan, dan menginstal Node.js.
2. Buat buku besar bernama`quick-start`.

   Untuk mempelajari cara membuat buku besar, lihatOperasi dasar untuk buku besar Amazon QLDB atauLangkah 1: Membuat buku besar baru di Memulai konsol.

Jika Anda menggunakan TypeScript, Anda juga harus melakukan langkah-langkah pengaturan berikut.

Menggunakan TypeScript

Untuk menginstal TypeScript

1. Pasang TypeScript paket. Driver QLDB berjalan pada TypeScript 3.8.x.

   ```
   $ npm install --global typescript@3.8.0
   ```

2. Setelah paket terinstal, gunakan perintah berikut untuk memastikan TypeScript kompilator terinstal.

   ```
   $ tsc --version
   ```

Untuk menjalankan kode dalam langkah-langkah berikut, perhatikan bahwa Anda harus terlebih dahulu transpile TypeScript file Anda ke JavaScript kode executable, sebagai berikut.

```
$ tsc app.ts; node app.js
```

## Langkah 1: Siapkan proyek Anda

Pertama, siapkan proyek Node.js Anda.

1. Membuat folder untuk aplikasi Anda.

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

2. Untuk menginisialisasi proyek Anda, masukkannpm perintah berikut dan jawab pertanyaan yang diajukan selama penyiapan. Anda dapat menggunakan default untuk sebagian besar pertanyaan.

   ```
   $ npm init
   ```

3. Instal driver Amazon QLDB untuk Node.js.
   - Menggunakan versi 3.x

     ```
     $ npm install amazon-qldb-driver-nodejs --save
     ```

   - Menggunakan versi 2.x

     ```
     $ npm install amazon-qldb-driver-nodejs@2.2.0 --save
     ```

   - Menggunakan versi 1.x

     ```
     $ npm install amazon-qldb-driver-nodejs@1.0.0 --save
     ```

4. Instal dependensi rekan driver.
   - Menggunakan versi 3.x

     ```
     $ npm install @aws-sdk/client-qldb-session --save
     $ npm install ion-js --save
     $ npm install jsbi --save
     ```

- Menggunakan versi 2.x atau 1.x

```
$ npm install aws-sdk --save
$ npm install ion-js@4.0.0 --save
$ npm install jsbi@3.1.1 --save
```

5. Buat file baru bernama`app.js` untuk JavaScript, atau`app.ts` untuk TypeScript.

   Kemudian, secara bertahap menambahkan contoh kode dalam langkah-langkah berikut untuk mencoba beberapa operasi CRUD dasar. Atau, Anda dapat melewati step-by-step tutorial dan bukannya menjalankan aplikasi lengkap.

## Langkah 2: Inisialisasi driver

Inisialisasi instance driver yang terhubung ke buku besar bernama`quick-start`. Tambahkan kode berikut ke`app.ts` file Anda`app.js` atau.

Menggunakan versi 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**
>
> - Dalam contoh kode ini, ganti *us-timur-1* denganWilayah AWS tempat Anda membuat buku besar Anda.
> - Untuk mempermudah, contoh kode yang tersisa dalam panduan ini menggunakan driver dengan pengaturan default, seperti yang ditentukan dalam contoh berikut untuk versi 1.x. Anda juga dapat menggunakan instans driver Anda sendiri dengan`RetryConfig` kustom.

Menggunakan versi 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

- Dalam contoh kode ini, ganti *us-timur-1* denganWilayah AWS tempat Anda membuat buku besar Anda.
- Versi 2.x memperkenalkan parameter opsional baru`RetryConfig` untuk inisialisasi`QldbDriver`.
- Untuk mempermudah, contoh kode yang tersisa dalam panduan ini menggunakan driver dengan pengaturan default, seperti yang ditentukan dalam contoh berikut untuk versi 1.x. Anda juga dapat menggunakan instans driver Anda sendiri dengan`RetryConfig` kustom.

- Contoh kode ini menginisialisasi driver yang menggunakan kembali koneksi yang ada dengan menetapkan opsi keep-alive. Untuk mempelajari selengkapnya, lihatRekomendasi pengaturan driver Node.js.

Menggunakan versi 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**
>
> Anda dapat mengatur variabel`AWS_REGION` lingkungan untuk menentukan Wilayah. Untuk informasi selengkapnya, lihat MengaturWilayah AWS di PanduanAWS SDK for JavaScript Developer.

## Langkah 3: Buat tabel dan indeks

Contoh kode berikut menunjukkan bagaimana menjalankan`CREATE TABLE` dan`CREATE INDEX` pernyataan.

1. Tambahkan fungsi berikut yang membuat tabel bernama`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. Tambahkan fungsi berikut yang menciptakan indeks untuk`firstName` bidang di atas`People` meja. Indeks diperlukan untuk mengoptimalkan kinerja kueri dan membantu membatasi pengecualian konflik kontrol konkurensi (OCC) yang optimis.

   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. Dalam`main` fungsi, Anda pertama kali menelepon`createTable`, dan kemudian menelepon`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. Jalankan kode untuk membuat tabel dan indeks.

JavaScript

```
$ node app.js
```

TypeScript

```
$ tsc app.ts; node app.js
```

## Langkah 4: Masukkan dokumen

Contoh kode berikut ini menunjukkan cara menjalankan`INSERT` pernyataan. QLDB mendukung bahasa kueri PartiQL (kompatibel dengan SQL) dan format data Amazon Ion (superset JSON).

1. Tambahkan fungsi berikut yang menyisipkan dokumen ke dalam`People` tabel.

   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);
   }
   ```

Contoh ini menggunakan tanda tanya (?) sebagai pengganti variabel untuk meneruskan informasi dokumen untuk pernyataan. `execute`Metode ini mendukung nilai dalam tipe Amazon Ion dan tipe asli Node.js.

> **Tip**
>
> Untuk menyisipkan beberapa dokumen dengan menggunakanSISIPKAN pernyataan tunggal, Anda dapat melewati parameter daftar jenis untuk pernyataan sebagai berikut.
>
> ```
> // people is a list
> txn.execute("INSERT INTO People ?", people);
> ```
>
> Anda tidak melampirkan variabel placeholder (?) dalam kurung sudut ganda (`<<...>>`) ketika melewati daftar. Dalam pernyataan PartiQL manual, kurung sudut ganda menunjukkan koleksi unordered dikenal sebagai tas.

2. Dalam`main` fungsi, menghapus`createTable` dan`createIndex` panggilan, dan menambahkan panggilan ke`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();
}
```

## Langkah 5: Cari Dokumen

Contoh kode berikut ini menunjukkan cara menjalankan`SELECT` pernyataan.

1. Tambahkan fungsi berikut yang query dokumen dari`People` tabel.

   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. Dalam`main` fungsi, tambahkan panggilan berikut untuk`fetchDocuments` setelah panggilan ke`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();
}
```

## Langkah 6: Perbarui dokumen

Contoh kode berikut ini menunjukkan cara menjalankanUPDATE pernyataan.

1. Tambahkan fungsi berikut yang memperbarui dokumen dalam`People` tabel dengan mengubah`lastName` ke`"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. Dalam`main` fungsi, tambahkan panggilan berikut untuk`updateDocuments` setelah panggilan ke`fetchDocuments`. Kemudian, hubungi`fetchDocuments` lagi untuk melihat hasil yang diperbarui.

   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. Jalankan kode untuk menyisipkan, query, dan memperbarui dokumen.

   JavaScript

   ```
   $ node app.js
   ```

   TypeScript

   ```
   $ tsc app.ts; node app.js
   ```

## Menjalankan aplikasi lengkap

Contoh kode berikut adalah versi lengkap`app.js` dan`app.ts`. Alih-alih melakukan langkah-langkah sebelumnya secara individual, Anda juga dapat menjalankan kode ini dari awal hingga akhir. Aplikasi ini menunjukkan beberapa operasi CRUD dasar pada buku besar bernama`quick-start`.

> **Note**
>
> Sebelum Anda menjalankan kode ini, pastikan bahwa Anda belum memiliki tabel aktif bernama`People` dalam`quick-start` buku besar.

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();
}
```

Untuk menjalankan aplikasi lengkap, masukkan perintah berikut.

JavaScript

```
$ node app.js
```

TypeScript

```
$ tsc app.ts; node app.js
```

## Driver Amazon QLDB untuk Node.js - referensi Cookbook

Panduan referensi ini menunjukkan kasus penggunaan umum driver QLDB Amazon untuk Node.js. Ini menyediakan JavaScript dan contoh TypeScript kode yang menunjukkan cara menggunakan driver untuk menjalankan operasi dasar buat, baca, dan hapus atau create, read, update, and delete (CRUD). Ini juga mencakup contoh kode untuk memproses data Amazon Ion. Selain itu, panduan ini menyoroti praktik terbaik untuk membuat transaksi idempoten dan menerapkan kendala keunikan.

Daftar Isi

## Mengimpor driver

Contoh kode berikut mengimpor 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**
>
> Contoh ini juga mengimpor paket Amazon Ion (`ion-js`). Anda memerlukan paket ini untuk memproses data Ion saat menjalankan beberapa operasi data dalam referensi ini. Untuk mempelajari selengkapnya, lihat Bekerja dengan Amazon Ion.

## Beri contoh driver

Contoh kode berikut membuat sebuah instance dari driver yang menghubungkan ke nama ledger tertentu menggunakan pengaturan default.

JavaScript

```
const qldbDriver = new qldb.QldbDriver("vehicle-registration");
```

TypeScript

```
const qldbDriver: QldbDriver = new QldbDriver("vehicle-registration");
```

## Operasi CRUD

QLDB menjalankan operasi buat, membaca, memperbarui, dan menghapus (CRUD) sebagai bagian dari transaksi.

> **Warning**
>
> Sebagai praktik terbaik, buatlah transaksi tulis Anda benar-benar idempoten.

### Melakukan transaksi idempoten

Kami menyarankan Anda melakukan transaksi tulis idempoten untuk menghindari efek samping yang tidak terduga dalam kasus percobaan ulang. Transaksi idempoten jika dapat berjalan beberapa kali dan menghasilkan hasil yang identik setiap kali.

Misalnya, pertimbangkan transaksi yang memasukkan dokumen ke dalam tabel bernama`Person`. Transaksi pertama-tama harus memeriksa apakah dokumen sudah ada dalam tabel atau tidak. Tanpa pemeriksaan ini, tabel mungkin berakhir dengan dokumen duplikat.

Misalkan QLDB berhasil melakukan transaksi di sisi server, tetapi klien kali keluar sambil menunggu respon. Jika transaksi tidak idempoten, dokumen yang sama dapat dimasukkan lebih dari sekali dalam kasus percobaan ulang.

Menggunakan indeks untuk menghindari pemindaian tabel penuh

Kami juga menyarankan Anda menjalankan pernyataan dengan klausa`WHERE` predikat menggunakan operator kesetaraan pada bidang yang diindeks atau ID dokumen; misalnya,`WHERE indexedField = 123` atau`WHERE indexedField IN (456, 789)`. Tanpa pencarian yang diindeks ini, QLDB perlu melakukan pemindaian tabel, yang dapat menyebabkan batas waktu transaksi atau konflik kontrol konkurensi (OCC) yang optimis.

Untuk informasi lebih lanjut tentang OCC, lihatModel Konkurensi Amazon QLDB.

Transaksi yang dibuat secara implisit

Metode QldbDriver.executeLambda menerima fungsi lambda yang menerima instance TransactionExecutor, yang dapat Anda gunakan untuk menjalankan pernyataan. Contoh`TransactionExecutor` membungkus transaksi yang dibuat secara implisit.

Anda dapat menjalankan pernyataan dalam fungsi lambda dengan menggunakan metode eksekusi dari pelaksana transaksi. Pengemudi secara implisit melakukan transaksi ketika fungsi lambda kembali.

> **Note**
>
> `execute`Metode ini mendukung tipe Amazon Ion dan tipe asli Node.js. Jika Anda meneruskan tipe asli Node.js sebagai argumen`execute`, driver mengubahnya menjadi tipe Ion menggunakan`ion-js` paket (asalkan konversi untuk tipe data Node.js yang diberikan didukung). Untuk tipe data dan aturan konversi yang didukung, lihat Ion JavaScript DOM README.

Bagian berikut menunjukkan cara menjalankan operasi CRUD dasar, menentukan logika percobaan ulang kustom, dan menerapkan batasan keunikan.

Daftar Isi

### Membuat Tabel

#### 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');
    });
}());
```

### Membuat indeks

#### 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)');
    });
}());
```

### Membaca dokumen

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']
        }
    });
}());
```

### Menggunakan parameter kueri

Contoh kode berikut menggunakan parameter jenis query asli.

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']
        }
    });
}());
```

Contoh kode berikut menggunakan parameter kueri tipe 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']
        }
    });
}());
```

Contoh kode berikut menggunakan beberapa parameter query.

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']
        }
    });
}());
```

Contoh kode berikut menggunakan daftar parameter query.

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**
>
> Ketika Anda menjalankan kueri tanpa pencarian yang diindeks, itu akan memanggil pemindaian tabel penuh. Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, kueri dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

### Memasukkan dokumen

Contoh kode berikut menyisipkan tipe data asli.

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

Contoh kode berikut menyisipkan tipe data 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);
        }
    });
}());
```

Transaksi ini menyisipkan dokumen ke dalam`Person` tabel. Sebelum memasukkan, pertama-tama memeriksa apakah dokumen sudah ada di tabel. Cek ini membuat transaksi idempoten di alam. Bahkan jika Anda menjalankan transaksi ini beberapa kali, itu tidak akan menyebabkan efek samping yang tidak diinginkan.

> **Note**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

Memasukkan beberapa dokumen dalam satu pernyataan

Untuk menyisipkan beberapa dokumen dengan menggunakanSISIPKAN pernyataan tunggal, Anda dapat melewati parameter daftar jenis untuk pernyataan sebagai berikut.

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

Anda tidak melampirkan variabel placeholder (?) dalam kurung sudut ganda (<<...>>) ketika melewati daftar. Dalam pernyataan PartiQL manual, kurung sudut ganda menunjukkan koleksi unordered dikenal sebagai tas.

Memperbarui dokumen

Contoh kode berikut menggunakan tipe data asli.

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');
    });
}());
```

Contoh kode berikut menggunakan tipe data 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**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

### Menghapus dokumen

Contoh kode berikut menggunakan tipe data asli.

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');
    });
}());
```

Contoh kode berikut menggunakan tipe data 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**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

Menjalankan beberapa pernyataan dalam transaksi

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;
    });
};
```

Logika coba lagi

`executeLambda`Metode pengemudi memiliki mekanisme coba ulang bawaan yang mencoba ulang transaksi jika terjadi pengecualian yang dapat dicoba ulang (seperti batas waktu atau konflik OCC). Jumlah maksimum upaya coba lagi dan strategi backoff dapat dikonfigurasi.

Batas percobaan ulang default adalah`4`, dan strategi backoff default adalah [defaultBackoffFunction](#)dengan basis`10` milidetik. Anda dapat mengatur konfigurasi coba ulang per contoh driver dan juga per transaksi dengan menggunakan instance [RetryConfig](#).

Contoh kode berikut menentukan logika coba lagi dengan batas percobaan ulang kustom dan strategi cadangan khusus untuk instance 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);
```

Contoh kode berikut menetapkan logika percobaan ulang dengan batas percobaan ulang kustom dan strategi cadangan khusus untuk eksekusi lambda tertentu. Konfigurasi ini untuk`executeLambda` menimpa logika coba ulang yang diatur untuk instance 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);
}());
```

Menerapkan kendala keunikan

QLDB tidak mendukung indeks unik, tetapi Anda dapat menerapkan perilaku ini dalam aplikasi Anda.

Misalkan Anda ingin menerapkan kendala keunikan pada`GovId` bidang dalam`Person` tabel. Untuk melakukan ini, Anda dapat menulis transaksi yang melakukan hal berikut:

1. Menegaskan bahwa tabel tidak memiliki dokumen yang ada dengan ditentukan`GovId`.
2. Masukkan dokumen jika pernyataan berlalu.

Jika transaksi yang bersaing secara bersamaan melewati pernyataan, hanya satu dari transaksi yang akan berhasil dilakukan. Transaksi lainnya akan gagal dengan pengecualian konflik OCC.

Contoh kode berikut menunjukkan cara menerapkan logika kendala keunikan ini.

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**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

## Bekerja dengan Amazon Ion

Bagian berikut menunjukkan cara menggunakan modul Amazon Ion untuk memproses data Ion.

Daftar Isi

### Mengimpor modul Ion

JavaScript

```
var ionjs = require('ion-js');
```

TypeScript

```
import { dom, dumpBinary, dumpText, load } from "ion-js";
```

### Membuat jenis Ion

Contoh kode berikut membuat objek Ion dari teks 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']
```

Contoh kode berikut membuat objek Ion dari kamus 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']
```

### Mendapatkan dump biner 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
```

### Mendapatkan dump teks 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"}
```

Untuk informasi selengkapnya tentang Ion, lihat dokumentasi Amazon Ion di GitHub. Untuk contoh kode lebih lanjut bekerja dengan Ion di QLDB, lihatBekerja dengan tipe data Amazon Ion di Amazon QLDB.

# Driver Amazon QLDB untuk Python

Untuk bekerja dengan data dalam buku besar Anda, Anda dapat terhubung ke Amazon QLDB dari aplikasi Python Anda dengan menggunakan driver yangAWS disediakan. Topik berikut menjelaskan cara memulai dengan driver QLDB.

Topik



## Sumber daya driver

Untuk informasi selengkapnya tentang fungsionalitas yang didukung oleh driver Python, lihat sumber daya berikut:

- Referensi API: 3.x, 2.x
- Kode sumber driver (GitHub)
- Contoh kode sumber aplikasi (GitHub)
- Contoh kode Amazon Ion

## Prasyarat

Sebelum memulai dengan driver QLDB untuk Python, Anda harus melakukan hal berikut:

1. Ikuti petunjukAWS pengaturan diMengakses Amazon QLDB. Ini termasuk yang berikut:
    1. Daftar ke AWS.
    2. Buat pengguna dengan izin QLDB yang sesuai.

3. Memberikan akses terprogram untuk pengembangan.

2. Instal salah satu versi Python berikut dari situs unduhan Python:

   - 3.6 atau yang lebih baru — Driver QLDB untuk Python v3
   - 3.4 atau yang lebih baru — driver QLDB untuk Python v2

3. SiapkanAWS kredensi dan default AndaWilayah AWS. Untuk petunjuknya, lihat Mulai Cepat diAWS SDK for Python (Boto3) dokumentasi.

   Untuk daftar lengkap Wilayah yang tersedia, lihat titik akhir dan kuota Amazon QLDB di bagian Referensi Umum AWS.

Selanjutnya, Anda dapat mengunduh aplikasi contoh tutorial lengkap—atau Anda hanya dapat menginstal driver dalam proyek Python dan menjalankan contoh kode pendek.

- Untuk menginstal driver QLDB danAWS SDK for Python (Boto3) dalam proyek yang ada, lanjutkan keInstalasi.
- Untuk menyiapkan proyek dan menjalankan contoh kode pendek yang menunjukkan transaksi data dasar pada buku besar, lihatTutorial Quick Start.
- Untuk menjalankan contoh yang lebih mendalam dari kedua data dan operasi API manajemen dalam aplikasi sampel tutorial lengkap, lihatTutorial Python.

## Instalasi

QLDB mendukung versi driver berikut dan dependensi Python mereka.

| Versi driver | Versi Python | Status | Tanggal rilis terbaru |
|---|---|---|---|
| 2.x | 3.4 atau yang lebih baru | Rilis produksi | 7 Mei 2020 |
| 3.x | 3.6 atau yang lebih baru | Rilis produksi | 28 Oktober 2021 |

Untuk menginstal driver QLDB dari PyPI menggunakan`pip` (manajer paket untuk Python), masukkan berikut di baris perintah.

3.x

```
pip install pyqldb
```

2.x

```
pip install pyqldb==2.0.2
```

Menginstal driver juga menginstal dependensinya, termasuk paket AWS SDK for Python (Boto3)dan Amazon Ion.

Menggunakan driver untuk terhubung ke buku besar

Kemudian Anda dapat mengimpor driver dan menggunakannya untuk terhubung ke buku besar. Contoh kode Python berikut ini menunjukkan cara membuat sesi untuk nama buku besar.

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)
```

Untuk contoh kode singkat tentang cara menjalankan transaksi data dasar pada buku besar, lihatReferensi buku masak.

# Driver Amazon QLDB untuk Python - Tutorial mulai cepat

Dalam tutorial ini, Anda belajar cara mengatur aplikasi sederhana menggunakan versi terbaru dari driver Amazon QLDB untuk Python. Panduan ini mencakup langkah-langkah untuk menginstal driver dan contoh kode pendek dari operasi dasar buat, baca, perbarui, dan hapus atau create, read, update, and delete (CRUD). Untuk contoh yang lebih mendalam yang menunjukkan operasi ini dalam aplikasi sampel lengkap, lihatTutorial Python.

Topik



## Prasyarat

Sebelum memulai, pastikan Anda melakukan hal berikut:

1. SelesaikanPrasyarat untuk driver Python, jika Anda belum melakukannya. Ini termasuk mendaftarAWS, memberikan akses terprogram untuk pengembangan, dan menginstal Python versi 3.6 atau yang lebih baru.
2. Buat buku besar bernama`quick-start`.

   Untuk mempelajari cara membuat buku besar, lihatOperasi dasar untuk buku besar Amazon QLDB atauLangkah 1: Membuat buku besar baru di Memulai konsol.

## Langkah 1: Siapkan proyek Anda

Pertama, siapkan proyek Python Anda.

> **Note**
>
> Jika Anda menggunakan IDE yang memiliki fitur untuk mengotomatiskan langkah-langkah penyiapan ini, Anda dapat melompat ke depanLangkah 2: Inisialisasi driver.

1. Membuat folder untuk aplikasi Anda.

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

2. Untuk menginstal driver QLDB untuk Python dari PyPI, masukkan`pip` perintah berikut.

   ```
   $ pip install pyqldb
   ```

   Menginstal driver juga menginstal dependensinya, termasuk paket AWS SDK for Python (Boto3)dan Amazon Ion.

3. Buat file baru dengan nama `app.py`.

   Kemudian, secara bertahap menambahkan contoh kode dalam langkah-langkah berikut untuk mencoba beberapa operasi CRUD dasar. Atau, Anda dapat melewati step-by-step tutorial dan sebagai gantinya menjalankan aplikasi lengkap.

## Langkah 2: Inisialisasi driver

Inisialisasi instance driver yang terhubung ke buku besar bernama`quick-start`. Tambahkan kode berikut ke`app.py` file Anda.

```
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)
```

## Langkah 3: Buat tabel dan indeks

Contoh kode berikut ini menunjukkan cara menjalankan`CREATE TABLE` dan`CREATE INDEX` pernyataan.

Tambahkan kode berikut yang membuat tabel bernama`People` dan indeks untuk`lastName` bidang pada tabel itu. Indeks diperlukan untuk mengoptimalkan kinerja kueri dan membantu membatasi pengecualian konflik kontrol konkurensi (OCC) yang optimis.

```
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))
```

## Langkah 4: Masukkan dokumen

Contoh kode berikut ini menunjukkan cara menjalankan`INSERT` pernyataan. QLDB mendukung bahasa kueri PartiQL (kompatibel dengan SQL) dan format data Amazon Ion (superset JSON).

Tambahkan kode berikut yang menyisipkan dokumen ke dalam`People` tabel.

```
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))
```

Contoh ini menggunakan tanda tanya (?) sebagai placeholder variabel untuk meneruskan informasi dokumen untuk pernyataan. `execute_statement`Metode ini mendukung nilai dalam tipe Amazon Ion dan tipe asli Python.

> **Tip**
>
> Untuk menyisipkan beberapa dokumen dengan menggunakanSISIPKAN pernyataan tunggal, Anda dapat melewati parameter daftar jenis untuk pernyataan sebagai berikut.
>
> ```
> # people is a list
> transaction_executor.execute_statement("INSERT INTO Person ?", people)
> ```
>
> Anda tidak melampirkan variabel placeholder (?) dalam kurung sudut ganda (`<<...>>`) ketika melewati daftar. Dalam pernyataan PartiQL manual, kurung sudut ganda menunjukkan koleksi unordered dikenal sebagai tas.

## Langkah 5: Cari Document

Contoh kode berikut ini menunjukkan cara menjalankan`SELECT` pernyataan.

Tambahkan kode berikut yang meminta dokumen dari`People` tabel.

```
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))
```

## Langkah 6: Perbarui dokumen

Contoh kode berikut ini menunjukkan cara menjalankan`UPDATE` pernyataan.

1. Tambahkan kode berikut yang memperbarui dokumen dalam`People` tabel dengan memperbaruui`age` ke`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. Kueri tabel lagi untuk melihat nilai yang diperbarui.

```
# Query the updated document
qldb_driver.execute_lambda(lambda executor: read_documents(executor))
```

3. Untuk menjalankan aplikasi, masukkan perintah berikut dari direktori proyek Anda.

```
$ python app.py
```

## Menjalankan aplikasi lengkap

Contoh kode berikut adalah versi lengkap dari`app.py` aplikasi. Alih-alih melakukan langkah-langkah sebelumnya secara individual, Anda juga dapat menyalin dan menjalankan contoh kode ini dari awal hingga akhir. Aplikasi ini menunjukkan beberapa operasi CRUD dasar pada buku besar bernama`quick-start`.

> **Note**
>
> Sebelum Anda menjalankan kode ini, pastikan bahwa Anda belum memiliki tabel aktif bernama`People` dalam`quick-start` buku besar.

```
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))
```

Untuk menjalankan aplikasi lengkap, masukkan perintah berikut dari direktori proyek Anda.

```
$ python app.py
```

## Driver Amazon QLDB untuk Python - referensi Cookbook

Panduan referensi ini menunjukkan kasus penggunaan umum driver QLDB Amazon untuk Python. Ini memberikan contoh kode Python yang menunjukkan cara menggunakan driver untuk menjalankan operasi dasar buat, baca, perbarui, dan hapus atau (CRUD). Ini juga mencakup contoh kode untuk memproses data Amazon Ion. Selain itu, panduan ini menyoroti praktik terbaik untuk membuat transaksi idempoten dan menerapkan kendala keunikan.

> **Note**
>
> Jika berlaku, beberapa kasus penggunaan memiliki contoh kode yang berbeda untuk setiap versi utama driver QLDB yang didukung untuk Python.

Daftar Isi

## Mengimpor driver

Contoh kode berikut mengimpor 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**
>
> Contoh ini juga mengimpor paket Amazon Ion (`amazon.ion.simpleion`). Anda memerlukan paket ini untuk memproses data Ion saat menjalankan beberapa operasi data dalam referensi ini. Untuk mempelajari selengkapnya, lihat Bekerja dengan Amazon Ion.

## Beri contoh driver

Contoh kode berikut membuat sebuah instance dari driver yang menghubungkan ke nama ledger tertentu menggunakan pengaturan default.

3.x

```
qldb_driver = QldbDriver(ledger_name='vehicle-registration')
```

2.x

```
qldb_driver = PooledQldbDriver(ledger_name='vehicle-registration')
```

## Operasi CRUD

QLDB menjalankan operasi buat, baca, dan hapus (CRUD) sebagai bagian dari transaksi.

> **Warning**
>
> Untuk praktik terbaik, buatlah transaksi tulis Anda benar-benar idempoten.

### Melakukan transaksi idempoten

Kami menyarankan Anda melakukan transaksi tulis idempoten untuk menghindari efek samping yang tidak terduga dalam kasus percobaan ulang. Transaksi idempoten jika dapat berjalan beberapa kali dan menghasilkan hasil yang identik setiap kali.

Misalnya, pertimbangkan transaksi yang memasukkan dokumen ke tabel bernama`Person`. Transaksi pertama-tama harus memeriksa apakah dokumen sudah ada dalam tabel atau tidak. Tanpa pemeriksaan ini, tabel mungkin berakhir dengan dokumen duplikat.

Misalkan QLDB berhasil melakukan transaksi di sisi server, tetapi klien kali keluar sambil menunggu respon. Jika transaksi tidak idempoten, dokumen yang sama dapat dimasukkan lebih dari sekali dalam kasus percobaan ulang.

Menggunakan indeks untuk menghindari pemindaian tabel penuh

Kami juga menyarankan Anda menjalankan pernyataan dengan klausa`WHERE` predikat menggunakan operator kesetaraan pada bidang yang diindeks atau ID dokumen; misalnya,`WHERE indexedField = 123` atau`WHERE indexedField IN (456, 789)`. Tanpa pencarian yang diindeks ini, QLDB perlu melakukan pemindaian tabel, yang dapat menyebabkan batas waktu transaksi atau konflik kontrol konkurensi (OCC) yang optimis.

Untuk informasi tentang OCC, lihatModel Konkurensi Amazon QLDB.

Transaksi yang dibuat secara implisit

Metode pyqldb.driver.qldb_driver.execute_lambda menerima fungsi lambda yang menerima instance PyQLDB.Execution.Executor.Executor, yang dapat Anda gunakan untuk menjalankan pernyataan. Contoh`Executor` membungkus transaksi yang dibuat secara implisit.

Anda dapat menjalankan pernyataan dalam fungsi lambda dengan menggunakan metode execute_statement dari pelaksana transaksi. Pengemudi secara implisit melakukan transaksi ketika fungsi lambda kembali.

> **Note**
>
> `execute_statement`Metode ini mendukung tipe Amazon Ion dan tipe asli Python. Jika Anda meneruskan tipe asli Python sebagai argumen`execute_statement`, driver mengubahnya menjadi tipe Ion menggunakan`amazon.ion.simpleion` modul (asalkan konversi untuk tipe data Python yang diberikan didukung). Untuk tipe data dan aturan konversi yang didukung, lihat kode sumber simpleion.

Bagian berikut menunjukkan cara menjalankan operasi CRUD dasar, menentukan logika percobaan ulang kustom, dan menerapkan batasan keunikan.

Daftar Isi

### Membuat Tabel

```
def create_table(transaction_executor):
    transaction_executor.execute_statement("CREATE TABLE Person")

qldb_driver.execute_lambda(lambda executor: create_table(executor))
```

### Membuat indeks

```
def create_index(transaction_executor):
    transaction_executor.execute_statement("CREATE INDEX ON Person(GovId)")

qldb_driver.execute_lambda(lambda executor: create_index(executor))
```

### Membaca dokumen

```
# 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))
```

### Menggunakan parameter kueri

Contoh kode berikut menggunakan parameter jenis query asli.

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId = ?",
 'TOYENC486FH')
```

Contoh kode berikut menggunakan parameter kueri tipe Ion.

```
name = ion.loads('Brent')
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE FirstName
 = ?", name)
```

Contoh kode berikut menggunakan beberapa parameter query.

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId = ?
 AND FirstName = ?", 'TOYENC486FH', "Brent")
```

Contoh kode berikut menggunakan daftar parameter query.

```
gov_ids = ['TOYENC486FH','ROEE1','YH844']
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId IN
 (?,?,?)", *gov_ids)
```

> **Note**
>
> Ketika Anda menjalankan kueri tanpa pencarian yang diindeks, itu akan memanggil pemindaian tabel penuh. Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, kueri dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

### Memasukkan dokumen

Contoh kode berikut menyisipkan tipe data asli.

```
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))
```

Contoh kode berikut menyisipkan tipe data 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))
```

Transaksi ini menyisipkan dokumen ke dalam`Person` tabel. Sebelum memasukkan, pertama-tama memeriksa apakah dokumen sudah ada di tabel. Cek ini membuat transaksi idempoten di alam. Bahkan jika Anda menjalankan transaksi ini beberapa kali, itu tidak akan menyebabkan efek samping yang tidak diinginkan.

> **Note**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

Memasukkan beberapa dokumen dalam satu pernyataan

Untuk menyisipkan beberapa dokumen dengan menggunakanSISIPKAN pernyataan tunggal, Anda dapat melewati parameter daftar jenis untuk pernyataan sebagai berikut.

```
# people is a list
transaction_executor.execute_statement("INSERT INTO Person ?", people)
```

Anda tidak melampirkan variabel placeholder (?) dalam kurung sudut ganda (<<...>>) ketika melewati daftar. Dalam pernyataan PartiQL manual, kurung sudut ganda menunjukkan koleksi unordered dikenal sebagai tas.

Memperbarui dokumen

Contoh kode berikut menggunakan tipe data asli.

```
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))
```

Contoh kode berikut menggunakan tipe data 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**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

### Menghapus dokumen

Contoh kode berikut menggunakan tipe data asli.

```
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))
```

Contoh kode berikut menggunakan tipe data 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**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

Menjalankan beberapa pernyataan dalam transaksi

```
# 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))
```

Logika coba lagi

`execute_lambda`Metode pengemudi memiliki mekanisme coba ulang bawaan yang mencoba ulang transaksi jika terjadi pengecualian yang dapat dicoba ulang (seperti batas waktu atau konflik OCC).

3.x

Jumlah maksimum upaya coba lagi dan strategi backoff dapat dikonfigurasi.

Batas percobaan ulang default adalah`4`, dan strategi backoff default adalah Exponential Backoff dan Jitter dengan basis`10` milidetik. Anda dapat mengatur konfigurasi coba ulang per instance driver dan juga per transaksi dengan menggunakan instance pyqldb.config.retry_config. RetryConfig.

Contoh kode berikut menentukan logika coba lagi dengan batas percobaan ulang kustom dan strategi cadangan khusus untuk instance 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)
```

Contoh kode berikut menetapkan logika percobaan ulang dengan batas percobaan ulang kustom dan strategi cadangan khusus untuk eksekusi lambda tertentu. Konfigurasi ini untuk`execute_lambda` menimpa logika coba ulang yang diatur untuk instance 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

Jumlah maksimum upaya coba lagi dapat dikonfigurasi. Anda dapat mengkonfigurasi batas coba lagi dengan menetapkan`retry_limit` properti saat menginisialisasi`PooledQldbDriver`.

Batas percobaan ulang default adalah4.

### Menerapkan kendala keunikan

QLDB tidak mendukung indeks unik, tetapi Anda dapat menerapkan perilaku ini dalam aplikasi Anda.

Misalkan Anda ingin menerapkan kendala keunikan pada`GovId` bidang dalam`Person` tabel. Untuk melakukan ini, Anda dapat menulis transaksi yang melakukan hal berikut:

1. Menegaskan bahwa tabel tidak memiliki dokumen yang ada dengan yang ditentukan`GovId`.
2. Masukkan dokumen jika pernyataan berlalu.

Jika transaksi yang bersaing secara bersamaan melewati pernyataan, hanya satu dari transaksi yang akan berhasil dilakukan. Transaksi lainnya akan gagal dengan pengecualian konflik OCC.

Contoh kode berikut menunjukkan cara menerapkan logika kendala keunikan ini.

```
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**
>
> Dalam contoh ini, kami sarankan memiliki indeks di`GovId` lapangan untuk mengoptimalkan kinerja. Tanpa indeks aktif`GovId`, pernyataan dapat memiliki lebih banyak latensi dan juga dapat menyebabkan pengecualian konflik OCC atau batas waktu transaksi.

## Bekerja dengan Amazon Ion

Bagian berikut menunjukkan cara menggunakan modul Amazon Ion untuk memproses data Ion.

Daftar Isi



### Mengimpor modul Ion

```
import amazon.ion.simpleion as simpleion
```

### Membuat jenis Ion

Contoh kode berikut membuat objek Ion dari teks 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
```

Contoh kode berikut membuat objek Ion dari 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
```

Mendapatkan dump biner 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'
```

Mendapatkan dump teks Ion

```
# ion_obj is an Ion struct
print(simpleion.dumps(ion_obj, binary=False)) # prints $ion_1_0
 {GovId:'TOYENC486FH',FirstName:"Brent"}
```

Untuk informasi tentang bekerja dengan Ion, lihat dokumentasi Amazon Ion GitHub. Untuk contoh kode lebih lanjut bekerja dengan Ion di QLDB, lihatBekerja dengan tipe data Amazon Ion di Amazon QLDB.

# Memahami manajemen sesi dengan driver di Amazon QLDB

Jika Anda memiliki pengalaman menggunakan sistem manajemen database relasional (RDBMS), Anda mungkin akrab dengan koneksi bersamaan. QLDB tidak memiliki konsep yang sama dari koneksi RDBMS tradisional karena transaksi dijalankan dengan permintaan HTTP dan pesan respon.

Dalam QLDB, konsep analog adalah sesi aktif. Sesi secara konseptual mirip dengan login pengguna—ia mengelola informasi tentang permintaan transaksi data Anda ke buku besar. Sesi aktif adalah sesi yang secara aktif menjalankan transaksi. Ini juga bisa menjadi sesi yang baru-baru ini menyelesaikan transaksi di mana layanan mengantisipasi akan segera memulai transaksi lain. QLDB mendukung satu transaksi yang berjalan secara aktif per sesi.

Batas sesi aktif bersamaan per buku besar didefinisikan dalamKuota dan batasan di Amazon QLDB. Setelah batas ini tercapai, setiap sesi yang mencoba memulai transaksi akan mengakibatkan error (`LimitExceededException`).

Untuk praktik terbaik untuk mengonfigurasi kumpulan sesi dalam aplikasi Anda menggunakan driver QLDB, lihatMengonfigurasi QldbDriver objek di rekomendasi driver Amazon QLDB.

Daftar Isi

## Siklus hidup sesi

Urutan operasi API Sesi QLDB berikut mewakili siklus hidup khas sesi QLDB:

1. `StartSession`
2. `StartTransaction`
3. `ExecuteStatement`
4. `CommitTransaction`
5. Ulangi langkah 2—4 untuk memulai lebih banyak transaksi (satu transaksi pada satu waktu).
6. `EndSession`

## Kedaluwarsa sesi

QLDB kedaluwarsa dan membuang sesi setelah total masa hidup 13-17 menit, terlepas dari apakah itu aktif menjalankan transaksi. Sesi dapat hilang atau terganggu karena sejumlah alasan, seperti kegagalan perangkat keras, kegagalan jaringan, atau restart aplikasi. QLDB memberlakukan seumur hidup maksimum pada sesi untuk memastikan bahwa aplikasi klien tangguh terhadap kegagalan sesi.

## Penanganan sesi di driver QLDB

Meskipun Anda dapat menggunakanAWS SDK untuk berinteraksi langsung dengan QLDB Session API, ini menambah kompleksitas dan mengharuskan Anda untuk menghitung commit digest. QLDB menggunakan commit digest ini untuk memastikan integritas transaksi. Alih-alih berinteraksi langsung dengan API ini, sebaiknya gunakan driver QLDB.

Pengemudi menyediakan lapisan abstraksi tingkat tinggi di atas API data transaksional. Ini merampingkan proses menjalankan pernyataan PartiQL pada data buku besar dengan mengelola panggilan SendCommandAPI. Panggilan API ini memerlukan beberapa parameter yang ditangani

driver untuk Anda, termasuk pengelolaan sesi, transaksi, dan kebijakan coba lagi jika terjadi kesalahan.

Topik



## Gambaran umum sesi

Dalam versi driver QLDB (misalnya, Java v1.1.0), kami menyediakan dua implementasi dari objek driver: standar, nonpooled`QldbDriver` dan`PooledQldbDriver`. Seperti namanya,`PooledQldbDriver` mempertahankan kumpulan sesi yang digunakan kembali di seluruh transaksi.

Berdasarkan umpan balik pengguna, pengembang lebih memilih untuk mendapatkan fitur penggabungan dan manfaatnya secara default, daripada menggunakan driver standar. Jadi, kami menghapus`PooledQldbDriver` dan memindahkan fungsionalitas penggabungan sesi ke`QldbDriver`. Perubahan ini termasuk dalam rilis terbaru setiap driver (misalnya, Java v2.0.0).

Pengemudi menyediakan tiga tingkat abstraksi:

- Driver (implementasi driver yang dikumpulkan) - Abstraksi tingkat atas. Pengemudi memelihara dan mengelola kumpulan sesi. Ketika Anda meminta pengemudi untuk menjalankan transaksi, pengemudi memilih sesi dari pool dan menggunakannya untuk menjalankan transaksi. Jika transaksi gagal karena kesalahan sesi (`InvalidSessionException`), pengemudi menggunakan sesi lain untuk mencoba kembali transaksi. Pada dasarnya, pengemudi menawarkan pengalaman sesi yang dikelola sepenuhnya.
- Sesi - Satu tingkat di bawah abstraksi driver. Sesi terkandung dalam kolam renang, dan pengemudi mengelola siklus hidup sesi. Jika transaksi gagal, pengemudi akan mencoba ulang hingga sejumlah upaya tertentu menggunakan sesi yang sama. Tetapi jika sesi menemukan error (`InvalidSessionException`), QLDB membuangnya secara internal. Pengemudi kemudian bertanggung jawab untuk mendapatkan sesi lain dari pool untuk mencoba kembali transaksi.
- Transaksi - Tingkat abstraksi terendah. Transaksi terkandung dalam sesi, dan sesi mengelola siklus hidup transaksi. Sesi ini bertanggung jawab untuk mencoba kembali transaksi jika terjadi kesalahan. Sesi ini juga memastikan bahwa itu tidak membocorkan transaksi terbuka yang belum dilakukan atau dibatalkan.

Dalam versi terbaru dari setiap driver, Anda dapat melakukan operasi pada tingkat driver abstraksi saja. Anda tidak memiliki kontrol langsung atas sesi dan transaksi individual (yaitu, tidak ada operasi API untuk memulai sesi atau transaksi baru secara manual).

## Penggabungan sesi dan logika transaksi

Rilis terbaru dari setiap driver tidak lagi menyediakan implementasi nonpooled dari objek driver. Secara default,`QldbDriver` objek mengelola kolam sesi. Ketika Anda melakukan panggilan untuk menjalankan transaksi, pengemudi melakukan langkah-langkah berikut:

1. Pengemudi memeriksa apakah telah mencapai batas sesi-pool. Jika demikian, pengemudi segera melempar`NoSessionAvailable` pengecualian. Jika tidak, dilanjutkan ke langkah berikutnya.
2. Pengemudi memeriksa apakah kolam memiliki sesi yang tersedia.
   - Jika sesi tersedia di pool, pengemudi menggunakannya untuk menjalankan transaksi.
   - Jika tidak ada sesi yang tersedia di pool, driver membuat sesi baru dan menggunakannya untuk menjalankan transaksi.
3. Setelah pengemudi mendapat sesi pada langkah 2, pengemudi memanggil`execute` operasi pada instance sesi.
4. Dalam`execute` pengoperasian sesi, pengemudi mencoba memulai transaksi dengan menelepon`startTransaction`.
   - Jika sesi tidak valid,`startTransaction` panggilan gagal, dan driver kembali ke langkah 1.
   - Jika`startTransaction` panggilan berhasil, pengemudi dilanjutkan ke langkah berikutnya.
5. Pengemudi menjalankan ekspresi lambda Anda. Ekspresi lambda ini dapat berisi satu atau lebih panggilan untuk menjalankan pernyataan PartiQL. Setelah ekspresi lambda selesai berjalan tanpa kesalahan, pengemudi melanjutkan untuk melakukan transaksi.
6. Komit transaksi dapat memiliki satu dari dua hasil:
   - Komit berhasil, dan driver mengembalikan kontrol ke kode aplikasi Anda.
   - Komit gagal karena konflik kontrol konkurensi (OCC) yang optimis. Dalam hal ini, driver mencoba ulang langkah 4—6 menggunakan sesi yang sama. Jumlah maksimum upaya coba lagi yang akan dilakukan dalam kode aplikasi Anda. Batas default adalah`4`.

> **Note**
>
> Jika`InvalidSessionException` dilemparkan selama langkah 4—6, driver menandai sesi sebagai tertutup dan kembali ke langkah 1 untuk mencoba kembali transaksi.
> Jika ada pengecualian lain yang dilemparkan selama langkah 4—6, pengemudi memeriksa apakah pengecualian dapat dicoba ulang. Jika demikian, ia mencoba ulang transaksi hingga jumlah upaya coba lagi yang ditentukan. Jika tidak, itu menyebarkan (gelembung dan melempar) pengecualian untuk kode aplikasi Anda.

### Mengembalikan sesi ke kolam renang

Jika`InvalidSessionException` terjadi kapan saja selama transaksi aktif, pengemudi tidak mengembalikan sesi ke pool. QLDB membuang sesi sebagai gantinya, dan pengemudi mendapat sesi lain dari kolam renang. Dalam semua kasus lain, pengemudi mengembalikan sesi ke kolam renang.

## Rekomendasi pengemudi Amazon QLDB

Bagian ini menjelaskan praktik terbaik untuk mengonfigurasi dan menggunakan driver Amazon QLDB untuk bahasa apa pun yang didukung. Contoh kode yang disediakan khusus untuk Java.

Rekomendasi ini berlaku untuk kasus penggunaan yang paling umum, tetapi satu ukuran tidak cocok untuk semua. Gunakan rekomendasi berikut sesuai keinginan Anda untuk aplikasi Anda.

Topik



### Mengonfigurasi QldbDriver objek

Parameter`QldbDriver`objek mengelola koneksi ke buku besar Anda dengan mempertahankan kolamsidangyang digunakan kembali di seluruh transaksi. SEBUAHsesimewakili koneksi tunggal ke buku besar. QLDB mendukung satu transaksi aktif berjalan per sesi.

**Important**

Untuk versi driver yang lebih lama, fungsi penyatuan sesi masih dalam`PooledQldbDriver`objek bukan`QldbDriver`. Jika Anda menggunakan salah satu versi berikut, ganti apa pun`QldbDriver`bersama`PooledQldbDriver`untuk sisa topik ini.

| Driver | Versi |
| --- | --- |
| Java | `1.1.0`atau sebelumnya |
| .NET | `0.1.0-beta` |
| Node.js | `1.0.0-rc.1` atau sebelumnya |
| Python | `2.0.2`atau sebelumnya |

Parameter`PooledQldbDriver`tidak digunakan lagi dalam versi terbaru dari driver. Kami merekomendasikan agar Anda melakukan peningkatan ke versi terbaru`PooledQldbDriver`kepada`QldbDriver`.

Konfigurasi QldbDriver sebagai objek global

Untuk mengoptimalkan penggunaan driver dan sesi, pastikan bahwa hanya satu contoh global driver yang ada di instans aplikasi Anda. Misalnya di Jawa, Anda dapat menggunakaninjeksi ketergantunganKerangka kerja seperti[Spring](),[Google Guice](), atau[Belati](). Contoh kode berikut menunjukkan cara mengkonfigurasi`QldbDriver`sebagai 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();
}
```

Konfigurasikan upaya coba ulang

Pengemudi secara otomatis mencoba kembali transaksi ketika pengecualian transien umum (sepertiSocketTimeoutExceptionatauNoHttpResponseException) terjadi. Untuk mengatur jumlah maksimum upaya coba lagi, Anda dapat menggunakanmaxRetriesparametertransactionRetryPolicyobjek konfigurasi saat membuat sebuah instanceQldbDriver. (Untuk versi driver yang lebih lama seperti yang tercantum di bagian sebelumnya, gunakanretryLimitparameterPooledQldbDriver.)

Nilai default maxRetries adalah 4.

Kesalahan sisi klien sepertiInvalidParameterExceptiontidak bisa dicoba lagi. Ketika mereka terjadi, transaksi dibatalkan, sesi dikembalikan ke kolam renang, dan pengecualian dilemparkan ke klien pengemudi.

Mengonfigurasi jumlah maksimum sesi dan transaksi

Jumlah maksimum sesi buku besar yang digunakan oleh instanceQldbDriveruntuk menjalankan transaksi didefinisikan olehmaxConcurrentTransactionsparameter. (Untuk versi driver yang lebih lama seperti yang tercantum di bagian sebelumnya, ini didefinisikan olehpoolLimitparameterPooledQldbDriver.)

Batas ini harus lebih besar dari nol dan kurang dari atau sama dengan jumlah maksimum koneksi HTTP terbuka yang memungkinkan klien sesi, sebagaimana didefinisikan oleh spesifikAWSSDK. Misalnya di Jawa, jumlah maksimum koneksi diatur dalamClientConfigurationobjek.

Nilai defaultmaxConcurrentTransactionsadalah pengaturan koneksi maksimumAWSSDK.

Ketika Anda mengkonfigurasiQldbDriverdalam aplikasi Anda, ambil pertimbangan penskalaan berikut:

- Kolam renang Anda harus selalu memiliki setidaknya sebanyak jumlah transaksi yang berjalan secara bersamaan yang Anda rencanakan.

- Dalam model multi-threaded di mana thread supervisor mendelegasikan ke thread pekerja, pengemudi harus memiliki setidaknya sebanyak sesi jumlah thread pekerja. Jika tidak, pada beban puncak, benang akan mengantri untuk sesi yang tersedia.
- Batas layanan sesi aktif bersamaan per buku besar didefinisikan dalamKuota dan batasan di Amazon QLDB. Pastikan bahwa Anda tidak mengkonfigurasi lebih dari batas sesi bersamaan ini untuk digunakan untuk satu buku besar di semua klien.

## Mencoba kembali pada pengecualian

Ketika mencoba kembali pada pengecualian yang terjadi di QLDB, mempertimbangkan rekomendasi berikut.

Mencoba kembali pada OCCConflictException

Kontrol konkurensi yang optimis(OCC) pengecualian konflik terjadi ketika data yang transaksi mengakses telah berubah sejak awal transaksi. QLDB melempar pengecualian ini ketika mencoba untuk melakukan transaksi. Pengemudi mencoba kembali transaksi hingga sebanyak`maxRetries`dikonfigurasi.

Untuk informasi lebih lanjut tentang OCC dan praktik terbaik untuk menggunakan indeks untuk membatasi konflik OCC, lihatModel Konkurensi Amazon QLDB.

Mencoba ulang pengecualian lain di luar QLDBDriver

Untuk mencoba kembali transaksi di luar driver ketika kustom, pengecualian yang ditetapkan aplikasi dilemparkan selama runtime, Anda harus membungkus transaksi. Misalnya di Jawa, kode berikut menunjukkan bagaimana menggunakanReslience4Jperpustakaan untuk mencoba kembali transaksi di 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**
>
> Mencoba kembali transaksi di luar driver QLDB memiliki efek multiplier. Misalnya, jika`QldbDriver`dikonfigurasi untuk mencoba lagi tiga kali, dan logika coba ulang kustom juga mencoba tiga kali, transaksi yang sama dapat dicoba ulang hingga sembilan kali.

Melakukan idempoten transaksi

Sebagai praktik terbaik, buat transaksi tulis Anda idempoten untuk menghindari efek samping yang tak terduga dalam kasus percobaan ulang. Transaksi adalah*idempoten*jika dapat berjalan beberapa kali dan menghasilkan hasil yang identik setiap kali.

Untuk mempelajari selengkapnya, lihat Model Konkurensi Amazon QLDB.

## Mengoptimalkan performa

Untuk mengoptimalkan kinerja saat Anda menjalankan transaksi menggunakan driver, ambil pertimbangan berikut:

- Parameter`execute`operasi selalu membuat minimal tiga`SendCommand`API panggilan ke QLDB, termasuk perintah berikut:
  1. `StartTransaction`
  2. `ExecuteStatement`

Perintah ini dipanggil untuk setiap pernyataan partiQL yang Anda jalankan di`execute`blok.

3. `CommitTransaction`

Pertimbangkan jumlah total panggilan API yang dibuat saat Anda menghitung keseluruhan beban kerja aplikasi Anda.

- Secara umum, kami sarankan memulai dengan penulis single-threaded dan mengoptimalkan transaksi dengan batching beberapa pernyataan dalam satu transaksi. Maksimalkan kuota pada ukuran transaksi, ukuran dokumen, dan jumlah dokumen per transaksi, sebagaimana didefinisikan dalamKuota dan batasan di Amazon QLDB.
- Jika batching tidak cukup untuk beban transaksi besar, Anda dapat mencoba multi-threading dengan menambahkan penulis tambahan. Namun, Anda harus mempertimbangkan dengan cermat persyaratan aplikasi Anda untuk urutan dokumen dan transaksi dan kompleksitas tambahan yang diperkenalkan ini.

## Menjalankan beberapa pernyataan per transaksi

Seperti yang dijelaskan dalambagian sebelumnya, Anda dapat menjalankan beberapa pernyataan per transaksi untuk mengoptimalkan kinerja aplikasi Anda. Dalam contoh kode berikut, Anda query tabel dan kemudian memperbarui dokumen dalam tabel dalam transaksi. Anda melakukan ini dengan melewatkan ekspresi lambda ke`execute`operasi.

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))
```

Pengemudi`execute`operasi implisit memulai sesi dan transaksi dalam sesi itu. Setiap pernyataan yang Anda jalankan dalam ekspresi lambda dibungkus dalam transaksi. Setelah semua pernyataan berjalan, driver otomatis melakukan transaksi. Jika ada pernyataan yang gagal setelah batas coba ulang otomatis habis, transaksi dibatalkan.

Memperbanyak pengecualian dalam transaksi

Saat menjalankan beberapa pernyataan per transaksi, kami umumnya tidak menyarankan Anda catch dan menelan pengecualian dalam transaksi.

Misalnya di Jawa, program berikut menangkap setiap contoh`RuntimeException`, log kesalahan, dan terus. Contoh kode ini dianggap praktik buruk karena transaksi berhasil bahkan ketika`UPDATE`gagal. Jadi, klien mungkin berasumsi bahwa pembaruan berhasil ketika tidak.

> **⚠ Warning**
>
> Jangan gunakan contoh kode ini. Ini disediakan untuk menunjukkan contoh anti-pola yang dianggap praktik buruk.

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

Menyebarkan (gelembung up) pengecualian sebagai gantinya. Jika ada bagian dari transaksi gagal, biarkan`execute`operasi batalkan transaksi sehingga klien dapat menangani pengecualian sesuai.

# Memahami kebijakan coba ulang dengan pengemudi di Amazon QLDB

Driver Amazon QLDB menggunakan kebijakan coba ulang untuk menangani pengecualian sementara dengan mencoba kembali transaksi yang gagal secara transparan. Pengecualian ini, seperti`CapacityExceededException`dan`RateExceededException`, biasanya memperbaiki diri setelah jangka waktu tertentu. Jika transaksi yang gagal dengan pengecualian dicoba kembali setelah penundaan yang sesuai, kemungkinan akan berhasil. Hal ini membantu untuk meningkatkan stabilitas aplikasi yang menggunakan QLDB.

Topik



## Jenis kesalahan retryable

Pengemudi secara otomatis mencoba kembali transaksi jika dan hanya jika salah satu pengecualian berikut terjadi selama operasi dalam transaksi tersebut:

- KapasitasExceedException— Kembali ketika permintaan melebihi kapasitas pemrosesan buku besar.
- InvalidSessionException— Kembali ketika sesi tidak lagi valid atau jika sesi tidak ada.
- PengecualianBatasTerlampaui— Kembali jika batas sumber daya seperti jumlah sesi aktif terlampaui.
- OCCConflictException— Kembali ketika transaksi tidak dapat ditulis ke jurnal karena kegagalan dalam tahap verifikasikontrol konkurensi yang optimis(OKS).
- RateExceedException— Tingkat permintaan melebihi throughput yang diizinkan.

## Kebijakan percobaan ulang bawaan

Kebijakan coba ulang terdiri dari kondisi coba lagi dan strategi backoff. Kondisi coba lagi mendefinisikan kapan transaksi harus dicoba lagi, sementara strategi backoff menentukan berapa lama menunggu sebelum mencoba kembali transaksi.

Saat membuat instance driver, kebijakan coba ulang default menentukan untuk mencoba ulang hingga empat kali, dan menggunakan strategi backoff eksponensial. Strategi backoff eksponensial menggunakan penundaan minimal 10 milidetik dan penundaan maksimum 5000 milidetik, dengan jitter yang sama. Jika transaksi tidak dapat dilakukan dengan sukses dalam kebijakan coba ulang, sebaiknya coba transaksi di lain waktu.

Konsep backoff eksponensial adalah menggunakan waktu tunggu yang semakin lama antara percobaan ulang untuk respons kesalahan berturut-turut. Untuk informasi selengkapnya, lihatAWSposting blogBackoff Eksponensial dan Jitter.

## Kesalahan umum dari driver Amazon QLDB

Bagian ini menjelaskan kesalahan waktu proses yang dapat dilemparkan oleh driver Amazon QLDB saat berinteraksi dengan API Sesi QLDB.

Berikut ini adalah daftar pengecualian umum yang dikembalikan oleh driver. Setiap pengecualian mencakup pesan kesalahan tertentu, diikuti dengan deskripsi singkat dan saran untuk solusi yang mungkin.

CapacityExceededException

> Pesan: Kapasitas terlampaui
>
> Amazon QLDB menolak permintaan karena melebihi kapasitas pemrosesan buku besar. QLDB memberlakukan batas penskalaan internal per buku besar untuk menjaga kesehatan dan kinerja layanan. Batas ini bervariasi tergantung pada ukuran beban kerja setiap permintaan individu. Misalnya, permintaan dapat memiliki beban kerja yang meningkat jika melakukan transaksi data yang tidak efisien, seperti pemindaian tabel yang dihasilkan dari kueri non-indeks yang memenuhi syarat.
>
> Kami menyarankan Anda menunggu sebelum mencoba ulang permintaan. Jika aplikasi Anda secara konsisten menemukan pengecualian ini, optimalkan pernyataan Anda dan kurangi tingkat dan volume permintaan yang Anda kirim ke buku besar. Contoh optimasi pernyataan termasuk menjalankan lebih sedikit pernyataan per transaksi dan menyetel indeks tabel Anda. Untuk mempelajari cara mengoptimalkan pernyataan dan menghindari pemindaian tabel, lihatMengoptimalkan kinerja kueri.
>
> Kami juga merekomendasikan menggunakan versi terbaru dari driver QLDB. Driver memiliki kebijakan coba ulang default yang menggunakan Exponential Backoff dan Jitter untuk

secara otomatis mencoba lagi pengecualian seperti ini. Konsep backoff eksponensial adalah menggunakan waktu tunggu yang semakin lama antara percobaan ulang untuk respons kesalahan yang berurutan.

InvalidSessionException

Pesan: *`Transaksi TransactionID`* telah kedaluwarsa

Transaksi melebihi masa maksimumnya. Transaksi dapat berjalan hingga 30 detik sebelum dilakukan. Setelah batas batas waktu ini, setiap pekerjaan yang dilakukan pada transaksi ditolak, dan QLDB membuang sesi. Batas ini melindungi klien dari sesi bocor dengan memulai transaksi dan tidak melakukan atau membatalkannya.

Jika ini adalah pengecualian umum dalam aplikasi Anda, kemungkinan transaksi hanya membutuhkan waktu terlalu lama untuk dijalankan. Jika runtime transaksi memakan waktu lebih dari 30 detik, optimalkan laporan Anda untuk mempercepat transaksi. Contoh optimasi pernyataan termasuk menjalankan lebih sedikit pernyataan per transaksi dan menyetel indeks tabel Anda. Untuk informasi selengkapnya, lihat Mengoptimalkan kinerja kueri.

InvalidSessionException

Pesan: Session *`sessionId`* telah kedaluwarsa

QLDB dibuang sesi karena melebihi total seumur hidup maksimum. QLDB membuang sesi setelah 13-17 menit, terlepas dari transaksi aktif. Sesi dapat hilang atau terganggu karena sejumlah alasan, seperti kegagalan perangkat keras, kegagalan jaringan, atau restart aplikasi. Jadi, QLDB memberlakukan seumur hidup maksimum pada sesi untuk memastikan bahwa perangkat lunak klien tangguh terhadap kegagalan sesi.

Jika Anda menemukan pengecualian ini, kami sarankan Anda memperoleh sesi baru dan mencoba kembali transaksi. Kami juga merekomendasikan penggunaan driver QLDB versi terbaru, yang mengelola kumpulan sesi dan kesehatannya atas nama aplikasi.

InvalidSessionException

Pesan: Tidak ada sesi seperti itu

Klien mencoba untuk bertransaksi dengan QLDB menggunakan sesi yang tidak ada. Dengan asumsi bahwa klien menggunakan sesi yang sebelumnya ada, sesi mungkin tidak lagi ada karena salah satu dari berikut ini:

- Jika sesi terlibat dalam kegagalan server internal (yaitu, kesalahan dengan kode respons HTTP 500), QLDB mungkin memilih untuk membuang sesi sepenuhnya, daripada memungkinkan

pelanggan untuk bertransaksi dengan sesi keadaan tidak pasti. Kemudian, setiap percobaan ulang pada sesi itu gagal dengan kesalahan ini.

- Sesi kedaluwarsa akhirnya dilupakan oleh QLDB. Kemudian, setiap upaya untuk terus menggunakan sesi mengakibatkan kesalahan ini, bukan awal`InvalidSessionException`.

Jika Anda menemukan pengecualian ini, kami sarankan Anda memperoleh sesi baru dan mencoba kembali transaksi. Kami juga merekomendasikan penggunaan driver QLDB versi terbaru, yang mengelola kumpulan sesi dan kesehatannya atas nama aplikasi.

RateExceededException

Pesan: Tarif terlampaui

QLDB mencekik klien berdasarkan identitas penelepon. QLDB memberlakukan throttling pada basis per-wilayah, per akun menggunakan algoritma token bucket throttling. QLDB melakukan ini untuk membantu performa layanan, dan untuk memastikan penggunaan yang adil bagi semua pelanggan QLDB. Misalnya, mencoba memperoleh sejumlah besar sesi bersamaan menggunakan`StartSessionRequest` operasi dapat menyebabkan pelambatan.

Untuk menjaga kesehatan aplikasi Anda dan mengurangi pelambatan lebih lanjut, Anda dapat mencoba lagi pengecualian ini menggunakan Exponential Backoff dan Jitter. Konsep backoff eksponensial adalah menggunakan waktu tunggu yang semakin lama antara percobaan ulang untuk respons kesalahan yang berurutan. Sebaiknya gunakan versi terbaru dari driver QLDB. Driver memiliki kebijakan coba ulang default yang menggunakan backoff eksponensial dan jitter untuk secara otomatis mencoba lagi pengecualian seperti ini.

Versi terbaru dari driver QLDB juga dapat membantu jika aplikasi Anda secara konsisten mendapatkan throttled oleh QLDB untuk`StartSessionRequest` panggilan. Pengemudi mempertahankan kumpulan sesi yang digunakan kembali di seluruh transaksi, yang dapat membantu mengurangi jumlah`StartSessionRequest` panggilan yang dibuat aplikasi Anda. Untuk meminta peningkatan batas pembatasan API, hubungi AWS SupportPusat.

LimitExceededException

Pesan: Melebihi batas sesi

Buku besar melebihi kuota (juga dikenal sebagai batas) pada jumlah sesi aktif. Kuota ini didefinisikan dalamKuota dan batasan di Amazon QLDB. Jumlah sesi aktif buku besar pada akhirnya konsisten, dan buku besar secara konsisten berjalan di dekat kuota mungkin secara berkala melihat pengecualian ini.

Untuk menjaga kesehatan aplikasi Anda, kami sarankan untuk mencoba kembali pengecualian ini. Untuk menghindari pengecualian ini, pastikan bahwa Anda belum mengkonfigurasi lebih dari 1.500 sesi bersamaan untuk digunakan untuk buku besar tunggal di semua klien. Misalnya, Anda dapat menggunakan maxConcurrentTransactionsmetode driver Amazon QLDB untuk Java untuk mengonfigurasi jumlah maksimum sesi yang tersedia dalam instance driver.

QldbClientException

Pesan: Hasil streaming hanya berlaku saat transaksi induk terbuka

Transaksi ditutup, dan tidak dapat digunakan untuk mengambil hasil dari QLDB. Transaksi ditutup saat transaksi dilakukan atau dibatalkan.

Pengecualian ini terjadi ketika klien bekerja secara langsung dengan`Transaction` objek, dan itu mencoba untuk mengambil hasil dari QLDB setelah melakukan atau membatalkan transaksi. Untuk mengurangi masalah ini, klien harus membaca data sebelum menutup transaksi.

# Memulai dengan Amazon QLDB menggunakan contoh tutorial aplikasi

Dalam tutorial ini, Anda menggunakan driver Amazon QLDB denganAWS SDK untuk membuat buku besar QLDB dan mengisinya dengan data sampel. Driver memungkinkan aplikasi Anda berinteraksi dengan QLDB menggunakan API data transaksional. AWSSDK mendukung interaksi dengan API manajemen sumber daya QLDB.

Buku besar sampel yang Anda buat dalam skenario ini adalah departemen database kendaraan bermotor (DMV) yang melacak informasi historis lengkap tentang pendaftaran kendaraan. Topik berikut menjelaskan cara menambahkan registrasi kendaraan, memodifikasinya, dan melihat riwayat perubahan pada pendaftaran tersebut. Panduan ini juga menunjukkan kepada Anda cara memverifikasi dokumen pendaftaran secara kriptografis, dan diakhiri dengan membersihkan sumber daya dan menghapus buku besar sampel.

Aplikasi sampel tersedia untuk bahasa pemrograman berikut.

Topik

- tutorial Amazon QLDB Java
- Amazon QLDB Node.js tutorial
- Tutorial Amazon QLDB Python

# tutorial Amazon QLDB Java

Dalam implementasi aplikasi contoh tutorial ini, Anda menggunakan driver Amazon QLDB denganAWS SDK for Java untuk membuat buku besar QLDB dan mengisinya dengan data sampel.

Ketika menggunakan tutorial ini, Anda dapat melihat Refensi AWS SDK for JavaAPI untuk operasi API manajemen API. Untuk operasi data transaksional, Anda dapat merujuk ke QLDB Driver for Java API Reference.

> **Note**
>
> Jika berlaku, beberapa langkah tutorial memiliki perintah atau contoh kode yang berbeda untuk setiap versi utama driver QLDB yang didukung untuk Java.

Topik



## Menginstal aplikasi sampel Amazon QLDB Java

Bagian ini menjelaskan cara menginstal dan menjalankan aplikasi sampel Amazon QLDB yang disediakan untuk tutorial step-by-step Java. Kasus penggunaan untuk aplikasi sampel ini adalah departemen database kendaraan bermotor (DMV) yang melacak informasi historis lengkap tentang pendaftaran kendaraan.

Aplikasi sampel DMV untuk Java adalah open source di GitHub repositori amazon-qldb-dmv-sampleaws-samples/-java.

### Prasyarat

Sebelum memulai, pastikan Anda menyelesaikan driver QLDB untuk JavaPrasyarat. Hal ini termasuk skenario berikut:

1. Daftar ke AWS.
2. Buat pengguna dengan izin QLDB yang sesuai. Untuk menyelesaikan semua langkah dalam tutorial ini, Anda memerlukan akses administratif penuh ke sumber daya buku besar Anda melalui QLDB API.
3. Jika Anda menggunakan IDE selainAWS Cloud9, instal Java dan berikan akses terprogram untuk pengembangan.

### Instalasi

Langkah-langkah berikut menjelaskan cara mengunduh dan menyiapkan contoh aplikasi dengan lingkungan pengembangan lokal. Atau, Anda dapat mengotomatiskan pengaturan aplikasi sampel dengan menggunakanAWS Cloud9 sebagai IDE Anda, danAWS CloudFormation template untuk menyediakan sumber daya pengembangan Anda.

#### Lingkungan pengembangan lokal

Petunjuk ini menjelaskan cara mengunduh dan menginstal aplikasi sampel QLDB Java menggunakan sumber daya dan lingkungan pengembangan Anda sendiri.

Untuk mengunduh dan menjalankan aplikasi sampel

1. Masukkan perintah berikut untuk mengkloning aplikasi sampel sampel dari 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
   ```

   Paket ini mencakup konfigurasi Gradle dan kode lengkap dari filetutorial java.

2. Muat dan jalankan aplikasi yang disediakan.
   - Jika Anda menggunakan Eclipse:
     a. Mulai Eclipse, dan pada menu Eclipse, pilih File, Import, dan kemudian Existing Gradle Project.
     b. Di direktori root proyek, telusuri dan pilih direktori aplikasi yang berisi`build.gradle` file. Kemudian, pilih Selesai untuk menggunakan pengaturan Gradle default untuk impor.
     c. Anda dapat mencoba menjalankan`ListLedgers` program sebagai contoh. Buka menu konteks (klik kanan) untuk`ListLedgers.java` file, lalu pilih Jalankan aplikasi Java.
   - Jika Anda menggunakan IntelliJ:
     a. Mulai IntelliJ, dan pada menu IntelliJ, pilih File dan kemudian Buka.
     b. Di direktori root proyek, telusuri dan pilih direktori aplikasi yang berisi`build.gradle` file. Kemudian, pilih OK. Simpan pengaturan default, dan pilih OK lagi.
     c. Anda dapat mencoba menjalankan`ListLedgers` program sebagai contoh. Buka menu konteks (klik kanan) untuk`ListLedgers.java` file, lalu pilih Jalankan menu konteks (klik kananListLedgers).
3. LanjutkanLangkah 1: Buat buku besar baru untuk memulai tutorial dan membuat buku besar.

AWS Cloud9

Petunjuk ini menjelaskan cara mengotomatiskan penyiapan aplikasi sampel pendaftaran kendaraan Amazon QLDB untuk Java, yang digunakan AWS Cloud9sebagai IDE Anda. Dalam panduan ini, Anda menggunakan AWS CloudFormationtemplate untuk menyediakan sumber daya pengembangan Anda.

Untuk informasi selengkapnya tentang AWS Cloud9, lihat AWS Cloud9Panduan Pengguna . Untuk mempelajari lebih lanjutAWS CloudFormation, lihat PanduanAWS CloudFormation Pengguna.

Topik

- Bagian 1: Untuk menyediakan sumber daya Anda
- Bagian 2: Siapkan IDE Anda
- Bagian 3: Jalankan aplikasi sampel sampel QLDB DMV

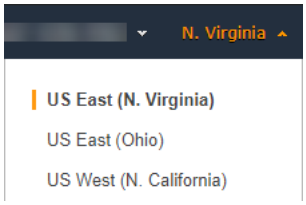
Bagian 1: Untuk menyediakan sumber daya Anda

Pada langkah pertama ini, Anda gunakanAWS CloudFormation untuk menyediakan sumber daya yang diperlukan untuk menyiapkan lingkungan pengembangan Anda dengan aplikasi sampel Amazon QLDB.

Untuk membukaAWS CloudFormation konsol dan memuat template aplikasi sampel QLDB

1. Masuk ke AWS Management Console dan buka konsol AWS CloudFormation di https://console.aws.amazon.com/cloudformation.

   Beralih ke Wilayah yang mendukung QLDB. Untuk daftar lengkapnya, lihat titik akhir dan kuota Amazon QLDB di bagian Referensi Umum AWS. Berikut screenshot dariAWS Management Console acara US (N. Virginia) sebagai yang dipilihWilayah AWS..

2. DiAWS CloudFormation konsol, pilih Buat tumpukan, lalu pilih Dengan sumber daya baru (standar).
3. Pada halaman Create stack di bawah Specify template, pilih URL Amazon S3.
4. Masukkan URL berikut, dan pilih Berikutnya.

   ```
   https://amazon-qldb-assets.s3.amazonaws.com/templates/QLDB-DMV-SampleApp.yml
   ```

5. Masukkan nama Stack (seperti**`qldb-sample-app`**), dan pilih Next.
6. Anda dapat menambahkan tag apa pun yang sesuai dan menyimpan opsi default. Kemudian pilih Selanjutnya.
7. Tinjau pengaturan tumpukan Anda, dan pilih Buat tumpukan. AWS CloudFormationSkrip mungkin memerlukan waktu beberapa menit untuk diselesaikan.

   Skrip ini menyediakanAWS Cloud9 lingkungan Anda dengan instans Amazon Elastic Compute Cloud (Amazon EC2) yang Anda gunakan untuk menjalankan aplikasi sampel QLDB dalam tutorial ini. Ini juga mengkloning repositori aws-samples/amazon-qldb-dmv-sample -java dari GitHub ke lingkunganAWS Cloud9 pengembangan Anda.

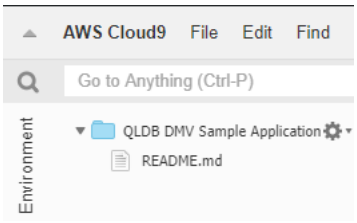
Bagian 2: Siapkan IDE Anda

Pada langkah ini, Anda menyelesaikan pengaturan lingkungan pengembangan cloud Anda. Anda mengunduh dan menjalankan skrip shell yang disediakan untuk mengaturAWS Cloud9 IDE Anda dengan dependensi aplikasi sampel.

Untuk mempersiapkanAWS Cloud9 lingkungan Anda

1. BukaAWS Cloud9 konsol di https://console.aws.amazon.com/cloud9/.
2. Di lingkungan Anda, cari kartu untuk lingkungan bernama QLDB DMV Sample Application, dan pilih Open IDE. Lingkungan Anda mungkin memerlukan waktu satu menit untuk dimuat saat instans EC2 yang mendasarinya diluncurkan.

   AWS Cloud9Lingkungan Anda telah dikonfigurasi sebelumnya dengan dependensi sistem yang Anda butuhkan untuk menjalankan tutorial. Di panel navigasi Lingkungan konsol Anda, konfirmasikan bahwa Anda melihat folder bernama`QLDB DMV Sample Application`. Screenshot berikut dariAWS Cloud9 konsol menunjukkan panel folder lingkungan Aplikasi Sampel DMV QLDB.

   Jika Anda tidak melihat panel navigasi, alihkan tab Lingkungan di sisi kiri konsol. Jika Anda tidak melihat folder apa pun di panel, aktifkan Tampilkan Akar Lingkungan menggunakan ikon pengaturan ( ).
3. Pada panel bawah konsol Anda, Anda akan melihat jendela`bash` terminal yang terbuka. Jika Anda tidak melihat ini, pilih Terminal Baru dari menu Window di bagian atas konsol Anda.
4. Selanjutnya, unduh dan jalankan skrip penyiapan untuk menginstal OpenJDK 8 dan—jika dapat diterapkan - periksa cabang yang sesuai dari repositori Git. DiAWS Cloud9 terminal yang Anda buat di langkah sebelumnya, jalankan dua perintah berikut secara berurutan:

   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
```

Setelah selesai, Anda akan melihat pesan berikut yang dicetak di terminal:

```
** DMV Sample App setup completed , enjoy!! **
```

5. Luangkan waktu sejenak untuk menelusuri contoh kode aplikasiAWS Cloud9, terutama di jalur direktori berikut:`src/main/java/software/amazon/qldb/tutorial`.

Bagian 3: Jalankan aplikasi sampel sampel QLDB DMV

Pada langkah ini, Anda mempelajari cara menjalankan tugas aplikasi sampel Amazon QLDB DMV menggunakanAWS Cloud9. Untuk menjalankan kode contoh, kembali keAWS Cloud9 terminal Anda atau buat jendela terminal baru seperti yang Anda lakukan di Bagian 2: Mengatur IDE Anda.

Untuk menjalankan aplikasi sampel sampel sampel sampel sampel sampel.

1. Jalankan perintah berikut di terminal Anda untuk beralih ke direktori root proyek:

```
cd ~/environment/amazon-qldb-dmv-sample-java
```

Pastikan Anda menjalankan contoh di jalur direktori berikut.

```
/home/ec2-user/environment/amazon-qldb-dmv-sample-java/
```

2. Perintah berikut menunjukkan sintaks Gradle untuk menjalankan setiap tugas.

```
./gradlew run -Dtutorial=Task
```

Sebagai contoh, jalankan perintah berikut untuk mencantumkan semua buku besar di Region AndaAkun AWS dan saat ini.

```
./gradlew run -Dtutorial=ListLedgers
```

3. LanjutkanLangkah 1: Buat buku besar baru untuk memulai tutorial dan membuat buku besar.
4. (Opsional) Setelah Anda menyelesaikan tutorial, bersihkanAWS CloudFormation sumber daya Anda jika tidak lagi membutuhkannya.

   a. BukaAWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation, dan hapus tumpukan yang Anda buat di Bagian 1: Menyediakan Sumber Daya Anda.

   b. Juga hapusAWS Cloud9 tumpukan yang dibuatAWS CloudFormation template untuk Anda.

## Langkah 1: Buat buku besar baru

Pada langkah ini, Anda membuat buku besar Amazon QLDB baru bernama`vehicle-registration`.

Untuk membuat buku besar baru

1. Tinjau file berikut (`Constants.java`), yang berisi nilai-nilai konstan yang digunakan oleh semua program lain dalam tutorial ini.

   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**
>
> Untuk paket Amazon Ion, Anda harus menggunakan namespace`com.amazon.ion` dalam aplikasi Anda. AWS SDK for JavaTergantung pada paket Ion lain di bawah namespace`software.amazon.ion`, tetapi ini adalah paket lama yang tidak kompatibel dengan 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**
>
> `Constants`Kelas ini mencakup instance dari`IonValueMapper` kelas Jackson open-source. Anda dapat menggunakan mapper ini untuk memproses data Amazon Ion Anda saat melakukan transaksi baca dan tulis.

`CreateLedger.java`File ini juga memiliki ketergantungan pada program berikut (`DescribeLedger.java`), yang menggambarkan status buku besar Anda saat ini.

```
/*
 * 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. Mengkompilasi dan menjalankan`CreateLedger.java` program untuk membuat buku besar bernama`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**
>
> - Dalam`createLedger` panggilan, Anda harus menentukan nama buku besar dan mode izin. Kami merekomendasikan untuk menggunakan mode`STANDARD` izin untuk memaksimalkan keamanan data buku besar Anda.
> - Ketika membuat buku besar, perlindungan penghapusan diaktifkan secara default. Ini adalah fitur dalam QLDB yang mencegah buku besar dihapus oleh pengguna mana pun Anda memiliki opsi untuk menonaktifkan perlindungan penghapusan pada

pembuatan buku besar menggunakan QLDB API atauAWS Command Line Interface (AWS CLI).

- Opsional, Anda juga dapat menentukan tag untuk melampirkan buku besar Anda.

Untuk memverifikasi koneksi Anda ke buku besar baru, lanjutkan keLangkah 2: Uji konektivitas ke buku besar.

## Langkah 2: Uji konektivitas ke buku besar

Pada langkah ini, Anda memverifikasi bahwa Anda dapat terhubung ke`vehicle-registration` buku besar di Amazon QLDB menggunakan endpoint API data transaksional.

Untuk menguji konektivitas buku besar

1. Tinjau program berikut (`ConnectToLedger.java`), yang menciptakan koneksi sesi data ke`vehicle-registration` buku besar.

   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**
>
> - Untuk menjalankan operasi data pada buku besar Anda, Anda harus membuat instance`QldbDriver` kelas untuk terhubung ke buku besar tertentu. Ini adalah objek klien yang berbeda dari`AmazonQLDB` klien yang Anda gunakan dalam langkah sebelumnya untuk membuat buku besar. Klien sebelumnya hanya digunakan untuk menjalankan operasi API manajemen yang tercantum dalamReferensi API Amazon QLDB.
> - Pertama, buat`QldbDriver` objek. Anda harus menentukan nama buku besar saat Anda membuat driver ini.
>
>   Kemudian, Anda dapat menggunakan`execute` metode driver ini untuk menjalankan pernyataan PartiQL.

- Opsional, Anda dapat menentukan jumlah upaya percobaan ulang maksimum untuk pengecualian transaksi. `execute`Metode ini secara otomatis mencoba ulang konflik kontrol konkurensi optimistik (OCC) dan pengecualian sementara umum lainnya hingga batas yang dapat dikonfigurasi ini. Nilai default-nya adalah `4`.

  Jika transaksi masih gagal setelah batas tercapai, maka pengemudi melempar pengecualian. Untuk mempelajari selengkapnya, lihat Memahami kebijakan coba ulang dengan pengemudi di 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**

- Untuk menjalankan operasi data pada buku besar Anda, Anda harus membuat instance`QldbDriver` kelas`PooledQldbDriver` atau untuk terhubung ke buku

besar tertentu. Ini adalah objek klien yang berbeda dari`AmazonQLDB` klien yang Anda gunakan pada langkah sebelumnya untuk membuat buku besar. Klien sebelumnya hanya digunakan untuk menjalankan operasi API manajemen yang tercantum dalamReferensi API Amazon QLDB.

Sebaiknya gunakan`PooledQldbDriver` kecuali Anda perlu mengimplementasikan pool sesi khusus dengan`QldbDriver`. Ukuran kolam default untuk`PooledQldbDriver` adalah jumlah maksimum koneksi HTTP terbuka yang diizinkan klien sesi.

- Pertama, buat`PooledQldbDriver` objek. Anda harus menentukan nama buku besar saat Anda membuat driver ini.

  Kemudian, Anda dapat menggunakan`execute` metode driver ini untuk menjalankan pernyataan PartiQL. Atau, Anda dapat secara manual membuat sesi dari objek driver yang dikumpulkan ini dan menggunakan`execute` metode sesi. Sesi merupakan koneksi tunggal dengan buku besar.

- Opsional, Anda dapat menentukan jumlah upaya percobaan ulang maksimum untuk pengecualian transaksi. `execute`Metode ini secara otomatis mencoba ulang konflik kontrol konkurensi optimistik (OCC) dan pengecualian sementara umum lainnya hingga batas yang dapat dikonfigurasi ini. Nilai default-nya adalah 4.

  Jika transaksi masih gagal setelah batas tercapai, maka pengemudi melempar pengecualian. Untuk mempelajari selengkapnya, lihat Memahami kebijakan coba ulang dengan pengemudi di Amazon QLDB.

2. Kompilasi dan jalankan`ConnectToLedger.java` program untuk menguji konektivitas sesi data Anda ke`vehicle-registration` buku besar.

MengesampingkanWilayah AWS

Contoh aplikasi terhubung ke QLDB di default AndaWilayah AWS, yang dapat Anda atur seperti yang dijelaskan dalam langkah prasyaratMengaturAWS kredensi dan Wilayah default. Anda juga dapat mengubah Wilayah dengan memodifikasi properti pembangun klien sesi QLDB.

2.x

Contoh kode berikut menunjukkan`QldbSessionClientBuilder` objek baru.

```
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);
```

Anda dapat menggunakan`region` metode untuk menjalankan kode Anda terhadap QLDB di Wilayah ketersediaan mana pun Untuk daftar lengkapnya, lihat titik akhir dan kuota Amazon QLDB di bagian Referensi Umum AWS.

1.x

Contoh kode berikut menunjukkan`AmazonQLDBSessionClientBuilder` objek baru.

```
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);
```

Anda dapat menggunakan`withRegion` metode untuk menjalankan kode Anda terhadap QLDB di Wilayah ketersediaan mana pun Untuk daftar lengkapnya, lihat titik akhir dan kuota Amazon QLDB di bagian Referensi Umum AWS.

Untuk membuat tabel di`vehicle-registration` buku besar, lanjutkan keLangkah 3: Buat tabel, indeks, dan data sampel.

## Langkah 3: Buat tabel, indeks, dan data sampel

Saat buku besar Amazon QLDB Anda aktif dan menerima koneksi, Anda dapat mulai membuat tabel untuk data tentang kendaraan, pemiliknya, dan informasi pendaftarannya. Setelah membuat tabel dan indeks, Anda dapat memuatnya dengan data.

Pada langkah ini, Anda membuat empat tabel di`vehicle-registration` buku besar:

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

- `DriversLicense`

Anda juga membuat indeks berikut.

| Nama tabel | Bidang |
| --- | --- |
| `VehicleRegistration` | `VIN` |
| `VehicleRegistration` | `LicensePlateNumber` |
| `Vehicle` | `VIN` |
| `Person` | `GovId` |
| `DriversLicense` | `LicenseNumber` |
| `DriversLicense` | `PersonId` |

Saat memasukkan data sampel, Anda terlebih dahulu memasukkan dokumen ke dalam`Person` tabel. Kemudian, Anda menggunakan sistem-ditugaskan`id` dari setiap`Person` dokumen untuk mengisi bidang yang sesuai dalam`DriversLicense` dokumen`VehicleRegistration` dan yang sesuai.

> **Tip**
>
> Sebagai praktik terbaik, gunakan sistem dokumen yang ditugaskan`id` sebagai kunci asing. Meskipun Anda dapat menentukan bidang yang dimaksudkan untuk menjadi pengenal unik (misalnya, VIN kendaraan), pengenal unik sebenarnya dari dokumen adalah`id`. Bidang ini termasuk dalam metadata dokumen, yang dapat Anda kueri dalam tampilan berkomitmen (tampilan tabel yang ditentukan sistem).
> Untuk informasi lebih lanjut tentang tampilan di QLDB, lihatKonsep inti. Untuk mempelajari metadata, lihatMelakukan Kueri Metadata Dokumen.

Untuk menyiapkan data sampel nya

1. Tinjau`.java` file-file berikut. Kelas model ini mewakili dokumen yang Anda simpan di`vehicle-registration` tabel. Mereka serializable ke dan dari format Amazon Ion.

> **Note**
>
> Dokumen QLDB B B B B B B B B disimpan dalam format Ion, yang merupakan superset dari JSON. Jadi, Anda dapat menggunakan pustaka FasterXMLJackson untuk memodelkan data di 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. Tinjau file berikut (`SampleData.java`), yang mewakili data sampel yang Anda masukkan ke dalam`vehicle-registration` tabel.

   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**
>
> Untuk paket Amazon Ion, Anda harus menggunakan namespace`com.amazon.ion` dalam aplikasi Anda. AWS SDK for JavaTergantung pada paket Ion lain di bawah namespace`software.amazon.ion`, tetapi ini adalah paket lama yang tidak kompatibel dengan 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**
>
> - Kelas ini menggunakan pustaka Ion untuk menyediakan metode pembantu yang mengonversi data Anda ke dan dari format Ion.
> - `getDocumentId`Metode ini menjalankan query di atas meja dengan awalan`_ql_committed_`. Ini adalah awalan reserved menandakan bahwa Anda ingin query pandangan berkomitmen dari tabel. Dalam tampilan ini, data Anda bersarang di`data` bidang, dan metadata bersarang di`metadata` bidang.

3. Mengkompilasi dan menjalankan program berikut (`CreateTable.java`) untuk membuat tabel yang disebutkan sebelumnya.

   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**
>
> Program ini menunjukkan bagaimana untuk lulus`TransactionExecutor` lambda ke`execute` metode. Dalam contoh ini, Anda menjalankan beberapa pernyataan`CREATE TABLE` PartiQL dalam satu transaksi dengan menggunakan ekspresi lambda. `execute`Metode ini secara implisit memulai transaksi, menjalankan semua pernyataan di lambda, dan kemudian melakukan transaksi secara otomatis.

4. Mengkompilasi dan menjalankan program berikut (`CreateIndex.java`) untuk membuat indeks pada tabel, seperti yang dijelaskan sebelumnya.

   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. Mengkompilasi dan menjalankan program berikut (`InsertDocument.java`) untuk memasukkan data sampel ke dalam tabel Anda.

   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**
>
> - Program ini menunjukkan bagaimana untuk memanggil`execute` metode dengan nilai-nilai parameter. Anda dapat melewati parameter data jenis`IonValue` selain pernyataan PartiQL yang ingin Anda jalankan. Gunakan tanda tanya (?) sebagai placeholder variabel dalam string pernyataan Anda.

- Jika`INSERT` pernyataan berhasil, ia mengembalikan`id` dari setiap dokumen dimasukkan.

Selanjutnya, Anda dapat menggunakan`SELECT` pernyataan untuk membaca data dari tabel di`vehicle-registration` buku besar. Lanjut ke Langkah 4: Cari tabel dalam buku besar.

## Langkah 4: Cari tabel dalam buku besar

Setelah membuat tabel di buku besar Amazon QLDB dan memuatnya dengan data, Anda dapat menjalankan kueri untuk meninjau data registrasi kendaraan yang baru saja Anda masukkan. QLDB menggunakan PartiQL sebagai bahasa kueri dan Amazon Ion sebagai model data berorientasi dokumen.

PartiQL adalah open-source, bahasa query SQL-kompatibel yang telah diperluas untuk bekerja dengan Ion. Dengan PartiQL, Anda dapat memasukkan, query, dan mengelola data Anda dengan operator SQL akrab. Amazon Ion adalah superset dari JSON. Ion adalah open-source, format data berbasis dokumen yang memberi Anda fleksibilitas menyimpan dan memproses data terstruktur, semistruktural, dan bersarang.

Pada langkah ini, Anda menggunakan`SELECT` pernyataan untuk membaca data dari tabel di`vehicle-registration` buku besar.

> **Warning**
>
> Ketika Anda menjalankan query di QLDB tanpa pencarian diindeks, memanggil scan tabel penuh. PartiQL mendukung query tersebut karena SQL kompatibel. Namun, jangan jalankan pemindaian tabel untuk kasus penggunaan produksi di QLDB. Pemindaian tabel dapat menyebabkan masalah kinerja pada tabel besar, termasuk konflik konkurensi dan batas waktu transaksi.
> Untuk menghindari pemindaian tabel, Anda harus menjalankan pernyataan dengan klausa`WHERE` predikat menggunakan operator kesetaraan pada bidang yang diindeks atau ID dokumen; misalnya,`WHERE indexedField = 123` atau`WHERE indexedField IN (456, 789)`. Untuk informasi selengkapnya, lihat Mengoptimalkan kinerja kueri.

Untuk query tabel

- Kompilasi dan jalankan program berikut (`FindVehicles.java`) untuk meminta semua kendaraan yang terdaftar di bawah seseorang di buku besar Anda.

  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**
>
> Pertama, program ini query`Person` tabel untuk dokumen dengan`GovId LEWISR261LL` untuk mendapatkan bidang`id` metadata.
> Kemudian, menggunakan dokumen ini`id` sebagai kunci asing untuk query`VehicleRegistration` tabel oleh`PrimaryOwner.PersonId`. Ini juga bergabung`VehicleRegistration` dengan`Vehicle` meja di`VIN` lapangan.

Untuk mempelajari tentang memodifikasi dokumen dalam tabel di`vehicle-registration` buku besar, lihatLangkah 5: Memodifikasi dokumen dalam buku besar.

## Langkah 5: Memodifikasi dokumen dalam buku besar

Sekarang setelah Anda memiliki data untuk digunakan, Anda dapat mulai membuat perubahan pada dokumen dalam`vehicle-registration` buku besar di Amazon QLDB. Pada langkah ini, contoh kode berikut menunjukkan bagaimana menjalankan laporan manipulasi data (DML). Pernyataan ini memperbarui pemilik utama satu kendaraan dan menambahkan pemilik sekunder ke kendaraan lain.

Untuk mengubah dokumen

1. Kompilasi dan jalankan program berikut (`TransferVehicleOwnership.java`) untuk memperbarui pemilik utama kendaraan dengan VIN`1N4AL11D75C109151` di buku besar Anda.

   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. Kompilasi dan jalankan program berikut (`AddSecondaryOwner.java`) untuk menambahkan pemilik sekunder ke kendaraan dengan VIN`KM8SRDHF6EU074761` di buku besar Anda.

   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.");
    }
}
```

Untuk meninjau perubahan ini dalam`vehicle-registration` buku besar, lihatLangkah 6: Lihat riwayat revisi dokumen.

## Langkah 6: Lihat riwayat revisi dokumen

Setelah memodifikasi data pendaftaran untuk kendaraan pada langkah sebelumnya, Anda dapat menanyakan riwayat semua pemilik terdaftar dan bidang terbaru lainnya. Pada langkah ini, Anda menanyakan riwayat revisi dokumen dalam`VehicleRegistration` tabel di`vehicle-registration` buku besar Anda.

Untuk melihat riwayat revisi

1. Tinjau program berikut (`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**
>
> - Anda dapat melihat riwayat revisi dokumen dengan menanyakan built-inFungsi Riwayat dalam sintaks berikut.
>
>   ```
>   SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
>   [ WHERE h.metadata.id = 'id' ]
>   ```
>
> - Start-time dan end-time keduanya opsional. Mereka adalah nilai literal Amazon Ion yang dapat dilambangkan dengan backticks (`` `...` ``). Untuk mempelajari selengkapnya, lihat Menanyakan Ion dengan PartiQL di Amazon QLDB.
> - Sebagai praktik terbaik, kualifikasi kueri riwayat dengan rentang tanggal (start-time dan end-time) dan ID dokumen (`metadata.id`). QLDB memprosesSELECT kueri dalam transaksi, yang tunduk pada batas batas waktu transaksi.
>
>   Riwayat QLDB diindeks oleh ID dokumen, dan Anda tidak dapat membuat indeks riwayat tambahan saat ini. Kueri riwayat yang menyertakan waktu mulai dan waktu akhir mendapatkan manfaat kualifikasi rentang tanggal.

2. Kompilasi dan jalankan`QueryHistory.java` program untuk menanyakan riwayat revisi`VehicleRegistration` dokumen dengan VIN`1N4AL11D75C109151`.

Untuk memverifikasi revisi dokumen secara kriptografis di`vehicle-registration` buku besar, lanjutkan keLangkah 7: Verifikasi dokumen di buku besar.

## Langkah 7: Verifikasi dokumen di buku besar

Dengan Amazon QLDB, Anda dapat secara efisien memverifikasi integritas dokumen dalam jurnal buku besar Anda dengan menggunakan hashing kriptografi dengan SHA-256. Untuk mempelajari lebih lanjut tentang cara kerja verifikasi dan hashing kriptografi di QLDB, lihatVerifikasi data di Amazon QLDB.

Pada langkah ini, Anda memverifikasi revisi dokumen dalam`VehicleRegistration` tabel di`vehicle-registration` buku besar Anda. Pertama, Anda meminta digest, yang dikembalikan sebagai file output dan bertindak sebagai tanda tangan dari seluruh riwayat perubahan buku besar Anda. Kemudian, Anda meminta bukti untuk revisi relatif terhadap intisari itu. Dengan menggunakan bukti ini, integritas revisi Anda diverifikasi jika semua pemeriksaan validasi lulus.

Untuk memverifikasi revisi dokumen

1. Tinjau`.java` file berikut, yang mewakili objek QLDB yang diperlukan untuk verifikasi dan utilitas kelas dengan metode pembantu untuk Ion dan nilai string.

   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. Gunakan dua `.java` file (`GetDigest.javadanGetRevision.java`) untuk melakukan langkah-langkah berikut:

   - Minta intisari baru dari`vehicle-registration` buku besar.

- Minta bukti untuk setiap revisi dokumen dari`VehicleRegistration` tabel.
- Verifikasi revisi menggunakan intisari dan bukti yang dikembalikan dengan menghitung ulang intisari.

`GetDigest.java`Program ini berisi kode berikut.

```
/*
 * 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**
>
> Gunakan`getDigest` metode untuk meminta intisari yang mencakup ujung jurnal saat ini di buku besar Anda. Ujung jurnal mengacu pada blok berkomitmen terbaru pada saat QLDB menerima permintaan Anda.

`GetRevision.java`Program ini berisi kode berikut.

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**
>
> Setelah`getRevision` metode mengembalikan bukti untuk revisi dokumen yang ditentukan, program ini menggunakan API sisi klien untuk memverifikasi revisi itu. Untuk ikhtisar algoritma yang digunakan oleh API ini, lihatMenggunakan bukti untuk menghitung ulang intisari Anda.

3. Kompilasi dan jalankan`GetRevision.java` program untuk memverifikasi`VehicleRegistration` dokumen secara kriptografis dengan VIN`1N4AL11D75C109151`.

Untuk mengekspor dan memvalidasi data jurnal di`vehicle-registration` buku besar, lanjutkan keLangkah 8: Ekspor dan validasi data jurnal dalam buku besar.

## Langkah 8: Ekspor dan validasi data jurnal dalam buku besar

Di Amazon QLDB, Anda dapat mengakses konten jurnal di buku besar Anda untuk berbagai tujuan seperti retensi data, analitik, dan audit. Untuk informasi selengkapnya, lihat Mengekspor data jurnal dari Amazon QLDB.

Pada langkah ini, Anda mengekspor blok jurnal dari`vehicle-registration` buku besar ke bucket Amazon S3. Kemudian, Anda menggunakan data yang diekspor untuk memvalidasi rantai hash antara blok jurnal dan komponen hash individual dalam setiap blok.

Entitas utamaAWS Identity and Access Management (IAM) yang Anda gunakan harus memiliki izin IAM yang memadai untuk membuat bucket Amazon S3 di bucket AndaAkun AWS. Untuk informasi, lihat Kebijakan dan Izin di Amazon S3 di Panduan Pengguna Amazon S3. Anda juga harus memiliki izin untuk membuat peran IAM dengan kebijakan izin terlampir yang memungkinkan QLDB untuk menulis objek ke bucket Amazon S3 Anda. Untuk mempelajari selengkapnya, lihat Izin yang diperlukan untuk mengakses sumber daya IAM di Panduan Pengguna IAM.

Mengekspor dan memvalidasi data jurnal

1. Tinjau file berikut (`JournalBlock.java`), yang mewakili blok jurnal dan isi datanya. Ini termasuk metode bernama`verifyBlockHash()` yang menunjukkan bagaimana menghitung setiap komponen individu dari hash blok.

```
/*
 * 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. Kompilasi dan jalankan program berikut (`ValidateQldbHashChain.java`) untuk melakukan langkah-langkah berikut:

   1. Ekspor blok jurnal dari`vehicle-registration` buku besar ke bucket Amazon S3 bernama**`qldb-tutorial-journal-export-`*`111122223333`*** (ganti denganAkun AWS nomor Anda).
   2. Validasi komponen hash individu dalam setiap blok dengan meneleponverifyBlockHash().
   3. Validasi rantai hash antara blok jurnal.

```
/*
 * 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;
        }
    }

}
```

Jika Anda tidak perlu lagi menggunakan`vehicle-registration` buku besar, lanjutkan keLangkah 9 (opsional): Bersihkan Sumber daya daya daya daya daya daya daya.

## Langkah 9 (opsional): Bersihkan Sumber daya daya daya daya daya daya daya daya

Anda dapat terus menggunakan`vehicle-registration` buku besar. Namun, jika Anda tidak lagi membutuhkannya, Anda harus menghapusnya.

Untuk menghapus buku besar

1. Kompilasi dan jalankan program berikut (`DeleteLedger.java`) untuk menghapus`vehicle-registration` buku besar Anda dan semua isinya.

```
/*
 * 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**
>
> Jika perlindungan penghapusan diaktifkan untuk buku besar Anda, Anda harus menonaktifkannya terlebih dahulu sebelum dapat menghapus buku besar menggunakan QLDB API.

2. Jika Anda mengekspor data jurnal pada langkah sebelumnya dan tidak lagi memerlukannya, gunakan konsol Amazon S3 untuk menghapus bucket S3 Anda.

   Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.

## Amazon QLDB Node.js tutorial

Dalam implementasi aplikasi contoh tutorial ini, Anda menggunakan driver Amazon QLDB dengan AWS SDK JavaScript untuk di Node.js untuk membuat buku besar QLDB dan mengisinya dengan data sampel.

Saat Anda mengerjakan tutorial ini, Anda dapat merujuk ke Referensi AWS SDK for JavaScript API untuk operasi API manajemen. Untuk operasi data transaksional, Anda dapat merujuk ke Driver QLDB untuk Referensi API Node.js.

> **Note**
>
> Jika berlaku, beberapa langkah tutorial memiliki perintah atau contoh kode yang berbeda untuk setiap versi utama yang didukung dari driver QLDB untuk Node.js.

Topik



### Menginstal aplikasi sampel Amazon QLDB Node.js

Bagian ini menjelaskan cara menginstal dan menjalankan contoh aplikasi Amazon QLDB yang disediakan untuk step-by-step tutorial Node.js. Kasus penggunaan untuk aplikasi sampel ini adalah

database departemen kendaraan bermotor (DMV) yang melacak informasi historis lengkap tentang pendaftaran kendaraan.

Contoh aplikasi DMV untuk Node.js adalah open source di GitHub repositori aws-samples/ -nodejs. amazon-qldb-dmv-sample

Prasyarat

Sebelum Anda memulai, pastikan bahwa Anda menyelesaikan driver QLDB untuk Node.js. Prasyarat Ini termasuk menginstal Node.js dan melakukan hal berikut:

1. Daftar ke AWS.
2. Buat pengguna dengan izin QLDB yang sesuai.
3. Memberikan akses terprogram untuk pengembangan.

Untuk menyelesaikan semua langkah dalam tutorial ini, Anda memerlukan akses administratif penuh ke sumber daya buku besar Anda melalui QLDB API.

Instalasi

Untuk menginstal aplikasi sampel

1. Masukkan perintah berikut untuk mengkloning aplikasi sampel dari GitHub.

   2.x

   ```
   git clone https://github.com/aws-samples/amazon-qldb-dmv-sample-nodejs.git
   ```

   1.x

   ```
   git clone -b v1.0.0 https://github.com/aws-samples/amazon-qldb-dmv-sample-
   nodejs.git
   ```

   Aplikasi sampel mengemas kode sumber lengkap dari tutorial ini dan dependensinya, termasuk driver Node.js dan AWSSDK untuk JavaScript di Node.js. Aplikasi ini ditulis dalam TypeScript.
2. Beralih ke direktori tempat `amazon-qldb-dmv-sample-nodejs` paket dikloning.

   ```
   cd amazon-qldb-dmv-sample-nodejs
   ```

3. Lakukan instalasi dependensi yang bersih.

```
npm ci
```

4. Transpile paket.

```
npm run build
```

JavaScript File transpiled ditulis dalam `./dist` direktori.

5. Lanjutkan Langkah 1: Buat buku besar baru untuk memulai tutorial dan membuat buku besar.

## Langkah 1: Buat buku besar baru

Pada langkah ini, Anda membuat buku besar Amazon QLDB baru bernama. `vehicle-registration`

Untuk membuat buku besar baru

1. Tinjau file berikut (`Constants.ts`), yang berisi nilai konstan yang digunakan oleh semua program lain dalam tutorial ini.

```
/*
 * 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. Gunakan program berikut (`CreateLedger.ts`) untuk membuat buku besar bernama`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**
>
> - Dalam `createLedger` panggilan, Anda harus menentukan nama buku besar dan mode izin. Sebaiknya gunakan mode `STANDARD` izin untuk memaksimalkan keamanan data buku besar Anda.
> - Saat Anda membuat buku besar, perlindungan penghapusan diaktifkan secara default. Ini adalah fitur di QLDB yang mencegah buku besar dihapus oleh pengguna mana pun. Anda memiliki opsi untuk menonaktifkan perlindungan penghapusan pada pembuatan buku besar menggunakan QLDB API atau (). AWS Command Line Interface AWS CLI
> - Secara opsional, Anda juga dapat menentukan tag untuk dilampirkan ke buku besar Anda.

3. Untuk menjalankan program transpiled, masukkan perintah berikut.

```
node dist/CreateLedger.js
```

Untuk memverifikasi koneksi Anda ke buku besar baru, lanjutkan keLangkah 2: Uji konektivitas ke buku besar.

## Langkah 2: Uji konektivitas ke buku besar

Pada langkah ini, Anda memverifikasi bahwa Anda dapat terhubung ke `vehicle-registration` buku besar di Amazon QLDB menggunakan titik akhir API data transaksional.

Untuk menguji konektivitas ke buku besar

1. Gunakan program berikut (`ConnectToLedger.ts`) untuk membuat koneksi sesi data ke `vehicle-registration` buku besar.

   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**
>
> Untuk menjalankan transaksi data pada buku besar Anda, Anda harus membuat objek driver QLDB untuk terhubung ke buku besar tertentu. Ini adalah objek klien yang berbeda dari `qldbClient` objek yang Anda gunakan pada langkah sebelumnya untuk membuat buku besar. Klien sebelumnya hanya digunakan untuk menjalankan operasi API manajemen yang tercantum dalamReferensi API Amazon QLDB.

2. Untuk menjalankan program transpiled, masukkan perintah berikut.

```
node dist/ConnectToLedger.js
```

Untuk membuat tabel di `vehicle-registration` buku besar, lanjutkan keLangkah 3: Buat tabel, indeks, dan data sampel.

## Langkah 3: Buat tabel, indeks, dan data sampel

Saat buku besar QLDB Amazon Anda aktif dan menerima koneksi, Anda dapat mulai membuat tabel untuk data tentang kendaraan, pemiliknya, dan informasi pendaftarannya. Setelah membuat tabel dan indeks, Anda dapat memuatnya dengan data.

Pada langkah ini, Anda membuat empat tabel di `vehicle-registration` buku besar:

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

Anda juga membuat indeks berikut.

| Nama tabel | Bidang |
|---|---|
| `VehicleRegistration` | `VIN` |
| `VehicleRegistration` | `LicensePlateNumber` |
| `Vehicle` | `VIN` |
| `Person` | `GovId` |
| `DriversLicense` | `LicenseNumber` |
| `DriversLicense` | `PersonId` |

Saat memasukkan data sampel, Anda terlebih dahulu memasukkan dokumen ke dalam `Person` tabel. Kemudian, Anda menggunakan sistem yang ditetapkan `id` dari setiap `Person` dokumen untuk mengisi bidang yang sesuai dalam dokumen yang sesuai`VehicleRegistration`. `DriversLicense`

> **Tip**
>
> Sebagai praktik terbaik, gunakan sistem dokumen yang ditugaskan `id` sebagai kunci asing. Meskipun Anda dapat menentukan bidang yang dimaksudkan untuk menjadi pengidentifikasi

unik (misalnya, VIN kendaraan), pengidentifikasi unik sebenarnya dari dokumen adalah miliknya. `id` Bidang ini disertakan dalam metadata dokumen, yang dapat Anda kueri dalam tampilan komited (tampilan tabel yang ditentukan sistem).
Untuk informasi selengkapnya tentang tampilan di QLDB, lihat. Konsep inti Untuk mempelajari lebih lanjut tentang metadata, lihat. Melakukan Kueri Metadata Dokumen

Untuk membuat tabel dan indeks

1. Gunakan program berikut (`CreateTable.ts`) untuk membuat tabel yang disebutkan sebelumnya.

```
/*
 * 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**
>
> Program ini menunjukkan bagaimana menggunakan `executeLambda` fungsi dalam contoh driver QLDB. Dalam contoh ini, Anda menjalankan beberapa pernyataan `CREATE TABLE` PartiQL dengan ekspresi lambda tunggal.
> Fungsi eksekusi ini secara implisit memulai transaksi, menjalankan semua pernyataan di lambda, dan kemudian melakukan transaksi secara otomatis.

2. Untuk menjalankan program transpiled, masukkan perintah berikut.

```
node dist/CreateTable.js
```

3. Gunakan program berikut (`CreateIndex.ts`) untuk membuat indeks pada tabel, seperti yang dijelaskan sebelumnya.

```
/*
 * 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. Untuk menjalankan program transpiled, masukkan perintah berikut.

```
node dist/CreateIndex.js
```

Untuk memuat data ke dalam tabel

1. Tinjau `.ts` file-file berikut.

    1. `SampleData.ts`— Berisi data sampel yang Anda masukkan ke dalam `vehicle-registration` tabel.

```
/*
 * 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`- Modul utilitas yang mengimpor dari `ion-js` paket untuk menyediakan fungsi pembantu yang mengonversi, mengurai, dan mencetak data 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**
>
> `getDocumentId`Fungsi ini menjalankan kueri yang mengembalikan ID dokumen yang ditetapkan sistem dari tabel. Untuk mempelajari selengkapnya, lihat Menggunakan klausa BY untuk query ID dokumen.

2. Gunakan program berikut (`InsertDocument.ts`) untuk memasukkan data sampel ke dalam tabel Anda.

```
/*
 * 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**
>
> - Program ini menunjukkan cara memanggil `execute` fungsi dengan nilai parameter. Anda dapat meneruskan parameter data selain pernyataan PartiQL yang ingin Anda jalankan. Gunakan tanda tanya (?) sebagai placeholder variabel dalam string pernyataan Anda.
> - Jika sebuah `INSERT` pernyataan berhasil, ia mengembalikan setiap dokumen `id` yang disisipkan.

3. Untuk menjalankan program transpiled, masukkan perintah berikut.

```
node dist/InsertDocument.js
```

Selanjutnya, Anda dapat menggunakan `SELECT` pernyataan untuk membaca data dari tabel di `vehicle-registration` buku besar. Lanjut ke Langkah 4: Kueri tabel dalam buku besar.

## Langkah 4: Kueri tabel dalam buku besar

Setelah membuat tabel di buku besar QLDB Amazon dan memuatnya dengan data, Anda dapat menjalankan kueri untuk meninjau data registrasi kendaraan yang baru saja Anda masukkan. QLDB menggunakan PartiQL sebagai bahasa kueri dan Amazon Ion sebagai model data berorientasi dokumen.

PartiQL adalah bahasa kueri open-source yang kompatibel dengan SQL yang telah diperluas untuk bekerja dengan Ion. Dengan PartiQL, Anda dapat menyisipkan, menanyakan, dan mengelola data Anda dengan operator SQL yang sudah dikenal. Amazon Ion adalah superset dari JSON. Ion adalah format data berbasis dokumen open-source yang memberi Anda fleksibilitas dalam menyimpan dan memproses data terstruktur, semi-terstruktur, dan bersarang.

Pada langkah ini, Anda menggunakan `SELECT` pernyataan untuk membaca data dari tabel di `vehicle-registration` buku besar.

> **Warning**
>
> Saat Anda menjalankan kueri di QLDB tanpa pencarian yang diindeks, itu akan memanggil pemindaian tabel lengkap. PartiQL mendukung kueri seperti itu karena SQL kompatibel. Namun, jangan jalankan pemindaian tabel untuk kasus penggunaan produksi di QLDB. Pemindaian tabel dapat menyebabkan masalah kinerja pada tabel besar, termasuk konflik konkurensi dan batas waktu transaksi.
> Untuk menghindari pemindaian tabel, Anda harus menjalankan pernyataan dengan klausa `WHERE` predikat menggunakan operator kesetaraan pada bidang yang diindeks atau ID dokumen; misalnya, atau. `WHERE indexedField = 123` `WHERE indexedField IN (456, 789)` Untuk informasi selengkapnya, lihat Mengoptimalkan kinerja kueri.

Untuk menanyakan tabel

1. Gunakan program berikut (`FindVehicles.ts`) untuk menanyakan semua kendaraan yang terdaftar di bawah seseorang di buku besar Anda.

```
/*
 * 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**
>
> Pertama, program ini menanyakan `Person` tabel untuk dokumen `GovId LEWISR261LL` untuk mendapatkan bidang `id` metadata-nya.
> Kemudian, ia menggunakan dokumen ini `id` sebagai kunci asing untuk menanyakan `VehicleRegistration` tabel dengan`PrimaryOwner.PersonId`. Itu juga bergabung `VehicleRegistration` dengan `Vehicle` meja di `VIN` lapangan.

2. Untuk menjalankan program transpiled, masukkan perintah berikut.

```
node dist/FindVehicles.js
```

Untuk mempelajari tentang memodifikasi dokumen dalam tabel di `vehicle-registration` buku besar, lihat. Langkah 5: Ubah dokumen dalam buku besar

## Langkah 5: Ubah dokumen dalam buku besar

Sekarang setelah Anda memiliki data untuk dikerjakan, Anda dapat mulai membuat perubahan pada dokumen di `vehicle-registration` buku besar di Amazon QLDB. Pada langkah ini, contoh kode berikut menunjukkan bagaimana menjalankan pernyataan bahasa manipulasi data (DHTML). Pernyataan ini memperbarui pemilik utama satu kendaraan dan menambahkan pemilik sekunder ke kendaraan lain.

Untuk memodifikasi dokumen

1. Gunakan program berikut (`TransferVehicleOwnership.ts`) untuk memperbarui pemilik utama kendaraan dengan VIN `1N4AL11D75C109151` di buku besar Anda.

```
/*
 * 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. Untuk menjalankan program transpiled, masukkan perintah berikut.

```
node dist/TransferVehicleOwnership.js
```

3. Gunakan program berikut (`AddSecondaryOwner.ts`) untuk menambahkan pemilik sekunder ke kendaraan dengan VIN `KM8SRDHF6EU074761` di buku besar Anda.

```
/*
 * 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. Untuk menjalankan program transpiled, masukkan perintah berikut.

```
node dist/AddSecondaryOwner.js
```

Untuk meninjau perubahan ini di `vehicle-registration` buku besar, lihatLangkah 6: Lihat riwayat revisi untuk dokumen.

## Langkah 6: Lihat riwayat revisi untuk dokumen

Setelah memodifikasi data registrasi untuk kendaraan pada langkah sebelumnya, Anda dapat menanyakan riwayat semua pemilik terdaftar dan bidang yang diperbarui lainnya. Pada langkah ini, Anda menanyakan riwayat revisi dokumen dalam `VehicleRegistration` tabel di `vehicle-registration` buku besar Anda.

Untuk melihat riwayat revisi

1. Gunakan program berikut (`QueryHistory.ts`) untuk menanyakan riwayat revisi `VehicleRegistration` dokumen dengan 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**
>
> - Anda dapat melihat riwayat revisi dokumen dengan menanyakan built-in Fungsi Riwayat dalam sintaks berikut.
>
>   ```
>   SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
>   [ WHERE h.metadata.id = 'id' ]
>   ```
>
> - Waktu mulai dan akhir waktu keduanya opsional. Itu adalah nilai literal Amazon Ion yang dapat dilambangkan dengan backticks (). `` `...` `` Untuk informasi selengkapnya, lihat Menanyakan Ion dengan PartiQL di Amazon QLDB.
> - Sebagai praktik terbaik, kualifikasikan kueri riwayat dengan rentang tanggal (waktu mulai dan akhir waktu) dan ID dokumen (). `metadata.id` QLDB SELECT memproses kueri dalam transaksi, yang tunduk pada batas waktu tunggu transaksi.
>
>   Riwayat QLDB diindeks oleh ID dokumen, dan Anda tidak dapat membuat indeks riwayat tambahan saat ini. Pertanyaan sejarah yang mencakup waktu mulai dan waktu akhir mendapatkan manfaat dari kualifikasi rentang tanggal.

2. Untuk menjalankan program transpiled, masukkan perintah berikut.

```
node dist/QueryHistory.js
```

Untuk memverifikasi revisi dokumen secara kriptografis di `vehicle-registration` buku besar, lanjutkan ke. Langkah 7: Verifikasi dokumen dalam buku besar

## Langkah 7: Verifikasi dokumen dalam buku besar

Dengan Amazon QLDB, Anda dapat memverifikasi integritas dokumen secara efisien dalam jurnal buku besar Anda dengan menggunakan hashing kriptografi dengan SHA-256. Untuk mempelajari lebih lanjut tentang cara kerja verifikasi dan hashing kriptografi di QLDB, lihat. Verifikasi data di Amazon QLDB

Pada langkah ini, Anda memverifikasi revisi dokumen dalam `VehicleRegistration` tabel di `vehicle-registration` buku besar Anda. Pertama, Anda meminta intisari, yang dikembalikan sebagai file output dan bertindak sebagai tanda tangan dari seluruh riwayat perubahan buku besar

Anda. Kemudian, Anda meminta bukti untuk revisi relatif terhadap intisari itu. Dengan menggunakan bukti ini, integritas revisi Anda diverifikasi jika semua pemeriksaan validasi lulus.

Untuk memverifikasi revisi dokumen

1. Tinjau `.ts` file berikut, yang berisi objek QLDB yang diperlukan untuk verifikasi.

    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. Gunakan dua `.ts` program (`GetDigest.ts`dan`GetRevision.ts`) untuk melakukan langkah-langkah berikut:

   - Minta intisari baru dari `vehicle-registration` buku besar.
   - Minta bukti untuk setiap revisi dokumen dengan VIN `1N4AL11D75C109151` dari `VehicleRegistration` tabel.
   - Verifikasi revisi menggunakan intisari dan bukti yang dikembalikan dengan menghitung ulang intisari.

   `GetDigest.ts`Program ini berisi kode berikut.

```
/*
 * 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**
>
> Gunakan `getDigest` fungsi untuk meminta intisari yang mencakup ujung jurnal saat ini di buku besar Anda. Tip jurnal mengacu pada blok komitmen terbaru pada saat QLDB menerima permintaan Anda.

`GetRevision.ts`Program ini berisi kode berikut.

```
/*
 * 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**
>
> Setelah `getRevision` fungsi mengembalikan bukti untuk revisi dokumen yang ditentukan, program ini menggunakan API sisi klien untuk memverifikasi revisi tersebut.

3. Untuk menjalankan program transpiled, masukkan perintah berikut.

```
node dist/GetRevision.js
```

Jika Anda tidak perlu lagi menggunakan `vehicle-registration` buku besar, lanjutkan keLangkah 8 (opsional): Bersihkan sumber daya.

## Langkah 8 (opsional): Bersihkan sumber daya

Anda dapat terus menggunakan `vehicle-registration` buku besar. Namun, jika Anda tidak lagi membutuhkannya, Anda harus menghapusnya.

Untuk menghapus buku besar

1. Gunakan program berikut (`DeleteLedger.ts`) untuk menghapus `vehicle-registration` buku besar Anda dan semua isinya.

```
/*
 * 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**
>
> Jika proteksi penghapusan diaktifkan untuk buku besar Anda, Anda harus menonaktifkannya terlebih dahulu sebelum dapat menghapus buku besar menggunakan QLDB API.

2. Untuk menjalankan program transpiled, masukkan perintah berikut.

```
node dist/DeleteLedger.js
```

## Tutorial Amazon QLDB Python

Dalam implementasi aplikasi contoh tutorial ini, Anda menggunakan driver Amazon QLDB denganAWS SDK for Python (Boto3) untuk membuat buku besar QLDB dan mengisinya dengan data sampel.

Dalam menggunakan tutorial ini, Anda dapat merujuk pada klien QLDB level rendah dalam AWSSDK for Python (Boto3) merujuk pada operasi API manajemen. Untuk operasi data transaksional, Anda dapat merujuk ke QLDB Driver untuk Python API Referensi.

> **Note**
>
> Jika berlaku, beberapa langkah tutorial memiliki perintah atau contoh kode yang berbeda untuk setiap versi utama driver QLDB yang didukung untuk Python.

Topik



## Menginstal aplikasi sampel Amazon QLDB Python

Bagian ini menjelaskan cara menginstal dan menjalankan aplikasi sampel Amazon QLDB yang disediakan untuk tutorial step-by-step Python. Kasus penggunaan untuk aplikasi sampel ini adalah departemen database kendaraan bermotor (DMV) yang melacak informasi historis lengkap tentang pendaftaran kendaraan.

Aplikasi sampel DMV untuk Python adalah open source di GitHub repositori aws-samples/amazon-qldb-dmv-sample -python.

Prasyarat

Sebelum memulai, pastikan Anda menyelesaikan driver QLDB for PythonPrasyarat. Ini termasuk menginstal Python dan melakukan hal berikut:

1. Daftar ke AWS.
2. Buat pengguna dengan izin QLDB yang sesuai.
3. Memberikan akses terprogram untuk pengembangan.

Untuk menyelesaikan semua langkah dalam tutorial ini, Anda memerlukan akses administratif penuh ke sumber daya buku besar Anda melalui QLDB API.

Instalasi

Untuk menginstal aplikasi sampel

1. Masukkan`pip` perintah berikut untuk mengkloning dan menginstal aplikasi sampel dari 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
   ```

   Contoh aplikasi paket kode sumber lengkap dari tutorial ini dan dependensinya, termasuk driver Python dan AWS SDK for Python (Boto3).
2. Sebelum Anda mulai menjalankan kode di baris perintah, alihkan direktori kerja Anda saat ini ke lokasi di mana`pyqldbsamples` paket diinstal. Masukkan perintah berikut.

   ```
   cd $(python -c "import pyqldbsamples; print(pyqldbsamples.__path__[0])")
   ```
3. LanjutkanLangkah 1: Membuat buku besar baru untuk memulai tutorial dan membuat buku besar.

## Langkah 1: Membuat buku besar baru

Pada langkah ini, Anda membuat buku besar Amazon QLDB baru bernama`vehicle-registration`.

Untuk membuat buku besar baru

1. Tinjau file berikut (`constants.py`), yang berisi nilai-nilai konstan yang digunakan oleh semua program lain dalam tutorial ini.

   ```
   # 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. Gunakan program berikut (`create_ledger.py`) untuk membuat buku besar bernama`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**
>
> - Dalam`create_ledger` panggilan, Anda harus menentukan nama buku besar dan mode izin. Kami merekomendasikan untuk menggunakan mode`STANDARD` izin untuk memaksimalkan keamanan data buku besar Anda.
> - Dalam menggunakan buku besar, perlindungan penghapusan diaktifkan secara default. Ini adalah fitur dalam QLDB yang mencegah buku besar dihapus oleh pengguna manapun. Anda memiliki opsi untuk menonaktifkan perlindungan penghapusan pada pembuatan buku besar menggunakan QLDB API atauAWS Command Line Interface (AWS CLI).
> - Opsional, Anda juga dapat menentukan tag untuk dilampirkan pada buku besar Anda.

3. Untuk menjalankan program, masukkan perintah berikut.

```
python create_ledger.py
```

Untuk memverifikasi koneksi Anda ke buku besar baru, lanjutkan keLangkah 2: Menguji konektivitas ke buku besar.

## Langkah 2: Menguji konektivitas ke buku besar

Pada langkah ini, Anda memverifikasi bahwa Anda dapat terhubung ke`vehicle-registration` buku besar di Amazon QLDB menggunakan endpoint API data transaksional.

Untuk menguji konektivitas ke buku besar

1. Gunakan program berikut (`connect_to_ledger.py`) untuk membuat koneksi sesi data ke`vehicle-registration` buku besar.

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**
>
> - Untuk menjalankan transaksi data pada buku besar Anda, Anda harus membuat objek driver QLDB untuk terhubung ke buku besar tertentu. Ini adalah objek klien yang berbeda dari`qldb_client` objek yang Anda gunakan dalam langkah sebelumnya untuk membuat buku besar. Klien sebelumnya hanya digunakan untuk menjalankan operasi API manajemen yang tercantum dalamReferensi API Amazon QLDB.
> - Anda harus menentukan nama buku besar saat Anda membuat objek driver ini.

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**
>
> - Untuk menjalankan transaksi data pada buku besar Anda, Anda harus membuat objek driver QLDB untuk terhubung ke buku besar tertentu. Ini adalah objek klien yang berbeda dari`qldb_client` objek yang Anda gunakan dalam langkah sebelumnya untuk membuat buku besar. Klien sebelumnya hanya digunakan untuk menjalankan operasi API manajemen yang tercantum dalamReferensi API Amazon QLDB.
> - Pertama, membuat obyek driver QLDB dikumpulkan. Anda harus menentukan nama buku besar saat Anda membuat driver ini.
> - Kemudian, Anda dapat membuat sesi dari objek driver yang dikumpulkan ini.

2. Untuk menjalankan program, masukkan perintah berikut.

```
python connect_to_ledger.py
```

Untuk membuat tabel di`vehicle-registration` buku besar, lanjutkan keLangkah 3: Buat tabel, indeks, dan data sampel.

## Langkah 3: Buat tabel, indeks, dan data sampel

Saat buku besar Amazon QLDB Anda aktif dan menerima koneksi, Anda dapat mulai membuat tabel untuk data tentang kendaraan, pemiliknya, dan informasi pendaftarannya. Setelah membuat tabel dan indeks, Anda dapat memuatnya dengan data.

Pada langkah ini, Anda membuat empat tabel di`vehicle-registration` buku besar:

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

Anda juga membuat indeks berikut.

| Nama tabel | Bidang |
|---|---|
| `VehicleRegistration` | `VIN` |
| `VehicleRegistration` | `LicensePlateNumber` |
| `Vehicle` | `VIN` |
| `Person` | `GovId` |
| `DriversLicense` | `LicenseNumber` |
| `DriversLicense` | `PersonId` |

Saat memasukkan data sampel, Anda terlebih dahulu memasukkan dokumen ke dalam`Person` tabel. Kemudian, Anda menggunakan sistem-ditugaskan`id` dari setiap`Person` dokumen untuk mengisi bidang yang sesuai dalam`DriversLicense` dokumen`VehicleRegistration` dan yang sesuai.

> **Tip**
>
> Sebagai praktik terbaik, gunakan sistem dokumen yang ditugaskan`id` sebagai kunci asing. Meskipun Anda dapat menentukan bidang yang dimaksudkan untuk menjadi pengidentifikasi unik (misalnya, VIN kendaraan), pengenal unik sebenarnya dari dokumen adalah`id`. Bidang ini termasuk dalam metadata dokumen, yang dapat Anda kueri dalam tampilan berkomitmen (tampilan tabel yang ditentukan sistem).
> Untuk informasi lebih lanjut tentang tampilan di QLDB, lihatKonsep inti. Untuk mempelajari selengkapnya tentang metadata, lihatMelakukan Kueri Metadata Dokumen.

Untuk membuat tabel dan indeks

1. Gunakan program berikut (`create_table.py`) untuk membuat tabel yang disebutkan sebelumnya.

   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**
>
> Program ini menunjukkan bagaimana menggunakan`execute_lambda` fungsi. Dalam contoh ini, Anda menjalankan beberapa pernyataan`CREATE TABLE` PartiQL dengan ekspresi lambda tunggal.
> Fungsi eksekusi ini secara implisit memulai transaksi, menjalankan semua pernyataan di lambda, dan kemudian melakukan transaksi secara otomatis.

2. Untuk menjalankan program, masukkan perintah berikut.

```
python create_table.py
```

3. Gunakan program berikut (`create_index.py`) untuk membuat indeks pada tabel, seperti yang dijelaskan sebelumnya.

   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. Untuk menjalankan program, masukkan perintah berikut.

```
python create_index.py
```

Untuk memuat data ke dalam tabel

1. Tinjau file berikut (`sample_data.py`), yang mewakili data sampel yang Anda masukkan ke dalam`vehicle-registration` tabel. File ini juga mengimpor dari`amazon.ion` paket untuk menyediakan fungsi pembantu yang mengonversi, mengurai, dan mencetak data 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**
>
> `get_document_ids`Fungsi ini menjalankan query yang mengembalikan ID dokumen sistem-ditugaskan dari tabel. Untuk mempelajari selengkapnya, lihat Menggunakan klausa BY untuk query ID dokumen.

2. Gunakan program berikut (`insert_document.py`) untuk memasukkan data sampel ke dalam tabel Anda.

   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**
>
> - Program ini menunjukkan bagaimana memanggil`execute_statement` fungsi dengan nilai-nilai parameter. Anda dapat melewati parameter data selain pernyataan PartiQL yang ingin Anda jalankan. Gunakan tanda tanya (?) sebagai placeholder variabel dalam string pernyataan Anda.
> - Jika`INSERT` pernyataan berhasil, ia mengembalikan`id` dari setiap dokumen dimasukkan.

3. Untuk menjalankan program, masukkan perintah berikut.

```
python insert_document.py
```

Selanjutnya, Anda dapat menggunakan`SELECT` pernyataan untuk membaca data dari tabel di`vehicle-registration` buku besar. Lanjut ke Langkah 4: Cari tabel dalam buku besar.

## Langkah 4: Cari tabel dalam buku besar

Setelah membuat tabel di buku besar Amazon QLDB dan memuatnya dengan data, Anda dapat menjalankan kueri untuk meninjau data registrasi kendaraan yang baru saja Anda masukkan. QLDB menggunakan PartiQL sebagai bahasa kueri dan Amazon Ion sebagai model data berorientasi dokumen.

PartiQL adalah open-source, bahasa query SQL-kompatibel yang telah diperluas untuk bekerja dengan Ion. Dengan PartiQL, Anda dapat memasukkan, query, dan mengelola data Anda dengan operator SQL akrab. Amazon Ion adalah superset dari JSON. Ion adalah open-source, format data berbasis dokumen yang memberi Anda fleksibilitas menyimpan dan memproses data terstruktur, semistruktural, dan bersarang.

Pada langkah ini, Anda menggunakan`SELECT` pernyataan untuk membaca data dari tabel di`vehicle-registration` buku besar.

> **Warning**
>
> Ketika Anda menjalankan query di QLDB tanpa pencarian diindeks, memanggil scan tabel penuh. PartiQL mendukung query tersebut karena SQL kompatibel. Namun, jangan jalankan pemindaian tabel untuk kasus penggunaan produksi di QLDB. Pemindaian tabel dapat menyebabkan masalah kinerja pada tabel besar, termasuk konflik konkurensi dan batas waktu transaksi.
>
> Untuk menghindari pemindaian tabel, Anda harus menjalankan pernyataan dengan klausa`WHERE` predikat menggunakan operator kesetaraan pada bidang yang diindeks atau ID dokumen; misalnya,`WHERE indexedField = 123` atau`WHERE indexedField IN (456, 789)`. Untuk informasi selengkapnya, lihat Mengoptimalkan kinerja kueri.

Untuk query tabel

1. Gunakan program berikut (`find_vehicles.py`) untuk meminta semua kendaraan yang terdaftar di bawah seseorang di buku besar Anda.

   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**
>
> Pertama, program ini query`Person` tabel untuk dokumen dengan`GovId LEWISR261LL` untuk mendapatkan bidang`id` metadata.

Kemudian, menggunakan dokumen ini`id` sebagai kunci asing untuk query`VehicleRegistration` tabel oleh`PrimaryOwner.PersonId`. Ini juga bergabung`VehicleRegistration` dengan`Vehicle` meja di`VIN` lapangan.

2. Untuk menjalankan program, masukkan perintah berikut.

```
python find_vehicles.py
```

Untuk mempelajari tentang memodifikasi dokumen dalam tabel di`vehicle-registration` buku besar, lihatLangkah 5: Memodifikasi dokumen dalam buku besar.

## Langkah 5: Memodifikasi dokumen dalam buku besar

Sekarang setelah Anda memiliki data untuk digunakan, Anda dapat mulai membuat perubahan pada dokumen dalam`vehicle-registration` buku besar di Amazon QLDB. Dalam menggunakan contoh kode berikut menunjukkan bagaimana menjalankan pernyataan bahasa manipulasi data (DMLL). Pernyataan ini memperbarui pemilik utama satu kendaraan dan menambahkan pemilik sekunder ke kendaraan lain.

Untuk mengubah dokumen

1. Gunakan program berikut (`transfer_vehicle_ownership.py`) untuk memperbarui pemilik utama kendaraan dengan VIN`1N4AL11D75C109151` di buku besar Anda.

   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. Untuk menjalankan program, masukkan perintah berikut.

```
python transfer_vehicle_ownership.py
```

3. Gunakan program berikut (`add_secondary_owner.py`) untuk menambahkan pemilik sekunder ke kendaraan dengan VIN`KM8SRDHF6EU074761` di buku besar Anda.

   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. Untuk menjalankan program, masukkan perintah berikut.

```
python add_secondary_owner.py
```

Untuk meninjau perubahan ini dalam`vehicle-registration` buku besar, lihatLangkah 6: Lihat riwayat revisi dokumen.

## Langkah 6: Lihat riwayat revisi dokumen

Setelah memodifikasi data pendaftaran untuk kendaraan pada langkah sebelumnya, Anda dapat menanyakan riwayat semua pemilik terdaftar dan bidang terbaru lainnya. Pada langkah ini, Anda menanyakan riwayat revisi dokumen dalam`VehicleRegistration` tabel di`vehicle-registration` buku besar Anda.

Untuk melihat riwayat revisi

1. Gunakan program berikut (`query_history.py`) untuk query sejarah revisi`VehicleRegistration` dokumen dengan VIN`1N4AL11D75C109151`.

   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**
>
> - Anda dapat melihat riwayat revisi dokumen dengan menanyakan built-inFungsi Riwayat dalam sintaks berikut.
>
>   ```
>   SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
>   [ WHERE h.metadata.id = 'id' ]
>   ```
>
> - Start-time dan end-time keduanya opsional. Mereka adalah nilai literal Amazon Ion yang dapat dilambangkan dengan backticks (`` `...` ``). Untuk mempelajari selengkapnya, lihat Menanyakan Ion dengan PartiQL di Amazon QLDB.
> - Sebagai praktik terbaik, kualifikasi kueri riwayat dengan rentang tanggal (start-time dan end-time) dan ID dokumen (`metadata.id`). QLDB memprosesSELECT kueri dalam transaksi, yang tunduk pada batas batas waktu transaksi.
>
>   Riwayat QLDB diindeks oleh ID dokumen, dan Anda tidak dapat membuat indeks riwayat tambahan saat ini. Kueri riwayat yang menyertakan waktu mulai dan waktu akhir mendapatkan manfaat kualifikasi rentang tanggal.

2. Untuk menjalankan program, masukkan perintah berikut.

```
python query_history.py
```

Untuk memverifikasi revisi dokumen secara kriptografis di`vehicle-registration` buku besar, lanjutkan keLangkah 7: Verifikasi dokumen di buku besar.

## Langkah 7: Verifikasi dokumen di buku besar

Dengan Amazon QLDB, Anda dapat secara efisien memverifikasi integritas dokumen dalam jurnal buku besar Anda dengan menggunakan hashing kriptografi dengan SHA-256. Untuk mempelajari lebih lanjut tentang cara kerja verifikasi dan hashing kriptografi di QLDB, lihatVerifikasi data di Amazon QLDB.

Pada langkah ini, Anda memverifikasi revisi dokumen dalam`VehicleRegistration` tabel di`vehicle-registration` buku besar Anda. Pertama, Anda meminta digest, yang dikembalikan sebagai file output dan bertindak sebagai tanda tangan dari seluruh riwayat perubahan buku besar Anda. Kemudian, Anda meminta bukti untuk revisi relatif terhadap intisari itu. Dengan menggunakan bukti ini, integritas revisi Anda diverifikasi jika semua pemeriksaan validasi lulus.

Untuk memverifikasi revisi dokumen

1. Tinjau`.py` file berikut, yang mewakili objek QLDB yang diperlukan untuk verifikasi dan modul utilitas dengan fungsi pembantu untuk mengubah jenis respon QLDB ke string.

   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. Gunakan dua `.py` program (`get_digest.py`dan`get_revision.py`) untuk melakukan langkah-langkah berikut:

   - Minta intisari baru dari`vehicle-registration` buku besar.
   - Minta bukti untuk setiap revisi dokumen dengan VIN`1N4AL11D75C109151` dari`VehicleRegistration` tabel.
   - Verifikasi revisi menggunakan intisari dan bukti yang dikembalikan dengan menghitung ulang intisari.

   `get_digest.py`Program ini berisi kode berikut.

```
# 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**
>
> Gunakan`get_digest_result` fungsi untuk meminta intisari yang mencakup ujung jurnal saat ini di buku besar Anda. Ujung jurnal mengacu pada blok berkomitmen terbaru pada saat QLDB menerima permintaan Anda.

`get_revision.py`Program ini berisi kode berikut.

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**
>
> Setelah`get_revision` fungsi mengembalikan bukti untuk revisi dokumen yang ditentukan, program ini menggunakan API sisi klien untuk memverifikasi revisi itu.

3. Untuk menjalankan program, masukkan perintah berikut.

```
python get_revision.py
```

Jika Anda tidak perlu lagi menggunakan`vehicle-registration` buku besar, lanjutkan keLangkah 8 (opsional): Bersihkan Sumber Daya.

## Langkah 8 (opsional): Bersihkan Sumber Daya

Anda dapat terus menggunakan`vehicle-registration` buku besar. Namun, jika Anda tidak lagi membutuhkannya, Anda harus menghapusnya.

Untuk menghapus buku besar

1. Gunakan program berikut (`delete_ledger.py`) untuk menghapus`vehicle-registration` buku besar Anda dan semua isinya.

```
# 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**
>
> Jika perlindungan penghapusan diaktifkan untuk buku besar Anda, Anda harus menonaktifkannya terlebih dahulu sebelum dapat menghapus buku besar menggunakan QLDB API.

`delete_ledger.py`File ini juga memiliki ketergantungan pada program berikut (`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. Untuk menjalankan program, masukkan perintah berikut.

```
python delete_ledger.py
```

# Bekerja dengan tipe data Amazon Ion di Amazon QLDB

Amazon QLDB menyimpan datanya dalam format Amazon Ion. Untuk bekerja dengan data di QLDB, Anda harus menggunakan pustaka Ion sebagai dependensi untuk bahasa pemrograman yang didukung.

Di bagian ini, pelajari cara mengonversi data dari tipe asli ke setara Ion mereka, dan sebaliknya. Panduan referensi ini menunjukkan contoh kode yang menggunakan driver QLDB untuk memproses data Ion dalam buku besar QLDB. Ini menunjukkan contoh kode untuk Java, .NET (C#), Go, Node.js (TypeScript), dan Python.

Topik

- Prasyarat

- Bool
- Int
- Desimal
- Decimal
- Timestamp
- String
- blob
- Daftar
- Struct
- Nilai null dan tipe dinamis
- Down-mengkonversi ke JSON

## Prasyarat

Contoh kode berikut mengasumsikan bahwa Anda memiliki contoh driver QLDB yang terhubung ke buku besar aktif dengan tabel bernama`ExampleTable`. Tabel berisi satu dokumen yang ada yang memiliki delapan bidang berikut:

- ExampleBool
- ExampleInt
- ExampleFloat
- ExampleDecimal
- ExampleTimestamp
- ExampleString
- ExampleBlob
- ExampleList

> **Note**
>
> Untuk keperluan referensi ini, asumsikan bahwa jenis yang disimpan di setiap bidang cocok dengan namanya. Dalam prakteknya, QLDB tidak menegakkan skema atau tipe data definisi untuk bidang dokumen.

# Bool

Contoh kode berikut ini menunjukkan cara memproses cara memproses tipe boolean 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**
>
> Untuk mengkonversi ke kode sinkron, menghapus`await` dan`async` kata kunci, dan mengubah`IAsyncResult` jenis untuk`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

Contoh kode berikut ini menunjukkan cara memproses cara memproses tipe integer.

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**
>
> Untuk mengkonversi ke kode sinkron, menghapus`await` dan`async` kata kunci, dan mengubah`IAsyncResult` jenis untuk`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))
```

## Desimal

Contoh kode berikut ini menunjukkan cara memproses cara memproses tipe mengambang Ion.

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

#### Menggunakan 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**
>
> Untuk mengkonversi ke kode sinkron, menghapus`await` dan`async` kata kunci, dan mengubah`IAsyncResult` jenis untuk`IResult`.

### Menggunakan C # ganda

```
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**
>
> Untuk mengkonversi ke kode sinkron, menghapus`await` dan`async` kata kunci, dan mengubah`IAsyncResult` jenis untuk`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))
```

# Decimal

Contoh kode berikut ini menunjukkan cara memproses cara memproses tipe desimal.

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**
>
> Untuk mengkonversi ke kode sinkron, menghapus`await` dan`async` kata kunci, dan mengubah`IAsyncResult` jenis untuk`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**
>
> Anda juga dapat menggunakan`ionValue.numberValue()` untuk mengubah desimal Ion ke JavaScript angka. Menggunakan`ionValue.numberValue()` memiliki kinerja yang lebih baik, tetapi kurang tepat.

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))
```

## Timestamp

Contoh kode berikut ini menunjukkan cara memproses cara memproses tipe 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**
>
> Untuk mengkonversi ke kode sinkron, menghapus`await` dan`async` kata kunci, dan mengubah`IAsyncResult` jenis untuk`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))
```

## String

Contoh kode berikut ini menunjukkan cara memproses cara memproses tipe string 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**
>
> Untuk mengkonversi ke kode sinkron, menghapus`await` dan`async` kata kunci, dan mengubah`IAsyncResult` jenis untuk`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

Contoh kode berikut ini menunjukkan cara memproses cara memproses tipe Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();
```

```
byte[] exampleBytes = driver.execute((txn) -> {
    // Transforming a Java byte array to Ion
    // Transform any arbitrary data to a byte array to store in QLDB
    String aString = "Hello world!";
    byte[] aByteArray = aString.getBytes();
    IonValue ionBlob = ionSystem.newBlob(aByteArray);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleBlob = ?", ionBlob);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleBlob from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java byte array
        // Cast IonValue to IonBlob first
        aByteArray = ((IonBlob)ionValue).getBytes();
    }

    // exampleBytes is now the value fetched from QLDB
    return aByteArray;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
using System.Text;
...

IValueFactory valueFactory = new ValueFactory();

// Transform any arbitrary data to a byte array to store in QLDB.
string nativeString = "Hello world!";
byte[] nativeByteArray = Encoding.UTF8.GetBytes(nativeString);
// Transforming a C# byte array to Ion.
IIonValue ionBlob = valueFactory.NewBlob(nativeByteArray);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleBlob = ?", ionBlob);
```

```
    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleBlob from ExampleTable");
});

byte[] retrievedByteArray = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# byte array.
    retrievedByteArray = ionValue.Bytes().ToArray();
}
```

> **Note**
>
> Untuk mengkonversi ke kode sinkron, menghapus`await` dan`async` kata kunci, dan mengubah`IAsyncResult` jenis untuk`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))
```

## Daftar

Contoh kode berikut ini menunjukkan cara memproses cara memproses tipe daftar 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**
>
> Untuk mengkonversi ke kode sinkron, menghapus`await` dan`async` kata kunci, dan mengubah`IAsyncResult` jenis untuk`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

Dalam QLDB, tipe`struct` data yang sangat unik dari jenis Ion lainnya. Dokumen tingkat atas yang Anda masukkan ke dalam tabel harus dari`struct` jenisnya. Bidang dokumen juga dapat menyimpan bersarang`struct`.

Untuk mempermudah, contoh berikut mendefinisikan dokumen yang memiliki`ExampleString` dan`ExampleInt` bidang saja.

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;
});
```

Atau, Anda dapat menggunakan perpustakaan Jackson untuk memetakan tipe data ke dan dari Ion. Pustaka mendukung tipe data Ion lainnya, tetapi contoh ini berfokus pada`struct` jenisnya.

```
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

Gunakan perpustakaan Amazon.qldb.driver.serialization untuk memetakan tipe data C # asli ke dan dari Ion. Pustaka mendukung tipe data Ion lainnya, tetapi contoh ini berfokus pada`struct` jenisnya.

```
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**
>
> Untuk mengkonversi ke kode sinkron, menghapus`await` dan`async` kata kunci, dan mengubah`IAsyncResult` jenis untuk`IResult`.

Atau, Anda dapat menggunakan Amazon. IonDotnet.Builders perpustakaan untuk memproses tipe data 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))
```

## Nilai null dan tipe dinamis

QLDB mendukung konten terbuka dan tidak menegakkan skema atau tipe data definisi untuk bidang dokumen. Anda juga dapat menyimpan nilai null Ion dalam dokumen QLDB. Semua contoh sebelumnya mengasumsikan bahwa setiap tipe data yang dikembalikan diketahui dan tidak null. Contoh berikut menunjukkan bagaimana bekerja dengan Ion ketika tipe data tidak diketahui atau mungkin null.

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

Keterbatasan Marshalling

Di Go,`nil` nilai-nilai tidak mempertahankan jenisnya ketika Anda marshal dan kemudian unmarshal mereka. `nil`Nilai marshals untuk Ion null, tapi Ion null unmarshals ke nilai nol daripada`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!
```

## Down-mengkonversi ke JSON

Jika aplikasi Anda memerlukan kompatibilitas JSON, Anda dapat mengubah data Amazon Ion ke JSON. Namun, mengubah Ion ke JSON hilang dalam kasus tertentu di mana data Anda menggunakan tipe Ion kaya yang tidak ada di JSON.

Untuk detail tentang aturan konversi Ion ke JSON, lihat Konversi ke JSON di Amazon Ion Cookbook.

# Bekerja dengan data dan riwayat di Amazon QLDB

Topik berikut memberikan contoh dasar pernyataan buat, perbarui, perbarui, dan hapus). Anda dapat menjalankan pernyataan ini secara manual dengan menggunakan editor PartiQL pada konsol QLDB, atau shell QLDB. Panduan ini juga memandu Anda melalui proses bagaimana QLDB menangani data Anda saat Anda membuat perubahan dalam buku besar Anda.

QLDB mendukung bahasa query PartiQL.

Untuk contoh kode yang menunjukkan cara pemrograman menjalankan pernyataan serupa menggunakan driver QLDB, lihat tutorial diMemulai dengan driver.

> **Tip**
>
> Berikut ini adalah ringkasan singkat tips dan praktik terbaik untuk bekerja dengan PartiQL di QLDB:
>
> - Memahami batas konkurensi dan transaksi — Semua pernyataan, termasuk`SELECT` kueri, tunduk pada konflik kontrol konkurensi (OCC) yang optimis dan batas transaksi, termasuk batas waktu transaksi 30 detik.
> - Gunakan indeks - Gunakan indeks kardinalitas tinggi dan jalankan kueri yang ditargetkan untuk mengoptimalkan pernyataan Anda dan menghindari pemindaian tabel penuh. Untuk mempelajari selengkapnya, lihat Mengoptimalkan kinerja kueri.
> - Gunakan predikat kesetaraan - Pencarian terindeks memerlukan operator kesetaraan (`=`atau`IN`). Operator ketidaksetaraan (`<>`,`LIKE`,,,`BETWEEN`) tidak memenuhi syarat untuk pencarian yang diindeks dan menghasilkan pemindaian tabel penuh.
> - Gunakan batin bergabung saja - QLDB mendukung batin bergabung saja. Sebagai praktik terbaik, bergabunglah di bidang yang diindeks untuk setiap tabel yang Anda ikuti. Pilih indeks kardinalitas tinggi untuk kriteria gabungan dan predikat kesetaraan.

Topik

- Memperbarui dan menghapus dokumen
- Melakukan Kueri Riwayat Riwayat Revisi
- Menyunting revisi dokumen
- Mengoptimalkan kinerja kueri
- Mendapatkan statistik pernyataan PartiQL
- Menanyakan katalog sistem
- Mengelola tabel
- Mengelola indeks
- ID unik di Amazon QLDB

# Membuat tabel dengan indeks dan memasukkan dokumen

Setelah membuat buku besar Amazon QLDB, langkah pertama Anda adalah membuat tabel dengan CREATE TABLE pernyataan dasar. Tabel terdiri dari Dokumen QLDB B B B B B B B, yang merupakan kumpulan data dalam `struct` format Amazon Ion.

Topik

- Membuat tabel dan indeks
- Memasukkan dokumen

## Membuat tabel dan indeks

Tabel memiliki sederhana, nama case-sensitive tanpa ruang nama. QLDB mendukung konten terbuka dan tidak menegakkan skema, sehingga Anda tidak menentukan atribut atau tipe data saat membuat tabel.

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle
```

Sebuah `CREATE TABLE` pernyataan mengembalikan ID sistem-ditugaskan dari tabel baru. Semua ID yang ditetapkan sistem di QLDB adalah pengidentifikasi unik universal (UUID) yang masing-masing diwakili dalam string yang dikodekan Base62.

> **Note**
>
> Secara opsional, Anda dapat menentukan tag untuk sumber daya tabel saat Anda membuat tabel. Untuk mempelajari caranya, lihat Beri tag pada tabel saat penciptaan.

Anda juga dapat membuat indeks pada tabel untuk mengoptimalkan kinerja kueri.

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

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

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

> **Important**
>
> QLDB membutuhkan indeks untuk secara efisien mencari dokumen. Tanpa indeks, QLDB perlu melakukan pemindaian meja penuh saat membaca dokumen. Hal ini dapat menyebabkan masalah kinerja pada tabel besar, termasuk konflik konkurensi dan batas waktu transaksi.
>
> Untuk menghindari pemindaian tabel, Anda harus menjalankan pernyataan dengan klausa`WHERE` predikat menggunakan operator kesetaraan (`=`or`IN`) pada bidang yang diindeks atau ID dokumen. Untuk informasi selengkapnya, lihat Mengoptimalkan kinerja kueri.

Perhatikan kendala berikut saat membuat indeks:

- Indeks hanya dapat dibuat pada satu bidang tingkat atas. Indeks komposit, bersarang, unik, dan berbasis fungsi tidak didukung.
- Anda dapat membuat indeks pada jenis data Ion apa pun, termasuk`list` dan`struct`. Namun, Anda hanya dapat melakukan pencarian terindeks dengan kesetaraan seluruh nilai Ion terlepas dari jenis Ion. Misalnya, saat menggunakan`list` tipe sebagai indeks, Anda tidak dapat melakukan pencarian terindeks oleh satu item di dalam daftar.
- Kinerja kueri ditingkatkan hanya jika Anda menggunakan predikat kesetaraan; misalnya,`WHERE indexedField = 123` atau`WHERE indexedField IN (456, 789)`.

QLDB tidak menghormati ketidaksetaraan dalam predikat query. Akibatnya, pemindaian rentang yang difilter tidak diimplementasikan.

- Nama bidang yang diindeks adalah huruf kapital dan dapat memiliki maksimal 128 karakter.
- Pembuatan indeks di QLDB bersifat asinkron. Jumlah waktu yang dibutuhkan untuk menyelesaikan membangun indeks pada tabel yang tidak kosong bervariasi tergantung pada ukuran tabel. Untuk informasi selengkapnya, lihat Mengelola indeks.

## Memasukkan dokumen

Kemudian Anda dapat memasukkan dokumen ke dalam tabel Anda. Dokumen QLDB disimpan dalam format Amazon Ion. SISIPKANPernyataan PartiQL berikut termasuk bagian dari data sampel pendaftaran kendaraan yang digunakan dalamMemulai dengan konsol 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'
} >>
```

sintaks PartiQL dan semantik

- Nama bidang disertakan dalam tanda kutip tunggal (`'...'`).
- Nilai string juga tertutup dalam tanda kutip tunggal (`'...'`).
- Stempel waktu disertakan dalam backtick (`` `...` ``). Backticks dapat digunakan untuk menunjukkan literal Ion apa pun.
- Bilangan bulat dan desimal adalah nilai literal yang tidak perlu dilambangkan.

Untuk rincian lebih lanjut tentang sintaks dan semantik PartiQL, lihatMenanyakan Ion dengan PartiQL di Amazon QLDB.

`INSERT`Pernyataan menciptakan revisi awal dokumen dengan nomor versi nol. Untuk mengidentifikasi setiap dokumen secara unik, QLDB menetapkan ID dokumen sebagai bagian dari metadata. Insert pernyataan mengembalikan ID dari setiap dokumen yang dimasukkan.

> **Important**
>
> Karena QLDB tidak menegakkan skema, Anda dapat memasukkan dokumen yang sama ke dalam tabel beberapa kali. Setiap pernyataan insert melakukan entri dokumen terpisah ke jurnal, dan QLDB memberikan setiap dokumen ID unik.

Untuk mempelajari cara menanyakan dokumen yang Anda masukkan ke dalam tabel Anda, lanjutkan keMelakukan Kueri Data.

# Melakukan Kueri Data

Tampilan pengguna mengembalikan revisi terbaru yang tidak dihapus dari data pengguna Anda saja. Ini adalah tampilan default di Amazon QLDB. Ini berarti bahwa tidak ada kualifikasi khusus yang diperlukan ketika Anda ingin query hanya data Anda.

Untuk detail tentang sintaks dan parameter dari contoh kueri berikut, lihatPilih di referensi Amazon QLDB PartiQL.

Topik

- Kueri dasar
- Proyeksi dan filter
- Bergabung
- Data yang tertumpuk

## Kueri dasar

`SELECT`Kueri dasar mengembalikan dokumen yang Anda masukkan ke dalam tabel.

> **Warning**
>
> Ketika Anda menjalankan query di QLDB tanpa pencarian diindeks, memanggil scan tabel penuh. PartiQL mendukung query tersebut karena SQL kompatibel. Namun, jangan jalankan pemindaian tabel untuk kasus penggunaan produksi di QLDB. Pemindaian tabel dapat menyebabkan masalah kinerja pada tabel besar, termasuk konflik konkurensi dan batas waktu transaksi.

Untuk menghindari pemindaian tabel, Anda harus menjalankan pernyataan dengan klausa`WHERE` predikat menggunakan operator kesetaraan pada bidang yang diindeks atau ID dokumen; misalnya,`WHERE indexedField = 123` atau`WHERE indexedField IN (456, 789)`. Untuk informasi selengkapnya, lihat Mengoptimalkan kinerja kueri.

Kueri berikut menunjukkan hasil untuk dokumen registrasi kendaraan yang sebelumnya Anda masukkanMembuat tabel dengan indeks dan memasukkan dokumen. Urutan hasilnya tidak spesifik dan dapat bervariasi untuk setiap`SELECT` kueri. Anda tidak boleh bergantung pada urutan hasil untuk kueri apa pun di 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**

Dalam PartiQL, Anda menggunakan tanda kutip tunggal untuk menunjukkan string dalam bahasa manipulasi data (DHTML) atau pernyataan query. Tetapi konsol QLDB dan shell QLDB mengembalikan kueri menghasilkan format teks Amazon Ion, sehingga Anda melihat string tertutup dalam tanda kutip ganda.
Sintaks ini memungkinkan bahasa kueri PartiQL untuk mempertahankan kompatibilitas SQL, dan format teks Amazon Ion untuk mempertahankan kompatibilitas JSON.

## Proyeksi dan filter

Anda dapat melakukan proyeksi (ditargetkan`SELECT`) dan filter standar lainnya (`WHERE`klausa). Query berikut mengembalikan subset bidang dokumen dari`VehicleRegistration` tabel. Filter untuk kendaraan dengan kriteria berikut:

- String filter - Ini terdaftar di Seattle.
- Filter desimal - Ini memiliki jumlah tiket penalti yang tertunda kurang dari`100.0`.
- Filter tanggal - Ini memiliki tanggal pendaftaran yang berlaku pada atau setelah 4 September 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" }]
    }
}
```

## Bergabung

Anda juga dapat menulis kueri gabungan batin. Contoh berikut menunjukkan batin implisit bergabung query yang mengembalikan semua dokumen pendaftaran bersama dengan atribut kendaraan terdaftar.

```
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"
}
```

Atau, Anda dapat menulis kueri gabungan batin yang sama dalam sintaks eksplisit sebagai berikut.

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

## Data yang tertumpuk

Anda dapat menggunakan PartiQL di QLDB untuk query data bersarang dalam dokumen. Contoh berikut menunjukkan subquery berkorelasi yang meratakan data bersarang. @Karakter secara teknis opsional di sini. Tetapi secara eksplisit menunjukkan bahwa Anda ingin`Owners` struktur dalam`VehicleRegistration`, bukan koleksi yang berbeda bernama`Owners` (jika ada).

```
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: []
}
```

Berikut ini menunjukkan subquery dalam`SELECT` daftar yang memproyeksikan data bersarang, di tambahan untuk bergabung batin.

```
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"]
}
```

Query berikut mengembalikan`PersonId` dan indeks (ordinal) jumlah setiap orang dalam`Owners.SecondaryOwners` daftar untuk`VehicleRegistration` dokumen.

```
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
}
```

Untuk mempelajari cara menanyakan metadata dokumen Anda, lanjutkan keMelakukan Kueri Metadata Dokumen.

# Melakukan Kueri Metadata Dokumen

`INSERT`Pernyataan menciptakan revisi awal dokumen dengan nomor versi nol. Untuk mengidentifikasi setiap dokumen secara unik, Amazon QLDB menetapkan ID dokumen sebagai bagian dari metadata.

Selain ID dokumen dan nomor versi, QLDB menyimpan metadata yang dihasilkan sistem lainnya untuk setiap dokumen dalam tabel. Metadata ini mencakup informasi transaksi, atribut jurnal, dan nilai hash dokumen.

Semua ID yang ditetapkan sistem adalah pengidentifikasi unik universal (UUID) yang masing-masing diwakili dalam string yang dikodekan Base62. Untuk informasi selengkapnya, lihat ID unik di Amazon QLDB.

Topik

- Tampilan berkomitmen
- Bergabung dengan tampilan berkomitmen dan pengguna

## Tampilan berkomitmen

Anda dapat mengakses metadata dokumen dengan menanyakan tampilan berkomitmen. Pandangan ini mengembalikan dokumen dari tabel yang didefinisikan sistem yang secara langsung sesuai dengan tabel pengguna Anda. Ini mencakup revisi terbaru yang berkomitmen dan tidak dihapus dari data Anda dan metadata yang dihasilkan sistem. Untuk menanyakan tampilan ini, tambahkan awalan`_ql_committed_` ke nama tabel dalam kueri Anda. (Awalan`_ql_` dicadangkan di QLDB untuk objek sistem.)

```
SELECT * FROM _ql_committed_VehicleRegistration AS r
WHERE r.data.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Dengan menggunakan data yang sebelumnya dimasukkanMembuat tabel dengan indeks dan memasukkan dokumen, output dari kueri ini menunjukkan isi sistem dari setiap revisi terbaru dokumen yang tidak dihapus. Dokumen sistem memiliki metadata yang bersarang di`metadata` bidang, dan data pengguna Anda bersarang di`data` bidang.

```
{
    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"
    }
}
```

Bidang tampilan berkomitmen

- `blockAddress`- Lokasi blok di jurnal buku besar Anda di mana revisi dokumen dilakukan. Alamat, yang dapat digunakan untuk verifikasi kriptografi, memiliki dua bidang berikut.
  - `strandId`- ID unik untai jurnal yang berisi blok.
  - `sequenceNo`- Nomor indeks yang menentukan lokasi blok di dalam untai.

  > **Note**
  >
  > Kedua dokumen dalam contoh ini memiliki identik`blockAddress` dengan yang sama`sequenceNo`. Karena dokumen-dokumen ini dimasukkan dalam satu transaksi (dan dalam hal ini, dalam satu pernyataan), mereka berkomitmen di blok yang sama.

- `hash`- Nilai hash SHA-256 Ion yang secara unik mewakili revisi dokumen. Hash mencakup revisi`data` dan`metadata` bidang dan dapat digunakan untuk verifikasi kriptografi.
- `data`- Atribut data pengguna dokumen.

  Jika Anda menyunting revisi,`data` struktur ini digantikan oleh`dataHash` bidang, yang nilainya adalah hash Ion dari`data` struktur yang dihapus.
- `metadata`- Atribut metadata dokumen.
  - `id`- ID unik yang ditugaskan sistem dari dokumen.
  - `version`— Nomor versi dokumen. Ini adalah integer berbasis nol yang bertambah dengan setiap revisi dokumen.

- `txTime`- Stempel waktu saat revisi dokumen berkomitmen pada jurnal.
- `txId`— ID unik dari transaksi yang melakukan revisi dokumen.

## Bergabung dengan tampilan berkomitmen dan pengguna

Anda dapat menulis kueri yang bergabung dengan tabel dalam tampilan berkomitmen dengan tabel di tampilan pengguna. Misalnya, Anda mungkin ingin bergabung dengan dokumen`id` satu tabel dengan bidang yang ditentukan pengguna dari tabel lain.

Query berikut bergabung dua tabel bernama`DriversLicense` dan`Person` pada`id` bidang`PersonId` dan dokumen mereka masing-masing, menggunakan pandangan berkomitmen untuk yang terakhir.

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS p
ON d.PersonId = p.metadata.id
WHERE p.metadata.id = '1CWScY2qHYI9G88C2SjvtH'
```

Untuk mempelajari cara menanyakan bidang ID dokumen dalam tampilan pengguna default, lanjutkan keMenggunakan klausa BY untuk query ID dokumen.

# Menggunakan klausa BY untuk query ID dokumen

Meskipun Anda dapat menentukan bidang yang dimaksudkan untuk menjadi pengidentifikasi unik (misalnya, VIN kendaraan), pengenal unik dokumen yang sebenarnya adalah bidang`id` metadata, seperti yang dijelaskan dalamMemasukkan dokumen. Untuk alasan ini, Anda dapat menggunakan`id` bidang untuk membuat hubungan antara tabel.

`id`Bidang dokumen langsung dapat diakses dalam tampilan berkomitmen saja, tetapi Anda juga dapat memproyeksikannya dalam tampilan pengguna default dengan menggunakanBY klausa. Sebagai contoh, lihat query berikut dan hasilnya.

```
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" }]
    }
}
```

Dalam query ini,`r_id` adalah alias user-defined yang dinyatakan dalam`FROM` klausa, menggunakan`BY` kata kunci. `r_id`Alias ini mengikat ke bidang`id` metadata untuk setiap dokumen dalam set hasil kueri. Anda dapat menggunakan alias ini dalam`SELECT` klausa dan juga dalam`WHERE` klausa query dalam tampilan pengguna.

Namun, untuk mengakses atribut metadata lainnya, Anda harus menanyakan tampilan berkomitmen.

## Bergabung di ID dokumen

Misalkan Anda menggunakan dokumen`id` satu tabel sebagai kunci asing di bidang yang ditentukan pengguna dari tabel lain. Anda dapat menggunakan`BY` klausa untuk menulis kueri gabungan batin untuk dua tabel di bidang ini (miripBergabung dengan tampilan berkomitmen dan pengguna dengan topik sebelumnya).

Contoh berikut bergabung dua tabel bernama`DriversLicense` dan`Person` pada`id` bidang`PersonId` dan dokumen mereka masing-masing, menggunakan`BY` klausa untuk yang terakhir.

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
ON d.PersonId = pid
WHERE pid = '1CWScY2qHYI9G88C2SjvtH'
```

Untuk mempelajari cara membuat perubahan pada dokumen di tabel Anda, lanjutkan keMemperbarui dan menghapus dokumen.

# Memperbarui dan menghapus dokumen

Di Amazon QLDB, revisi dokumen adalah struktur Amazon Ion yang mewakili satu versi dari urutan dokumen yang diidentifikasi oleh ID dokumen unik. Setiap revisi berisi kumpulan data lengkap

dokumen, termasuk data pengguna dan metadata yang dihasilkan sistem Anda. Setiap revisi diidentifikasi secara unik dengan kombinasi ID dokumen dan nomor versi berbasis nol.

Ketika Anda memperbarui dokumen, QLDB membuat revisi baru dengan ID dokumen yang sama dan nomor versi bertambah. Siklus hidup dokumen berakhir saat Anda menghapusnya dari tabel. Ini berarti bahwa tidak ada revisi dokumen dengan ID dokumen yang sama dapat dibuat lagi.

## Membuat revisi dokumen

Misalnya, pernyataan berikut memasukkan registrasi kendaraan baru, memperbarui kota pendaftaran, dan kemudian menghapus pendaftaran. Ini menghasilkan tiga revisi dokumen.

```
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**
>
> Menyisipkan pernyataan dan pernyataan DML-lainnya mengembalikan ID dari setiap dokumen yang terkena dampak. Sebelum melanjutkan, simpan ID ini karena Anda memerlukannya untuk fungsi riwayat di topik berikutnya. Anda juga dapat menemukan ID dokumen dengan kueri berikut.
>
> ```
> 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'
```

Untuk contoh dan informasi selengkapnya tentang sintaks pernyataan DML-ini, lihatPERBARUI danHAPUS dalam referensi PartiQL Amazon QLDB.

Untuk menyisipkan dan menghapus elemen tertentu dalam dokumen, Anda dapat menggunakanUPDATE pernyataan atau pernyataan DML-lain yang dimulai denganFROM kata kunci. Untuk informasi dan contoh, lihatDARI (INSERT, HAPUS, atau SET) referensi.

Setelah Anda menghapus dokumen, Anda tidak dapat lagi menanyainya di tampilan berkomitmen atau pengguna. Untuk mempelajari cara menanyakan riwayat revisi dokumen ini menggunakan fungsi riwayat bawaan, lanjutkan keMelakukan Kueri Riwayat Riwayat Revisi.

# Melakukan Kueri Riwayat Riwayat Revisi

Amazon QLDB menyimpan riwayat lengkap setiap dokumen dalam tabel. Anda dapat melihat ketiga revisi dokumen registrasi kendaraan yang sebelumnya Anda masukkan, diperbarui, dan dihapusMemperbarui dan menghapus dokumen dengan menanyakan fungsi riwayat bawaan.

Topik

- Fungsi Riwayat
- Contoh kueri Riwayat

## Fungsi Riwayat

Fungsi sejarah di QLDB adalah ekstensi PartiQL yang mengembalikan revisi dari tampilan yang ditentukan sistem tabel Anda. Jadi, ini mencakup data Anda dan metadata terkait dalam skema yang sama dengan tampilan berkomitmen.

Sintaks

```
SELECT * FROM history( table_name | 'table_id' [, `start-time` [, `end-time` ] ] ) AS h
[ WHERE h.metadata.id = 'id' ]
```

Pendapat

***table_name*** | '***table_id***'

Entah nama tabel atau ID tabel. Nama tabel adalah identifier PartiQL yang dapat Anda tunjukkan dengan tanda kutip ganda atau tidak ada tanda kutip. Sebuah ID tabel adalah string literal yang harus tertutup dalam tanda kutip tunggal. Untuk mempelajari selengkapnya menggunakan ID tabel, lihatMenanyakan sejarah tabel tidak aktif.

`` `***start-time***`,`***akhir waktu***` ``

(Opsional) Menentukan rentang waktu di mana setiap revisi aktif. Parameter ini tidak menentukan rentang waktu selama revisi berkomitmen pada jurnal dalam transaksi.

Waktu mulai dan akhir adalah literal timestamp Ion yang dapat dilambangkan dengan backticks (`` `...` ``). Untuk mempelajari selengkapnya, lihat Menanyakan Ion dengan PartiQL di Amazon QLDB.

Parameter waktu ini memiliki perilaku berikut:

- Start-time dan end-time keduanya inklusif. Mereka harus berada dalam format tanggal dan waktu ISO 8601 dan dalam Waktu Universal Terkoordinasi (UTC).
- Waktu mulai harus kurang dari atau sama dengan waktu akhir dan dapat berupa tanggal sewenang-wenang di masa lalu.
- Waktu akhir harus kurang atau sama dengan tanggal dan waktu UTC saat ini.
- Jika Anda menentukan waktu mulai tetapi bukan waktu akhir, kueri Anda akan menetapkan waktu akhir ke tanggal dan waktu saat ini. Jika Anda tidak menentukan keduanya, kueri Anda akan mengembalikan keseluruhan riwayat.

'***id***'

(Opsional) ID dokumen yang ingin Anda kueri pada riwayat revisi, dilambangkan dengan tanda kutip tunggal.

> **Tip**
>
> Sebagai praktik terbaik, kualifikasi kueri riwayat dengan rentang tanggal (start-time dan end-time) dan ID dokumen (`metadata.id`). Dalam QLDB, setiap`SELECT` permintaan diproses dalam transaksi dan tunduk pada batas batas waktu transaksi.
> Kueri sejarah tidak menggunakan indeks yang Anda buat di atas meja. Riwayat QLDB diindeks oleh ID dokumen saja, dan Anda tidak dapat membuat indeks riwayat tambahan

saat ini. Kueri riwayat yang menyertakan waktu mulai dan waktu akhir mendapatkan manfaat kualifikasi rentang tanggal.

## Contoh kueri Riwayat

Untuk menanyakan riwayat dokumen registrasi kendaraan, gunakan dokumen`id` yang sebelumnya Anda simpan[Memperbarui dan menghapus dokumen](). Misalnya, kueri riwayat berikut menampilkan revisi apa pun untuk ID dokumen`ADR2Ll1fGsU4Jr4EqTdnQF` yang pernah aktif di antara`2019-06-05T00:00:00Z` dan`2019-06-05T23:59:59Z`.

> **Note**
>
> Ingat bahwa parameter waktu mulai dan akhir tidak menentukan rentang waktu ketika revisi berkomitmen ke jurnal dalam transaksi. Misalnya, jika revisi dilakukan sebelumnya`2019-06-05T00:00:00Z` dan tetap aktif melewati waktu mulai itu, kueri contoh ini akan mengembalikan revisi tersebut dalam hasil.

Pastikan untuk mengganti`id`, waktu mulai, dan waktu akhir dengan nilai Anda sendiri.

```
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
```

Hasil kueri Anda akan terlihat serupa dengan yang berikut ini:

```
{
    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"
    }
}
```

Output mencakup atribut metadata yang memberikan detail tentang kapan setiap item dimodifikasi, dan transaksi mana. Dari data ini, Anda bisa melihat yang berikut ini:

- Dokumen ini diidentifikasi secara unik oleh sistem-ditugaskan`id`:`ADR2Ll1fGsU4Jr4EqTdnQF`. Ini adalah UUID yang diwakili dalam string Base62 dikodekan.
- `INSERT`Pernyataan menciptakan revisi awal dokumen (versi`0`).
- Setiap pembaruan berikutnya membuat revisi baru dengan dokumen yang sama`id` dan nomor versi yang bertambah.
- `txId`Bidang menunjukkan transaksi yang dilakukan setiap revisi, dan`txTime` menunjukkan kapan masing-masing berkomitmen.
- Sebuah`DELETE` pernyataan menciptakan revisi baru namun akhir dari sebuah dokumen. Revisi akhir ini hanya memiliki metadata.

Untuk mempelajari cara menghapus revisi secara permanen, lanjutkan keMenyunting revisi dokumen.

## Menyunting revisi dokumen

Di Amazon QLDB,`DELETE` pernyataan hanya secara logis menghapus dokumen dengan membuat revisi baru yang menandainya sebagai dihapus. QLDB juga mendukung operasi redaksi data yang memungkinkan Anda menghapus revisi dokumen yang tidak aktif secara permanen dalam sejarah tabel.

> **Note**
>
> Buku besar apa pun yang dibuat sebelum 22 Juli 2021 saat ini tidak memenuhi syarat untuk redaksi. Anda dapat melihat waktu pembuatan buku besar Anda di konsol Amazon QLDB.

Operasi redaksi hanya menghapus data pengguna dalam revisi yang ditentukan dan membiarkan urutan jurnal dan metadata dokumen tidak berubah. Ini menjaga integritas data keseluruhan buku besar Anda.

Sebelum memulai redaksi data di QLDB, pastikan Anda meninjauPertimbangan dan batasan redaksi di referensi Amazon QLDB PartiQL.

Topik

- Prosedur redaksi disimpan
- Memeriksa apakah redaksi selesai
- Contoh redaksi
- Menghapus dan menyunting revisi aktif
- Menyunting bidang tertentu dalam revisi

## Prosedur redaksi disimpan

Anda dapat menggunakan prosedur yangREDACT_REVISI disimpan untuk menghapus revisi individu yang tidak aktif secara permanen dalam buku besar. Prosedur tersimpan ini menghapus semua data pengguna dalam revisi yang ditentukan di penyimpanan terindeks dan penyimpanan jurnal. Namun, ia meninggalkan urutan jurnal dan metadata dokumen, termasuk ID dokumen dan hash, tidak berubah. Operasi ini tidak dapat diubah.

Revisi dokumen yang ditentukan harus merupakan revisi tidak aktif dalam sejarah. Revisi aktif terbaru dari dokumen tidak memenuhi syarat untuk redaksi.

Untuk menyunting beberapa revisi, Anda harus menjalankan prosedur yang disimpan satu kali untuk setiap revisi. Anda dapat menyunting satu revisi per transaksi.

Sintaks

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

Pendapat

*`blok-alamat`*

Lokasi blok jurnal revisi dokumen yang akan disunting. Alamat adalah struktur Amazon Ion yang memiliki dua bidang:`strandId` dan`sequenceNo`.

Ini adalah nilai literal Ion yang dilambangkan dengan backticks. Misalnya:

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

'*tabel-id*'

ID unik dari tabel yang revisi dokumennya ingin Anda edit, dilambangkan dengan tanda kutip tunggal.

'*dokumen-id*'

ID dokumen unik dari revisi yang akan disunting, dilambangkan dengan tanda kutip tunggal.

## Memeriksa apakah redaksi selesai

Saat Anda mengirimkan permintaan redaksi dengan menjalankan prosedur yang disimpan, QLDB memproses redaksi data secara asinkron. Setelah selesai, data pengguna dalam revisi (diwakili oleh`data` struktur) dihapus secara permanen. Untuk memeriksa apakah permintaan redaksi telah selesai, Anda dapat menggunakan salah satu dari berikut ini:

- Ekspor jurnal
- Aliran jurnal
- GetBlock Operasi API
- GetRevision Operasi API
- Fungsi Riwayat- Catatan: Setelah redaksi selesai di jurnal, diperlukan beberapa waktu sebelum kueri sejarah menunjukkan hasil redaksi. Anda mungkin melihat beberapa revisi disunting sebelum yang lain karena redaksi asinkron selesai, tetapi kueri riwayat akan menampilkan hasil yang diselesaikan pada akhirnya.

Setelah redaksi revisi selesai,`data` struktur revisi digantikan oleh`dataHash` bidang baru. Nilai bidang ini adalah hash Ion dari`data` struktur yang dihapus, seperti yang ditunjukkan pada contoh berikut. Akibatnya, buku besar mempertahankan integritas data secara keseluruhan dan tetap dapat diverifikasi secara kriptografis melalui operasi API verifikasi yang ada. Untuk mempelajari selengkapnya tentang verifikasi, lihatVerifikasi data di Amazon QLDB.

## Contoh redaksi

Pertimbangkan dokumen registrasi kendaraan yang sebelumnya Anda ulasMelakukan Kueri Riwayat Riwayat Revisi. Misalkan Anda ingin menyunting revisi kedua (`version:1`). Contoh query berikut menunjukkan revisi ini sebelum redaksi. Dalam hasil kueri,`data` struktur yang akan diedit disorot dalam *huruf miring merah*.

```
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"
    }
}
```

Catatan`blockAddress` dalam hasil query karena Anda harus lulus nilai ini untuk prosedur yang`REDACT_REVISION` disimpan. Kemudian, temukan ID unik`VehicleRegistration` tabel dengan menanyakan katalog sistem, sebagai berikut.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Gunakan ID tabel ini bersama dengan ID dokumen dan alamat blok untuk dijalankan`REDACT_REVISION`. ID tabel dan ID dokumen adalah literal string yang harus dilampirkan dalam tanda kutip tunggal, dan alamat blok adalah literal Ion yang tertutup dalam backticks. Pastikan untuk mengganti argumen ini dengan nilai-nilai Anda sendiri yang sesuai.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

> **Tip**
>
> Bila Anda menggunakan konsol QLDB atau shell QLDB untuk query untuk ID tabel atau ID dokumen (atau string nilai literal), nilai yang dikembalikan tertutup dalam tanda kutip ganda. Namun, ketika Anda menentukan ID tabel dan ID dokumen argumen dari prosedur yang`REDACT_REVISION` disimpan, Anda harus melampirkan nilai-nilai dalam tanda kutip tunggal.
>
> Ini karena Anda menulis pernyataan dalam format PartiQL, tetapi QLDB mengembalikan hasil dalam format Amazon Ion. Untuk rincian tentang sintaks dan semantik PartiQL di QLDB, lihatKueri Ion dengan PartiQL.

Permintaan redaksi yang valid mengembalikan struktur Ion yang mewakili revisi dokumen yang Anda edit, sebagai berikut.

```
{
  blockAddress: {
    strandId: "JdxjkR9bSYB5jMHWcI464T",
    sequenceNo: 17
  },
  tableId: "5PLf9SXwndd63lPaSIa0O6",
  documentId: "ADR2Ll1fGsU4Jr4EqTdnQF",
  version: 1
}
```

Saat Anda menjalankan prosedur tersimpan ini, QLDB memproses permintaan redaksi Anda secara asinkron. Setelah menyelesaikan redaksi,`data` struktur dihapus secara permanen dan diganti

dengan *dataHash*bidang baru. Nilai bidang ini adalah hash Ion dari`data` struktur yang dihapus, sebagai berikut.

> **Note**
>
> `dataHash`Contoh ini disediakan untuk tujuan informasi saja dan bukan nilai hash yang dihitung nyata.

```
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:17
    },
    hash:{{LGSFZ4iEYWZeMwmAqcxxNyT4wbCtuMOmFCj8pEd6Mp0=}},
    dataHash: {{s83jd7sfhsdfhksj7hskjdfjfpIPP/DP2hvionas2d4=}},
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:1,
        txTime:2019-06-05T21:01:442d-3Z,
        txId:"9cArhIQV5xf5Tf5vtsPwPq"
    }
}
```

## Menghapus dan menyunting revisi aktif

Revisi dokumen aktif (yaitu, revisi terbaru yang tidak dihapus dari setiap dokumen) tidak memenuhi syarat untuk redaksi data. Sebelum Anda dapat menyunting revisi aktif, Anda harus terlebih dahulu memperbarui atau menghapusnya. Ini memindahkan revisi yang sebelumnya aktif ke riwayat dan membuatnya memenuhi syarat untuk redaksi.

Jika kasus penggunaan Anda mengharuskan seluruh dokumen ditandai sebagai dihapus, Anda pertama kali menggunakan pernyataan DELETE. Sebagai contoh, pernyataan berikut logis menghapus`VehicleRegistration` dokumen dengan VIN dari`1HVBBAANXWH544237`.

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

Kemudian, sunting revisi sebelumnya sebelum penghapusan ini, seperti yang dijelaskan sebelumnya. Jika diperlukan, Anda juga dapat menyunting revisi sebelumnya secara individual.

Jika kasus penggunaan Anda mengharuskan dokumen untuk tetap aktif, pertama-tama Anda menggunakan pernyataan UPDATE atau FROM untuk mengaburkan atau menghapus bidang yang ingin Anda edit. Proses ini dijelaskan di bagian berikut.

### Menyunting bidang tertentu dalam revisi

QLDB tidak mendukung redaksi bidang tertentu dalam revisi dokumen. Untuk melakukannya, pertama-tama Anda dapat menggunakan pernyataan UPDATE-REMOVE atau FROM-REMOVE untuk menghapus bidang yang ada dari revisi. Misalnya, pernyataan berikut menghapus`LicensePlateNumber` bidang dari`VehicleRegistration` dokumen dengan VIN dari`1HVBBAANXWH544237`.

```
UPDATE VehicleRegistration AS r
REMOVE r.LicensePlateNumber
WHERE r.VIN = '1HVBBAANXWH544237'
```

Kemudian, menyunting revisi sebelumnya sebelum penghapusan ini, seperti yang dijelaskan sebelumnya. Jika diperlukan, Anda juga dapat secara individual menyunting revisi sebelumnya yang menyertakan bidang yang sekarang dihapus ini.

Untuk mempelajari cara mengoptimalkan kueri Anda, lanjutkan keMengoptimalkan kinerja kueri.

## Mengoptimalkan kinerja kueri

Amazon QLDB dimaksudkan untuk memenuhi kebutuhan beban kerja pemrosesan transaksi online (OLTP). Ini berarti bahwa QLDB dioptimalkan untuk serangkaian pola kueri tertentu, meskipun mendukung kemampuan query seperti SQL. Sangat penting untuk merancang aplikasi dan model data mereka untuk bekerja dengan pola kueri ini. Jika tidak, seiring pertumbuhan tabel Anda, Anda akan mengalami masalah kinerja yang signifikan, termasuk latensi kueri, batas waktu transaksi, dan konflik konkurensi.

Bagian ini menjelaskan kendala query di QLDB dan memberikan panduan untuk menulis query optimal mengingat kendala ini.

Topik

- Batas waktu habis transaksi
- Konkurensi Konkurensi
- Pola kueri optimal

- Pola kueri yang harus dihindari
- Memantau kinerja

## Batas waktu habis transaksi

Dalam QLDB, setiap pernyataan PartiQL (termasuk setiap`SELECT` permintaan) diproses dalam transaksi dan tunduk pada batas batas waktu transaksi. Transaksi dapat berjalan hingga 30 detik sebelum dilakukan. Setelah batas ini, QLDB menolak setiap pekerjaan yang dilakukan pada transaksi dan membuang sesi yang menjalankan transaksi. Batas ini melindungi klien layanan dari sesi bocor dengan memulai transaksi dan tidak melakukan atau membatalkannya.

## Konkurensi Konkurensi

QLDB mengimplementasikan kontrol konkurensi dengan menggunakan kontrol konkurensi optimis (OCC). Kueri suboptimal juga dapat menyebabkan lebih banyak konflik OCC. Untuk informasi tentang OCC, lihatModel Konkurensi Amazon QLDB.

## Pola kueri optimal

Sebagai praktik terbaik, Anda harus menjalankan pernyataan dengan klausa`WHERE` predikat yang menyaring pada bidang yang diindeks atau ID dokumen. QLDB membutuhkan operator kesetaraan (`=`atau`IN`) pada bidang diindeks untuk secara efisien mencari dokumen.

Berikut ini adalah contoh pola query yang optimal dalam tampilan pengguna.

```
--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'
```

Setiap kueri yang tidak mengikuti pola-pola ini memanggil pemindaian tabel penuh. Pemindaian tabel dapat menyebabkan batas waktu transaksi untuk kueri pada tabel besar atau kueri yang mengembalikan set hasil yang besar. Mereka juga dapat menyebabkan konflik OCC dengan transaksi yang bersaing.

Indeks kardinalitas tinggi

Kami merekomendasikan bidang pengindeksan yang berisi nilai kardinalitas tinggi. Misalnya,`VIN` dan`LicensePlateNumber` bidang dalam`VehicleRegistration` tabel diindeks bidang yang dimaksudkan untuk menjadi unik.

Hindari mengindeks bidang kardinalitas rendah seperti kode status, negara alamat atau provinsi, dan kode pos. Jika Anda mengindeks bidang seperti itu, kueri Anda dapat menghasilkan kumpulan hasil besar yang lebih mungkin menghasilkan batas waktu transaksi atau menyebabkan konflik OCC yang tidak diinginkan.

Kueri tampilan berkomitmen

Kueri yang Anda jalankan dalam tampilan berkomitmen mengikuti pedoman pengoptimalan yang sama dengan kueri tampilan pengguna. Indeks yang Anda buat di atas meja juga digunakan untuk kueri dalam tampilan berkomitmen.

Kueri fungsi sejarah

Kueri fungsi sejarah tidak menggunakan indeks yang Anda buat di atas meja. Riwayat QLDB diindeks oleh ID dokumen saja, dan Anda tidak dapat membuat indeks riwayat tambahan saat ini.

Sebagai praktik terbaik, kualifikasi kueri riwayat dengan rentang tanggal (waktu mulai dan waktu akhir) dan ID dokumen (`metadata.id`). Kueri riwayat yang menyertakan waktu mulai dan waktu akhir mendapatkan manfaat kualifikasi rentang tanggal.

Kueri gabungan batin

Untuk kueri gabungan dalam, gunakan kriteria gabungan yang mencakup setidaknya bidang yang diindeks untuk tabel di sisi kanan gabungan. Tanpa indeks gabungan, kueri gabungan memanggil beberapa pemindaian tabel—untuk setiap dokumen di tabel kiri gabungan, kueri sepenuhnya memindai tabel yang tepat. Praktik terbaik adalah untuk bergabung pada bidang yang diindeks untuk setiap tabel yang Anda bergabung, selain menentukan predikat`WHERE` kesetaraan untuk setidaknya satu tabel.

Misalnya, query berikut bergabung`VehicleRegistration` dan`Vehicle` tabel pada`VIN` bidang masing-masing, yang keduanya diindeks. Query ini juga memiliki predikat kesetaraan pada`VehicleRegistration.VIN`.

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

Pilih indeks kardinalitas tinggi untuk kriteria gabungan dan predikat kesetaraan dalam kueri gabungan Anda.

## Pola kueri yang harus dihindari

Berikut ini adalah beberapa contoh pernyataan suboptimal yang tidak skala baik untuk tabel yang lebih besar di QLDB. Kami sangat menyarankan agar Anda tidak bergantung pada jenis kueri ini untuk tabel yang tumbuh seiring waktu karena kueri Anda pada akhirnya akan menghasilkan batas waktu transaksi. Karena tabel berisi dokumen yang ukurannya bervariasi, sulit untuk menentukan batas yang tepat untuk kueri yang tidak diindeks.

```
--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)
```

Secara umum, kami tidak menyarankan untuk menjalankan jenis pola kueri berikut untuk kasus penggunaan produksi di QLDB:

- Kueri pemrosesan analitis online (OLAP)
- Kueri eksplorasi tanpa klausa predikat
- Kueri pelaporan
- Pencarian teks

Sebagai gantinya, kami merekomendasikan streaming data Anda ke layanan database yang dibuat khusus yang dioptimalkan untuk kasus penggunaan analitis. Misalnya, Anda dapat melakukan streaming data QLDB ke Amazon OpenSearch Service untuk memberikan kemampuan pencarian teks lengkap melalui dokumen. Untuk contoh aplikasi yang menunjukkan kasus penggunaan ini, lihat GitHub repositori aws-samples/amazon-qldb-streaming-amazon -opensearch-service-sample-python. Untuk informasi tentang aliran QLDB, lihatStreaming data jurnal dari Amazon QLDB.

### Memantau kinerja

Driver QLDB menyediakan dikonsumsi I/O penggunaan dan waktu informasi dalam objek hasil pernyataan. Anda dapat menggunakan metrik ini untuk mengidentifikasi pernyataan PartiQL yang tidak efisien. Untuk mempelajari lebih lanjut, lanjutkan keMendapatkan statistik pernyataan PartiQL.

Anda juga dapat menggunakan Amazon CloudWatch untuk melacak kinerja buku besar Anda untuk operasi data. Pantau`CommandLatency` metrik untuk yang ditentukan`LedgerName` dan`CommandType`. Untuk informasi selengkapnya, lihat Pemantauan CloudWatch dengan Amazon. Untuk mempelajari bagaimana QLDB menggunakan perintah untuk mengelola operasi data, lihatManajemen sesi dengan pengemudi.

## Mendapatkan statistik pernyataan PartiQL

Amazon QLDB menyediakan statistik eksekusi pernyataan yang dapat membantu Anda mengoptimalkan penggunaan QLDB dengan menjalankan pernyataan PartiQL yang lebih efisien. QLDB mengembalikan statistik ini bersama dengan hasil pernyataan. Ini mencakup metrik yang mengukur penggunaan I/O yang dikonsumsi dan waktu pemrosesan sisi server, yang dapat Anda gunakan untuk mengidentifikasi pernyataan yang tidak efisien.

Fitur ini saat ini tersedia di editor PartiQL pada konsol QLDB, shell QLDB, dan versi terbaru dari driver QLDB untuk semua bahasa yang didukung. Anda juga dapat melihat statistik pernyataan untuk riwayat kueri Anda di konsol.

Topik

- Penggunaan I/O
- Informasi waktu

## Penggunaan I/O

Metrik penggunaan I/O menjelaskan jumlah permintaan I/O yang dibaca. Jika jumlah permintaan I/O baca lebih tinggi dari yang diharapkan, ini menunjukkan bahwa pernyataan tersebut tidak dioptimalkan, seperti kurangnya indeks. Kami menyarankan Anda meninjauPola kueri optimal topik sebelumnya, Mengoptimalkan kinerja kueri.

> **Note**
>
> Saat Anda menjalankan`CREATE INDEX` pernyataan pada tabel yang tidak kosong, metrik penggunaan I/O hanya menyertakan permintaan baca untuk panggilan pembuatan indeks sinkron.
> QLDB membangun indeks untuk setiap dokumen yang ada dalam tabel asynchronously. Permintaan baca asinkron ini tidak disertakan dalam metrik penggunaan I/O dari hasil pernyataan Anda. Permintaan baca asinkron dibebankan secara terpisah dan ditambahkan ke I/Os baca total Anda setelah build indeks selesai.

### Menggunakan konsol QLDB

Untuk mendapatkan penggunaan I/O baca pernyataan dengan menggunakan konsol QLDB, lakukan langkah-langkah berikut:

1. Buka konsol Amazon QLDB di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih editor PartiQL.
3. Pilih buku besar dari daftar dropdown buku besar.
4. Di jendela editor kueri, masukkan pernyataan pilihan Anda, lalu pilih Jalankan. Berikut ini adalah contoh query.

```
SELECT * FROM testTable WHERE firstName = 'Jim'
```

Untuk menjalankan pernyataan, Anda juga dapat menggunakan pintasan keyboardCtrl +Enter untuk Windows, atauCmd +Return untuk macOS. Untuk pintasan keyboard lainnya, lihatPintasan keyboard editor PartiQL.

5. Di bawah jendela editor kueri, hasil kueri Anda termasuk membaca I/Os, yang merupakan jumlah permintaan baca yang dibuat oleh pernyataan.

Anda juga dapat melihat I/O baca riwayat kueri Anda dengan melakukan langkah-langkah berikut:

1. Di panel navigasi, pilih Kueri terbaru di bawah editor PartiQL.
2. Kolom Baca I/Os menampilkan jumlah permintaan baca yang dibuat oleh setiap pernyataan.

## Menggunakan driver QLDB

Untuk mendapatkan penggunaan I/O pernyataan dengan menggunakan driver QLDB, memanggil`getConsumedIOs` operasi kursor aliran hasil atau kursor buffered.

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan membaca I/Os dari kursor aliran hasil pernyataan.

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**
>
> Untuk mengkonversi ke kode sinkron, menghapus`await` dan`async` kata kunci, dan mengubah`IAsyncResult` jenis untuk`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))
```

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan membaca I/Os dari kursor buffered dari hasil pernyataan. Ini mengembalikan total membaca I/Os dari`ExecuteStatement` dan`FetchPage` permintaan.

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**
>
> Untuk mengkonversi ke kode sinkron, menghapus`await` dan`async` kata kunci, dan mengubah`IAsyncResult` jenis untuk`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**
>
> Kursor aliran stateful karena paginasi hasil set. Oleh karena itu,`getConsumedIOs` dan`getTimingInformation` operasi mengembalikan akumulasi metrik dari waktu yang Anda panggil mereka.
> Kursor buffer buffer hasil diatur dalam memori dan mengembalikan total akumulasi metrik.

## Informasi waktu

Metrik informasi pengaturan waktu menjelaskan waktu pemrosesan sisi server dalam milidetik. Waktu pemrosesan sisi server didefinisikan sebagai jumlah waktu yang dihabiskan QLDB untuk memproses

pernyataan. Ini tidak termasuk waktu yang dihabiskan untuk panggilan jaringan atau jeda. Metrik ini membedakan waktu pemrosesan di sisi layanan QLDB dari waktu pemrosesan di sisi klien.

### Menggunakan konsol QLDB

Untuk mendapatkan informasi pengaturan waktu pernyataan dengan menggunakan konsol QLDB, lakukan langkah-langkah berikut:

1. Buka konsol Amazon QLDB di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih editor PartiQL.
3. Pilih buku besar dari daftar dropdown buku besar.
4. Di jendela editor kueri, masukkan pernyataan pilihan Anda, lalu pilih Jalankan. Berikut ini adalah contoh query.

   ```
   SELECT * FROM testTable WHERE firstName = 'Jim'
   ```

   Untuk menjalankan pernyataan, Anda juga dapat menggunakan pintasan keyboardCtrl +Enter untuk Windows, atauCmd +Return untuk macOS. Untuk pintasan keyboard lainnya, lihatPintasan keyboard editor PartiQL.
5. Di bawah jendela editor kueri, hasil kueri Anda mencakup latensi sisi server, yang merupakan jumlah waktu antara saat QLDB menerima permintaan pernyataan, dan ketika mengirim respons. Ini adalah bagian dari total durasi kueri.

Anda juga dapat melihat informasi waktu riwayat kueri Anda dengan melakukan langkah-langkah berikut:

1. Di panel navigasi, pilih Kueri terbaru di bawah editor PartiQL.
2. Waktu Eksekusi (ms) kolom menampilkan informasi waktu ini untuk setiap pernyataan.

### Menggunakan driver QLDB

Untuk mendapatkan informasi waktu pernyataan dengan menggunakan driver QLDB, panggil`getTimingInformation` operasi kursor aliran hasil atau kursor buffer.

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan waktu pemrosesan dari kursor aliran hasil pernyataan.

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**
>
> Untuk mengkonversi ke kode sinkron, menghapus`await` dan`async` kata kunci, dan mengubah`IAsyncResult` jenis untuk`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))
```

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan waktu pemrosesan dari kursor buffered dari hasil pernyataan. Ini mengembalikan total waktu pemrosesan dari`ExecuteStatement` dan`FetchPage` permintaan.

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**
>
> Untuk mengkonversi ke kode sinkron, menghapus`await` dan`async` kata kunci, dan mengubah`IAsyncResult` jenis untuk`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**
>
> Kursor aliran stateful karena paginasi hasil set. Oleh karena itu,`getConsumedIOs` dan`getTimingInformation` operasi mengembalikan akumulasi metrik dari waktu yang Anda panggil mereka.
> Kursor buffer buffer hasil diatur dalam memori dan mengembalikan total akumulasi metrik.

Untuk mempelajari cara query katalog sistem, lanjutkan keMenanyakan katalog sistem.

## Menanyakan katalog sistem

Setiap tabel yang Anda buat dalam buku besar Amazon QLDB memiliki ID unik yang ditetapkan sistem. Anda dapat menemukan ID tabel, daftar indeksnya, dan metadata lainnya dengan menanyakan tabel katalog sistem`information_schema.user_tables`.

Semua ID yang ditetapkan sistem adalah pengidentifikasi unik universal (UUID) yang masing-masing diwakili dalam string yang dikodekan Base62. Untuk informasi selengkapnya, lihat ID unik di Amazon QLDB.

Contoh berikut menunjukkan hasil query yang mengembalikan atribut metadata dari`VehicleRegistration` tabel.

```
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"
}
```

Bidang metadata tabel

- `tableId`- ID unik dari tabel.
- `name`- Nama tabel.
- `indexes`— Daftar indeks di atas meja.
  - `indexId`- ID unik indeks.
  - `expr`- Jalur dokumen yang diindeks. Bidang ini adalah string dalam bentuk:`[fieldName]`.
  - `status`- Status indeks saat ini (`BUILDING`,`FINALIZING`,`ONLINE`,`FAILED`, atau`DELETING`). QLDB tidak menggunakan indeks dalam kueri sampai statusnya`ONLINE`.
  - `message`- Pesan kesalahan yang menjelaskan alasan bahwa indeks memiliki`FAILED` status. Bidang ini hanya disertakan untuk indeks gagal.
- `status`- Status tabel saat ini (`ACTIVE`atau`INACTIVE`). Sebuah meja menjadi`INACTIVE` ketika Anda`DROP` itu.

Untuk mempelajari cara mengelola tabel menggunakan`DROP TABLE` dan`UNDROP TABLE` pernyataan, lanjutkan keMengelola tabel.

# Mengelola tabel

Bagian ini menjelaskan cara mengelola tabel menggunakan`DROP TABLE` dan`UNDROP TABLE` pernyataan di Amazon QLDB. Bagian ini juga menjelaskan cara memberi tag pada tabel saat Anda membuatnya. Kuota untuk jumlah tabel aktif dan total tabel yang dapat Anda buat didefinisikan dalamKuota dan batasan di Amazon QLDB.

Topik

- Beri tag pada tabel saat penciptaan
- Menjatuhkan tabel
- Menanyakan sejarah tabel tidak aktif
- Mengaktifkan kembali tabel

## Beri tag pada tabel saat penciptaan

> **Note**
>
> Tabel penandaan pada pembuatan saat ini didukung untuk bukuSTANDARD besar dalam mode izin saja.

Anda dapat menandai sumber daya tabel Anda. Untuk mengelola tag untuk tabel yang ada, gunakanAWS Management Console atau operasi APITagResource,UntagResource, danListTagsForResource. Untuk informasi selengkapnya, lihat Pemberian tag pada sumber daya Amazon QLDB.

Anda juga dapat menentukan tag tabel saat Anda membuat tabel dengan menggunakan konsol QLDB, atau dengan menentukan mereka dalam pernyataanCREATE TABLE PartiQL. Contoh berikut membuat tabel bernamaVehicle dengan tagenvironment=production.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

Dengan menandai sumber daya saat sedang dibuat, Anda dapat menghilangkan kebutuhan untuk menjalankan skrip penandaan khusus setelah pembuatan sumber daya. Setelah tabel ditandai, Anda dapat mengontrol akses ke tabel berdasarkan tag tersebut. Misalnya, Anda dapat memberikan akses penuh hanya ke tabel yang memiliki tanda tertentu. Untuk contoh kebijakan JSON, lihatAkses penuh ke semua tindakan berdasarkan tag tabel.

## Menjatuhkan tabel

Untuk menjatuhkan meja, gunakanMEJA DROP pernyataan dasar. Ketika Anda menjatuhkan meja di QLDB, Anda hanya menonaktifkannya.

Misalnya, pernyataan berikut menonaktifkanVehicleRegistration tabel.

```
DROP TABLE VehicleRegistration
```

SebuahDROP TABLE pernyataan mengembalikan ID sistem-ditugaskan dari tabel. Status sekarangVehicleRegistration harusINACTIVE dalam tabel katalog sistem information_schema.user_tables.

```
SELECT status FROM information_schema.user_tables
```

```
WHERE name = 'VehicleRegistration'
```

## Menanyakan sejarah tabel tidak aktif

Selain nama tabel, Anda juga dapat query QLDBFungsi Riwayat dengan ID tabel sebagai argumen masukan pertama. Anda harus menggunakan ID tabel untuk menanyakan riwayat tabel tidak aktif. Setelah tabel dinonaktifkan, Anda tidak dapat lagi query sejarah dengan nama tabel.

Pertama, temukan ID tabel dengan menanyakan tabel katalog sistem. Misalnya, query berikut`tableId` mengembalikan`VehicleRegistration` tabel.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Kemudian, Anda dapat menggunakan ID ini untuk menjalankan kueri riwayat yang samaMelakukan Kueri Riwayat Riwayat Revisi. Berikut ini adalah contoh yang query sejarah ID dokumen`ADR2Ll1fGsU4Jr4EqTdnQF` dari ID tabel`5PLf9SXwndd63lPaSIa0O6`. ID tabel adalah string literal yang harus tertutup dalam tanda kutip tunggal.

```
--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'
```

## Mengaktifkan kembali tabel

Setelah Anda menonaktifkan tabel di QLDB, Anda dapat menggunakanTABEL UNDROP pernyataan untuk mengaktifkannya kembali.

Pertama, temukan ID tabel dari`information_schema.user_tables`. Misalnya, query berikut`tableId` mengembalikan`VehicleRegistration` tabel. Statusnya seharusnya`INACTIVE`.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Kemudian, gunakan ID ini untuk mengaktifkan kembali tabel. Berikut ini adalah contoh yang undrops ID tabel`5PLf9SXwndd63lPaSIa0O6`. Dalam hal ini, ID tabel adalah pengenal unik yang Anda lampirkan dalam tanda kutip ganda.

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

Status sekarang`VehicleRegistration` seharusnya`ACTIVE`.

Untuk mempelajari cara membuat, menggambarkan, dan menjatuhkan indeks, lanjutkan keMengelola indeks.

# Mengelola indeks

Bagian ini menjelaskan cara membuat, dan menjatuhkan indeks di Amazon QLDB. Kuota untuk jumlah indeks per tabel yang dapat Anda buat didefinisikan dalamKuota dan batasan di Amazon QLDB.

Topik

- Membuat indeks
- Menggambarkan indeks
- Menjatuhkan indeks
- Kesalahan umum

## Membuat indeks

Seperti juga dijelaskan dalamMembuat tabel dan indeks, Anda dapat menggunakan pernyataan CREATE INDEX untuk membuat indeks di atas meja untuk bidang tingkat atas tertentu, sebagai berikut. Nama tabel dan nama field yang diindeks keduanya peka huruf besar.

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

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

Setiap indeks yang Anda buat di atas meja memiliki ID unik yang ditetapkan sistem. Untuk menemukan ID indeks ini, lihat bagian berikutMenggambarkan indeks.

> **Important**
>
> QLDB membutuhkan indeks untuk secara efisien mencari dokumen. Tanpa indeks, QLDB perlu melakukan pemindaian meja penuh saat membaca dokumen. Hal ini dapat menyebabkan masalah kinerja pada tabel besar, termasuk konflik konkurensi dan batas waktu transaksi.

Untuk menghindari pemindaian tabel, Anda harus menjalankan pernyataan dengan klausa`WHERE` predikat menggunakan operator kesetaraan (`=`or`IN`) pada bidang yang diindeks atau ID dokumen. Untuk informasi selengkapnya, lihat Mengoptimalkan kinerja kueri.

Perhatikan kendala berikut saat membuat indeks:

- Indeks hanya dapat dibuat pada satu bidang tingkat atas. Indeks komposit, bersarang, unik, dan berbasis fungsi tidak didukung.
- Anda dapat membuat indeks pada jenis data Ion apa pun, termasuk`list` dan`struct`. Namun, Anda hanya dapat melakukan pencarian terindeks dengan kesetaraan seluruh nilai Ion terlepas dari jenis Ion. Misalnya, saat menggunakan`list` tipe sebagai indeks, Anda tidak dapat melakukan pencarian terindeks oleh satu item di dalam daftar.
- Kinerja kueri ditingkatkan hanya jika Anda menggunakan predikat kesetaraan; misalnya,`WHERE indexedField = 123` atau`WHERE indexedField IN (456, 789)`.

  QLDB tidak menghormati ketidaksetaraan dalam predikat query. Akibatnya, pemindaian rentang yang difilter tidak diimplementasikan.
- Nama bidang yang diindeks adalah huruf kapital dan dapat memiliki maksimal 128 karakter.
- Pembuatan indeks di QLDB bersifat asinkron. Jumlah waktu yang dibutuhkan untuk menyelesaikan membangun indeks pada tabel yang tidak kosong bervariasi tergantung pada ukuran tabel. Untuk informasi selengkapnya, lihat Mengelola indeks.

## Menggambarkan indeks

Pembuatan indeks di QLDB bersifat asinkron. Jumlah waktu yang dibutuhkan untuk menyelesaikan membangun indeks pada tabel yang tidak kosong bervariasi tergantung pada ukuran tabel. Untuk memeriksa status build indeks, Anda dapat menanyakan tabel katalog sistem information_schema.user_tables.

Misalnya, pernyataan berikut query katalog sistem untuk semua indeks di atas`VehicleRegistration` meja.

```
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"
}
```

Bidang indeks

- `indexId`- ID unik indeks.
- `expr`- Jalur dokumen yang diindeks. Bidang ini adalah string dalam bentuk:`[fieldName]`.
- `status`- Status indeks saat ini. Status indeks dapat berupa salah satu nilai berikut:
  - `BUILDING`— Secara aktif membangun indeks untuk tabel.
  - `FINALIZING`— Telah selesai membangun indeks dan mulai mengaktifkannya untuk digunakan.
  - `ONLINE`- Aktif dan siap digunakan dalam kueri. QLDB tidak menggunakan indeks dalam kueri sampai statusnya online.
  - `FAILED`- Tidak dapat membangun indeks karena kesalahan yang tidak dapat dipulihkan. Indeks dalam keadaan ini masih dihitung terhadap kuota indeks per tabel. Untuk informasi selengkapnya, lihat Kesalahan umum.
  - `DELETING`- Secara aktif menghapus indeks setelah pengguna menjatuhkannya.
- `message`- Pesan kesalahan yang menjelaskan alasan bahwa indeks memiliki`FAILED` status. Bidang ini hanya disertakan untuk indeks gagal.

Menggunakan konsol

Anda juga dapat menggunakanAWS Management Console untuk memeriksa status indeks.

Untuk memeriksa status indeks (konsol)

1. Masuk keAWS Management Console, dan buka konsol Amazon QLDB di https://console.aws.amazon.com/qldb.

2. Di panel navigasi, pilih Ledger.
3. Dalam daftar Buku Besar, pilih nama buku besar yang indeksnya ingin Anda kelola.
4. Pada halaman detail buku besar, di bawah tab Tabel, pilih nama tabel yang indeksnya ingin Anda periksa.
5. Pada halaman detail tabel, cari kartu bidang yang diindeks. Kolom status Indeks menampilkan status saat ini dari setiap indeks di atas meja.

## Menjatuhkan indeks

GunakanDROP INDEX pernyataan untuk menjatuhkan indeks. Ketika Anda menjatuhkan indeks, itu dihapus secara permanen dari tabel.

Pertama, temukan ID indeks dari`information_schema.user_tables`. Misalnya, query berikut mengembalikan`indexId` dari`LicensePlateNumber` bidang diindeks di atas`VehicleRegistration` meja.

```
SELECT indexes.indexId
FROM information_schema.user_tables info, info.indexes indexes
WHERE info.name = 'VehicleRegistration' and indexes.expr = '[LicensePlateNumber]'
```

Kemudian, gunakan ID ini untuk menjatuhkan indeks. Berikut adalah contoh yang menjatuhkan ID indeks`4tPW3fUhaVhDinRgKRLhGU`. ID Indeks adalah pengenal unik yang harus tertutup dalam tanda kutip ganda.

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

> **Note**
>
> Klausa`WITH (purge = true)` ini diperlukan untuk semua`DROP INDEX` pernyataan, dan saat`true` ini satu-satunya nilai yang didukung.
> Kata kunci`purge` adalah peka huruf besar/kecil dan harus berupa huruf kecil.

### Menggunakan konsol

Anda juga dapat menggunakanAWS Management Console untuk menjatuhkan indeks.

Untuk menjatuhkan indeks (konsol)

1. Masuk keAWS Management Console, dan buka konsol Amazon QLDB di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih Ledger.
3. Dalam daftar Buku Besar, pilih nama buku besar yang indeksnya ingin Anda kelola.
4. Pada halaman detail buku besar, di bawah tab Tabel, pilih nama tabel yang indeksnya ingin Anda jatuhkan.
5. Pada halaman detail tabel, cari kartu bidang yang diindeks. Pilih indeks yang ingin Anda jatuhkan, lalu pilih Drop index.

## Kesalahan umum

Bagian ini menjelaskan kesalahan umum yang mungkin Anda temui saat membuat indeks, dan menyarankan solusi yang mungkin.

> **Note**
>
> Indeks yang memiliki status`FAILED` masih dihitung terhadap kuota indeks per tabel. Indeks yang gagal juga mencegah Anda memodifikasi atau menghapus dokumen apa pun yang menyebabkan pembuatan indeks gagal di atas meja.
> Anda harus secara eksplisit menjatuhkan indeks untuk menghapusnya dari kuota.

Dokumen berisi beberapa nilai untuk bidang diindeks: ***`fieldName`***.

QLDB tidak dapat membangun indeks untuk nama field yang ditentukan karena tabel berisi dokumen dengan beberapa nilai untuk bidang yang sama (yaitu, nama field duplikat).

Anda harus terlebih dahulu menjatuhkan indeks gagal. Kemudian, pastikan bahwa semua dokumen dalam tabel hanya memiliki satu nilai untuk setiap nama bidang sebelum mencoba ulang pembuatan indeks. Anda juga dapat membuat indeks untuk bidang lain yang tidak memiliki duplikat.

QLDB juga mengembalikan kesalahan ini jika Anda mencoba untuk menyisipkan dokumen yang berisi beberapa nilai untuk bidang yang sudah diindeks di atas meja.

Melebihi batas indeks: Tabel ***tableName*** sudah memiliki ***n*** indeks, dan tidak dapat membuat lebih.

> QLDB memberlakukan batas lima indeks per tabel, termasuk indeks gagal. Anda harus menjatuhkan indeks yang ada sebelum membuat yang baru.

Tidak ada indeks didefinisikan dengan identifier: ***indexID***.

> Anda mencoba untuk menjatuhkan indeks yang tidak ada untuk tabel tertentu dan indeks ID kombinasi. Untuk mempelajari cara memeriksa indeks yang ada, lihatMenggambarkan indeks.

# ID unik di Amazon QLDB

Bagian ini menjelaskan properti dan pedoman penggunaan pengenal unik yang ditetapkan sistem di Amazon QLDB. Ini juga menyediakan beberapa contoh ID unik QLDB.

Topik

- Properti
- Penggunaan
- Contoh

## Properti

Semua ID yang ditetapkan sistem di QLDB adalah pengidentifikasi unik universal (UUID). Setiap ID memiliki properti berikut:

- Nomor UUID 128-bit
- Diwakili dalam teks Base62 dikodekan
- Panjang tetap string alfanumerik 22 karakter (misalnya:`3Qv67yjXEwB9SjmvkuG6Cp`)

## Penggunaan

Saat menggunakan ID unik QLDB di aplikasi Anda, perhatikan panduan berikut:

Lakukan

- Perlakukan ID sebagai string.

Jangan

- Cobalah untuk memecahkan kode string.
- Menganggap makna semantik untuk string (seperti menurunkan komponen waktu).
- Urutkan senar dalam urutan semantik.

## Contoh

Atribut berikut adalah beberapa contoh ID unik di QLDB:

- ID Dokumen
- ID Indeks
- ID Untaian
- ID Tabel
- ID Transaksi

# Model Konkurensi Amazon QLDB

Amazon QLDB dimaksudkan untuk memenuhi kebutuhan beban kerja pemrosesan transaksi online (OLTP) berkinerja tinggi. QLDB mendukung kemampuan query SQL-seperti, dan memberikan transaksi ACID penuh. Selain itu, item data QLDB adalah dokumen, memberikan fleksibilitas skema dan pemodelan data intuitif. Dengan jurnal pada intinya, Anda dapat menggunakan QLDB untuk mengakses riwayat lengkap dan dapat diverifikasi dari semua perubahan pada data Anda, dan melakukan streaming transaksi yang koheren ke layanan data lain sesuai kebutuhan.

Topik



## Pengendalian konkurensi Optimis

Dalam QLDB, kontrol konkurensi diimplementasikan menggunakan kontrol konkurensi optimis (OCC). OCC beroperasi pada prinsip bahwa beberapa transaksi sering dapat diselesaikan tanpa mengganggu satu sama lain.

Menggunakan OCC, transaksi di QLDB tidak memperoleh kunci pada sumber daya database dan beroperasi dengan isolasi serializable penuh. QLDB menjalankan transaksi bersamaan secara serial, sehingga menghasilkan efek yang sama seperti jika transaksi tersebut dimulai secara serial.

Sebelum melakukan, setiap transaksi melakukan pemeriksaan validasi untuk memastikan bahwa tidak ada transaksi berkomitmen lainnya yang memodifikasi data yang diakses. Jika pemeriksaan ini mengungkapkan modifikasi yang bertentangan, atau keadaan perubahan data, transaksi yang melakukan ditolak. Namun, transaksi dapat dimulai ulang.

Ketika transaksi menulis ke QLDB, pemeriksaan validasi model OCC diimplementasikan oleh QLDB itu sendiri. Jika transaksi tidak dapat ditulis ke jurnal karena kegagalan dalam fase verifikasi OCC, QLDB mengembalikan`OccConflictException` ke lapisan aplikasi. Perangkat lunak aplikasi

bertanggung jawab untuk memastikan bahwa transaksi dimulai ulang. Aplikasi harus membatalkan transaksi yang ditolak dan kemudian mencoba kembali seluruh transaksi dari awal.

Untuk mempelajari cara driver QLDB menangani dan mencoba ulang konflik OCC dan pengecualian sementara lainnya, lihatMemahami kebijakan coba ulang dengan pengemudi di Amazon QLDB.

## Menggunakan indeks untuk menghindari pemindaian tabel penuh

Dalam QLDB, setiap pernyataan PartiQL (termasuk setiap`SELECT` permintaan) diproses dalam transaksi dan tunduk pada batas batas waktu transaksi.

Sebagai praktik terbaik, Anda harus menjalankan pernyataan dengan klausa`WHERE` predikat yang menyaring pada bidang yang diindeks atau ID dokumen. QLDB membutuhkan operator kesetaraan pada bidang diindeks untuk secara efisien mencari dokumen; misalnya,`WHERE indexedField = 123` atau`WHERE indexedField IN (456, 789)`.

Tanpa pencarian yang diindeks ini, QLDB perlu melakukan pemindaian tabel penuh saat membaca dokumen. Hal ini dapat menyebabkan latensi kueri dan batas waktu transaksi, dan juga meningkatkan kemungkinan konflik OCC dengan transaksi yang bersaing.

Misalnya, pertimbangkan tabel bernama`Vehicle` yang memiliki indeks di`VIN` lapangan saja. Itu berisi dokumen-dokumen berikut.

| VIN | Membuat | Model | Warna |
|---|---|---|---|
| `"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"` |

Dua pengguna bersamaan bernama Alice dan Bob bekerja dengan tabel yang sama dalam buku besar. Mereka ingin memperbarui dua dokumen yang berbeda, sebagai berikut.

Alice:

```
UPDATE Vehicle AS v
SET v.Color = 'Blue'
WHERE v.VIN = '1N4AL11D75C109151'
```

Bob:

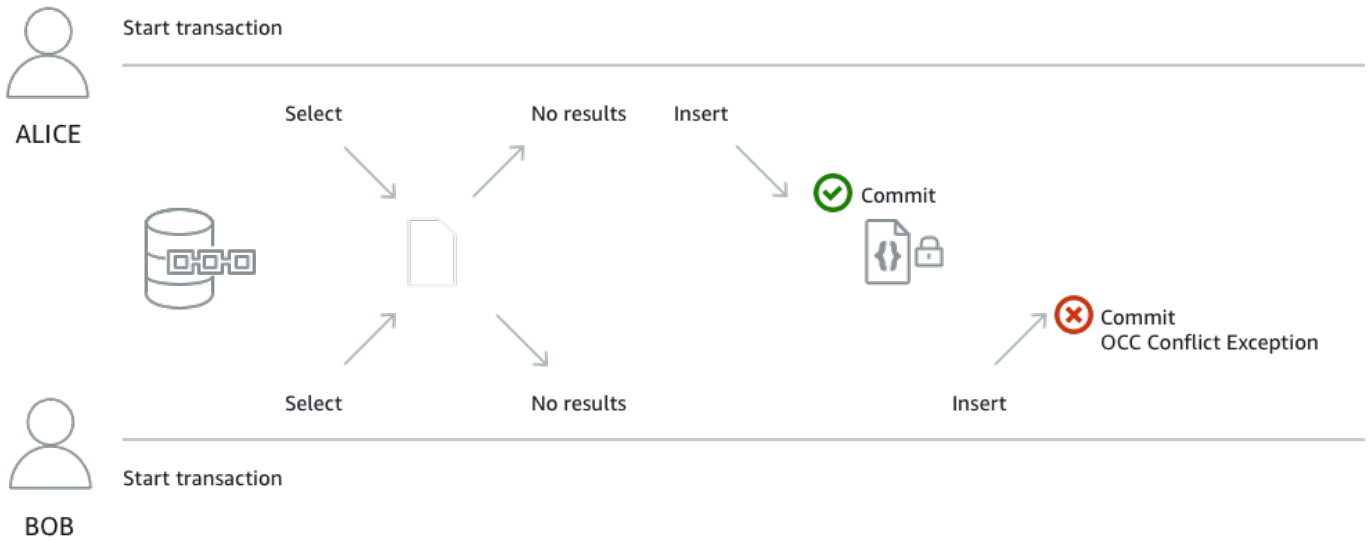
```
UPDATE Vehicle AS v
SET v.Color = 'Red'
WHERE v.Make = 'Tesla' AND v.Model = 'Model S'
```

Misalkan Alice dan Bob memulai transaksi mereka pada saat yang sama. `UPDATE`Pernyataan Alice melakukan pencarian diindeks di`VIN` lapangan, sehingga hanya perlu membaca bahwa satu dokumen. Alice selesai dan berhasil melakukan transaksinya terlebih dahulu.

pernyataan Bob menyaring pada bidang non-diindeks, sehingga melakukan scan tabel dan pertemuan`OccConflictException`. Ini karena transaksi Alice yang berkomitmen memodifikasi data yang diakses oleh pernyataan Bob, yang mencakup setiap dokumen dalam tabel—tidak hanya dokumen yang diperbarui Bob.

## Penyisipan konflik OCC

Konflik OCC dapat mencakup dokumen yang baru disisipkan—tidak hanya dokumen yang sebelumnya ada. Pertimbangkan diagram berikut, di mana dua pengguna serentak (Alice dan Bob) bekerja dengan tabel yang sama di buku besar. Mereka berdua ingin memasukkan dokumen baru hanya di bawah kondisi bahwa nilai predikat belum ada.




Dalam contoh ini, baik Alice dan Bob menjalankan berikut`SELECT` dan`INSERT` pernyataan dalam satu transaksi. Aplikasi mereka menjalankan`INSERT` pernyataan hanya jika`SELECT` pernyataan tidak mengembalikan hasil.

```
SELECT * FROM Vehicle v WHERE v.VIN = 'ABCDE12345EXAMPLE'
```

```
INSERT INTO Vehicle VALUE
{
    'VIN' : 'ABCDE12345EXAMPLE',
    'Type' : 'Wagon',
    'Year' : 2019,
    'Make' : 'Subaru',
    'Model' : 'Outback',
    'Color' : 'Gray'
}
```

Misalkan Alice dan Bob memulai transaksi mereka pada saat yang sama. Kedua`SELECT` pertanyaan mereka kembali tidak ada dokumen yang ada dengan`VIN` dari`ABCDE12345EXAMPLE`. Jadi, aplikasi mereka melanjutkan dengan`INSERT` pernyataan itu.

Alice selesai dan berhasil melakukan transaksinya terlebih dahulu. Kemudian, Bob mencoba untuk melakukan transaksinya, tetapi QLDB menolaknya dan melempar`OccConflictException`. Ini karena transaksi berkomitmen Alice memodifikasi kumpulan hasil`SELECT` kueri Bob, dan OCC mendeteksi konflik ini sebelum melakukan transaksi Bob.

`SELECT`Query diperlukan untuk contoh transaksi ini menjadi idempoten. Bob kemudian dapat mencoba kembali seluruh transaksinya dari awal. Tapi`SELECT` query berikutnya akan kembali dokumen yang Alice dimasukkan, sehingga aplikasi Bob tidak akan menjalankan`INSERT`.

## Melakukan transaksi idempoten

Transaksi insert di bagian sebelumnya juga merupakan contoh transaksi idempoten. Dengan kata lain, menjalankan transaksi yang sama beberapa kali menghasilkan hasil yang identik. Jika Bob menjalankan`INSERT` tanpa terlebih dahulu memeriksa apakah tertentu`VIN` sudah ada, tabel mungkin berakhir dengan dokumen yang memiliki`VIN` nilai duplikat.

Pertimbangkan skenario percobaan ulang lainnya selain konflik OCC. Misalnya, ada kemungkinan bahwa QLDB berhasil melakukan transaksi di sisi server, tetapi klien kali keluar sambil menunggu respon. Sebagai praktik terbaik, buat transaksi tulis Anda idempoten untuk menghindari efek samping yang tidak terduga dalam kasus konkurensi atau percobaan ulang.

## Redaksi Konflik OCC

QLDB mencegah redaksi bersamaan revisi pada blok jurnal yang sama. Pertimbangkan contoh di mana dua pengguna bersamaan (Alice dan Bob) ingin menyunting dua revisi dokumen berbeda yang dilakukan pada blok yang sama dalam buku besar. Pertama, Alice meminta redaksi satu revisi dengan menjalankan prosedur yang`REDACT_REVISION` disimpan, sebagai berikut.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

Kemudian, sementara permintaan Alice masih diproses, Bob meminta redaksi revisi lain, sebagai berikut.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

QLDB menolak permintaan Bob dengan`OccConflictException` meskipun mereka mencoba untuk menyunting dua revisi dokumen yang berbeda. Ini karena revisi Bob terletak di blok yang sama dengan revisi yang disunting Alice. Setelah permintaan Alice selesai diproses, Bob kemudian dapat mencoba lagi permintaan redaksinya.

Demikian pula, jika dua transaksi bersamaan mencoba menyunting revisi yang sama, hanya satu permintaan yang dapat diproses. Permintaan lainnya gagal dengan pengecualian konflik OCC

sampai redaksi selesai. Setelah itu, setiap permintaan untuk menyunting revisi yang sama akan menghasilkan kesalahan yang menunjukkan revisi sudah dihapus.

## Mengelola sesi serentak

Jika Anda memiliki pengalaman menggunakan sistem manajemen database relasional (RDBMS), Anda mungkin terbiasa dengan batas koneksi konkurensi. QLDB tidak memiliki konsep yang sama dari koneksi RDBMS tradisional karena transaksi dijalankan dengan permintaan HTTP dan pesan respon.

Dalam QLDB, konsep analog adalah sesi aktif. Sesi secara konseptual mirip dengan login pengguna—ia mengelola informasi tentang permintaan transaksi data Anda ke buku besar. Sesi aktif adalah sesi yang secara aktif menjalankan transaksi. Ini juga bisa menjadi sesi yang baru-baru ini menyelesaikan transaksi di mana layanan mengantisipasi akan segera memulai transaksi lain. QLDB mendukung satu transaksi yang berjalan secara aktif per sesi.

Batas sesi aktif bersamaan per buku besar didefinisikan dalamKuota dan batasan di Amazon QLDB. Setelah batas ini tercapai, setiap sesi yang mencoba memulai transaksi akan mengakibatkan error (`LimitExceededException`).

Untuk informasi tentang siklus hidup sesi dan cara driver QLDB menangani sesi saat menjalankan transaksi data, lihatManajemen sesi dengan pengemudi. Untuk praktik terbaik untuk mengonfigurasi kumpulan sesi dalam aplikasi Anda menggunakan driver QLDB, lihatMengonfigurasi QldbDriver objek di rekomendasi driver Amazon QLDB.

# Verifikasi data di Amazon QLDB

Dengan Amazon QLDB, Anda dapat mempercayai bahwa riwayat perubahan pada data aplikasi Anda akurat. QLDB menggunakan log transaksional yang tidak dapat diubah, yang dikenal sebagai jurnal, untuk penyimpanan data. Jurnal melacak setiap perubahan pada data komitmen Anda dan mempertahankan riwayat perubahan yang lengkap dan dapat diverifikasi dari waktu ke waktu.

QLDB menggunakan fungsi hash SHA-256 dengan model berbasis pohon Merkle untuk menghasilkan representasi kriptografi jurnal Anda, yang dikenal sebagai intisari. Intisari bertindak sebagai tanda tangan unik dari seluruh riwayat perubahan data Anda pada suatu titik waktu. Anda menggunakan intisari untuk memverifikasi integritas revisi dokumen Anda relatif terhadap tanda tangan tersebut.

Topik



## Jenis data apa yang dapat Anda verifikasi di QLDB?

Dalam QLDB, setiap buku besar memiliki tepat satu jurnal. Jurnal dapat memiliki banyak untaian, yang merupakan partisi jurnal.

> **Note**
>
> QLDB saat ini mendukung jurnal dengan untai tunggal saja.

Blok adalah objek yang berkomitmen pada untaian jurnal selama transaksi. Blok ini berisi objek entri, yang mewakili revisi dokumen yang dihasilkan dari transaksi. Anda dapat memverifikasi revisi individu atau seluruh blok jurnal di QLDB.

Diagram berikut menggambarkan struktur jurnal ini.

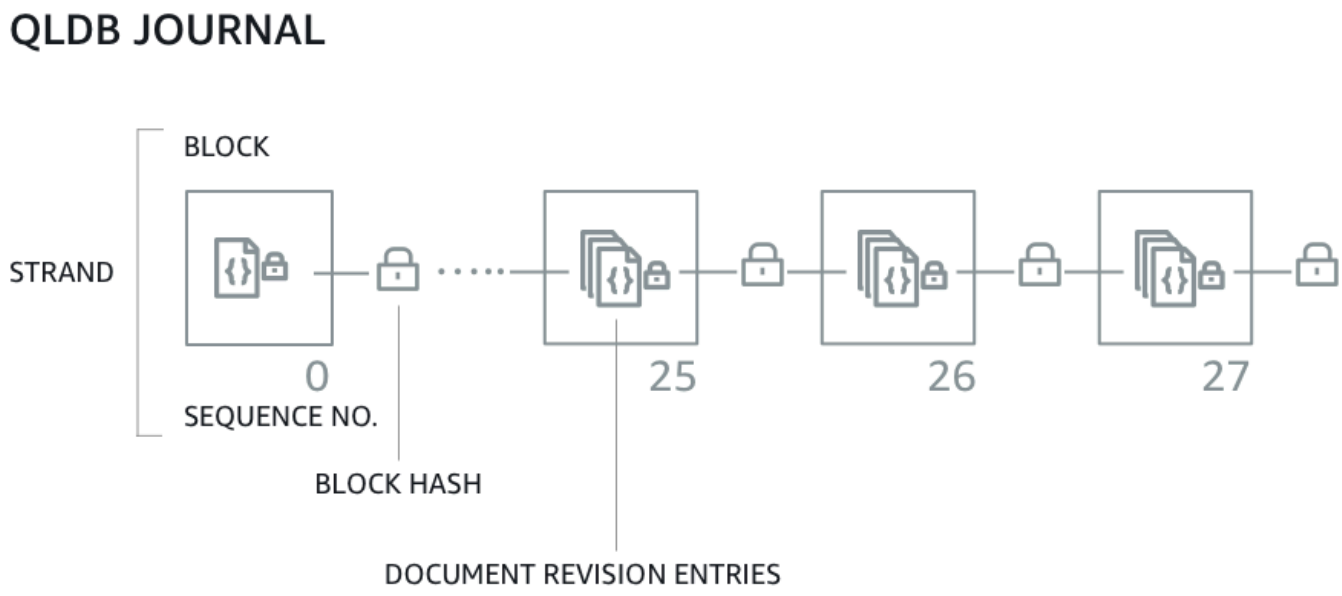



Diagram menunjukkan bahwa transaksi dilakukan pada jurnal sebagai blok yang berisi entri revisi dokumen. Ini juga menunjukkan bahwa setiap blok dirantai hash ke blok berikutnya dan memiliki nomor urut untuk menentukan alamatnya di dalam untai.

Untuk informasi tentang konten data dalam blok, lihatIsi jurnal di Amazon QLDB.

## Apa arti integritas data?

Integritas data dalam QLDB berarti bahwa jurnal buku besar Anda sebenarnya tidak dapat diubah. Dengan kata lain, data Anda (khususnya, setiap revisi dokumen) berada dalam keadaan di mana berikut ini benar:

1. Itu ada di lokasi yang sama di jurnal Anda tempat pertama kali ditulis.
2. Itu belum diubah dengan cara apa pun sejak ditulis.

# Bagaimana cara kerja verifikasi?

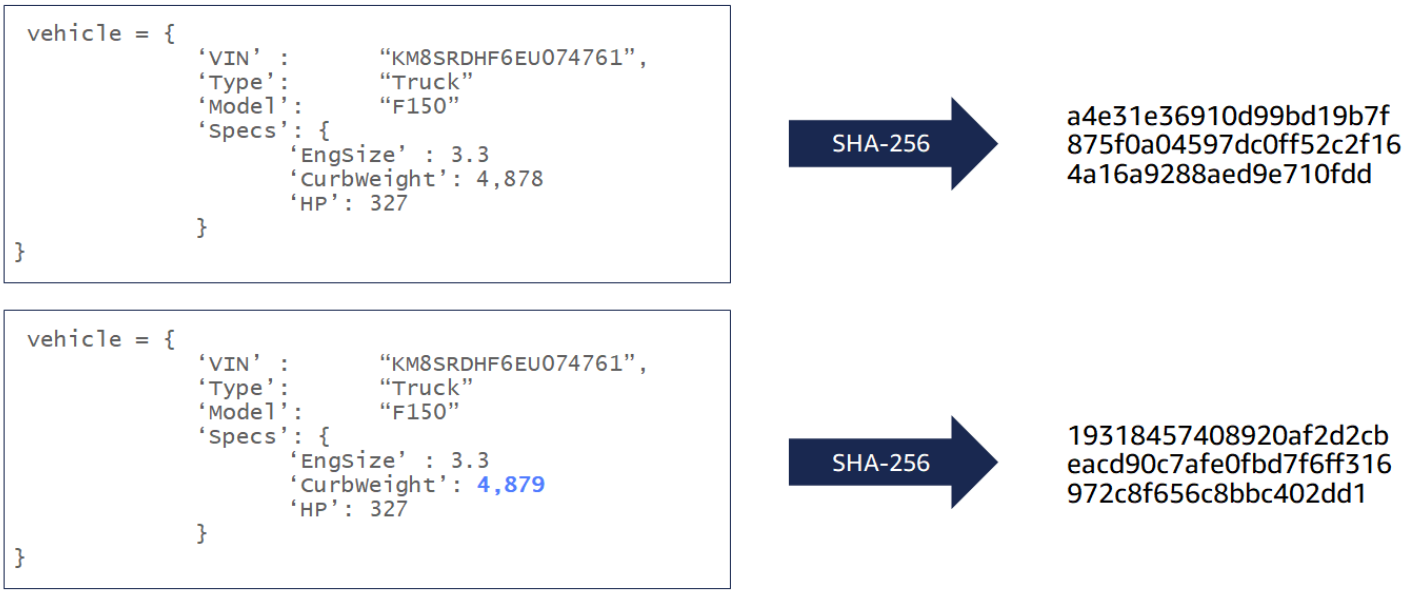
Untuk memahami cara kerja verifikasi di Amazon QLDB, Anda dapat memecah konsep menjadi empat komponen dasar.

- Hashing
- Digest
- Pohon Merkle
- Bukti

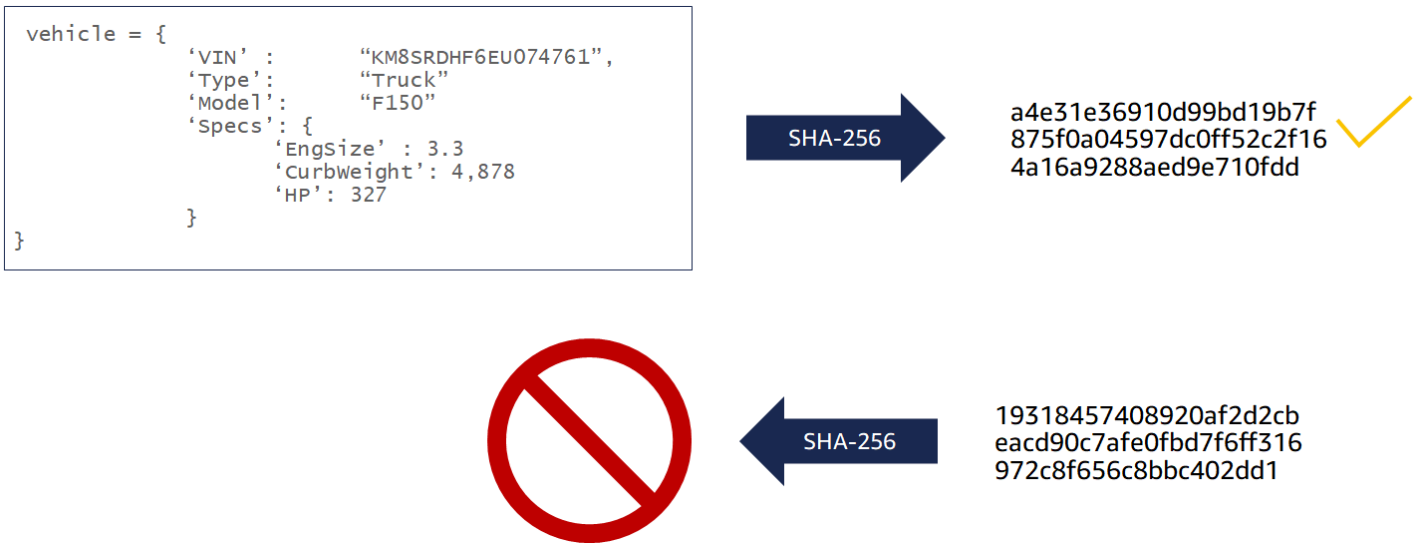
## Hashing

QLDB menggunakan fungsi hash kriptografi SHA-256 untuk membuat nilai hash 256-bit. Hash bertindak sebagai tanda tangan unik dan panjang tetap dari jumlah data input yang sewenang-wenang. Jika Anda mengubah bagian mana pun dari input—bahkan satu karakter atau bit—maka hash keluaran berubah sepenuhnya.

Diagram berikut menunjukkan bahwa fungsi hash SHA-256 menciptakan nilai hash yang benar-benar unik untuk dua dokumen QLDB yang berbeda hanya dengan satu digit.




Fungsi hash SHA-256 adalah salah satu cara, yang berarti bahwa secara matematis tidak layak untuk menghitung input ketika diberi output. Diagram berikut menunjukkan bahwa tidak layak untuk menghitung dokumen QLDB masukan ketika diberi nilai hash output.




Input data berikut di-hash di QLDB untuk tujuan verifikasi:

- Revisi dokumen
- Pernyataan PartiQL
- Entri revisi
- Blok jurnal

## Digest

Intisari adalah representasi kriptografi dari seluruh jurnal buku besar Anda pada suatu titik waktu. Sebuah jurnal hanya ditambahkan, dan blok jurnal diurutkan dan dirantai hash mirip dengan blockchain.

Anda dapat meminta intisari untuk buku besar kapan saja. QLDB menghasilkan intisari dan mengembalikannya kepada Anda sebagai file keluaran aman. Kemudian Anda menggunakan intisari itu untuk memverifikasi integritas revisi dokumen yang dilakukan pada titik waktu sebelumnya. Jika Anda menghitung ulang hash dengan memulai dengan revisi dan diakhiri dengan intisari, Anda membuktikan bahwa data Anda belum diubah di antaranya.

## Pohon Merkle

Seiring bertambahnya ukuran buku besar Anda, semakin tidak efisien untuk menghitung ulang rantai hash penuh jurnal untuk verifikasi. QLDB menggunakan model pohon Merkle untuk mengatasi inefisiensi ini.

Pohon Merkle adalah struktur data pohon di mana setiap simpul daun mewakili hash dari blok data. Setiap node non-daun adalah hash dari node anaknya. Umumnya digunakan dalam blockchain, pohon Merkle membantu Anda memverifikasi kumpulan data besar secara efisien dengan mekanisme bukti audit. Untuk informasi lebih lanjut tentang pohon Merkle, lihat halaman Wikipedia pohon Merkle. Untuk mempelajari lebih lanjut tentang bukti audit Merkle dan untuk contoh kasus penggunaan, lihat Cara Kerja Bukti Log di situs Transparansi Sertifikat.

Implementasi QLDB dari pohon Merkle dibangun dari rantai hash penuh jurnal. Dalam model ini, node daun adalah kumpulan semua hash revisi dokumen individual. Simpul akar mewakili intisari seluruh jurnal pada titik waktu.

Menggunakan bukti audit Merkle, Anda dapat memverifikasi revisi dengan memeriksa hanya sebagian kecil dari riwayat revisi buku besar Anda. Anda melakukan ini dengan melintasi pohon dari simpul daun tertentu (revisi) ke akarnya (intisari). Sepanjang jalur traversal ini, Anda secara rekursif melakukan hash pasangan node saudara kandung untuk menghitung hash induknya sampai Anda berakhir dengan intisari. Traversal ini memiliki kompleksitas waktu `log(n)` node di pohon.

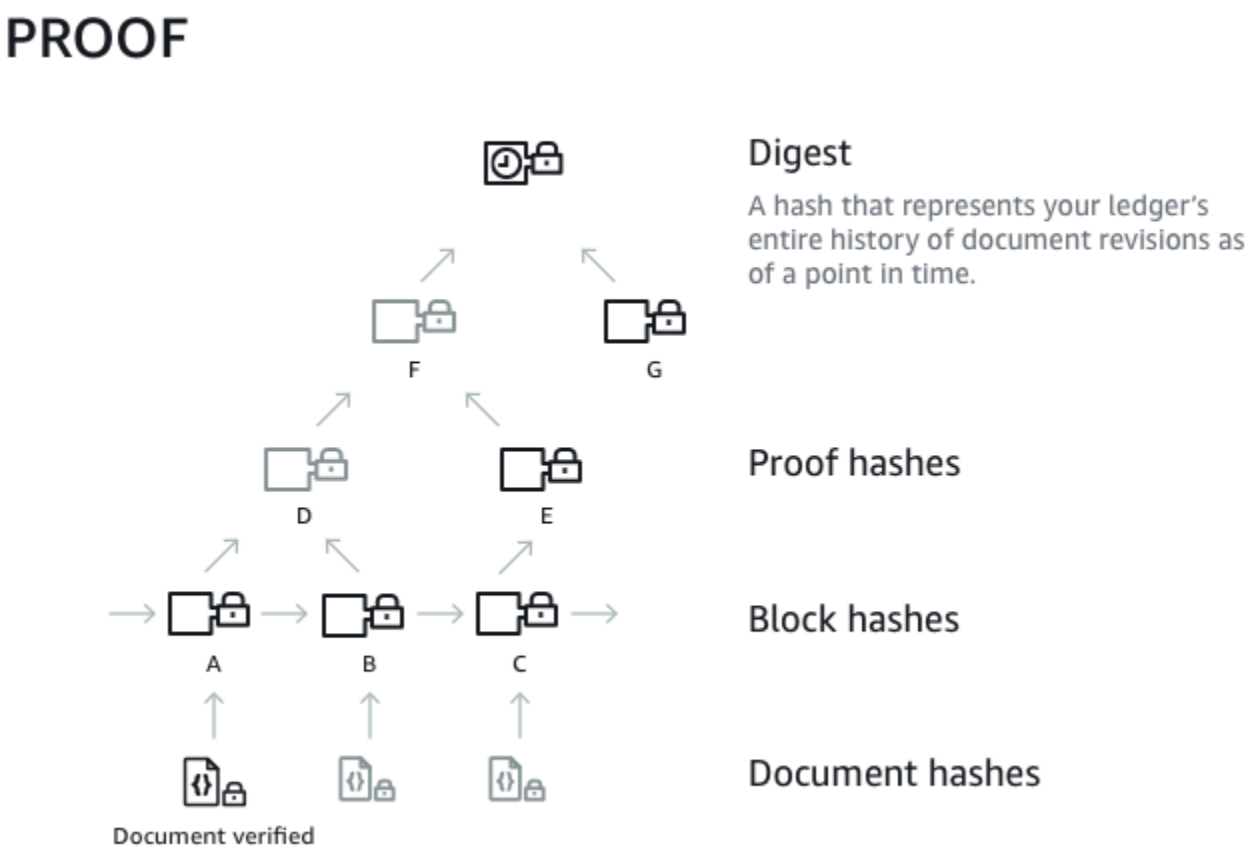
### Bukti

Bukti adalah daftar urutan hash node yang dikembalikan QLDB untuk intisari dan revisi dokumen tertentu. Ini terdiri dari hash yang diperlukan oleh model pohon Merkle untuk merantai hash simpul daun yang diberikan (revisi) ke hash root (intisari).

Mengubah data yang berkomitmen antara revisi dan intisari merusak rantai hash jurnal Anda dan membuatnya tidak mungkin untuk menghasilkan bukti.

## Contoh verifikasi

Diagram berikut mengilustrasikan model pohon hash Amazon QLDB. Ini menunjukkan satu set hash blok yang menggulung ke simpul akar atas, yang mewakili intisari untai jurnal. Dalam buku besar dengan jurnal untai tunggal, simpul akar ini juga merupakan intisari dari seluruh buku besar.




Misalkan node A adalah blok yang berisi revisi dokumen yang hash ingin Anda verifikasi. Node berikut mewakili daftar hash yang diurutkan yang disediakan QLDB dalam bukti Anda: B, E, G. Hash ini diperlukan untuk menghitung ulang intisari dari hash A.

Untuk menghitung ulang intisari, lakukan hal berikut:

1. Mulailah dengan hash A dan sambungkan dengan hash B. Kemudian, hash hasilnya untuk menghitung D.
2. Gunakan D dan E untuk menghitung F.
3. Gunakan F dan G untuk menghitung intisari.

Verifikasi berhasil jika intisari yang dihitung ulang sesuai dengan nilai yang diharapkan. Mengingat hash revisi dan intisari, tidak layak untuk merekayasa balik hash sebagai bukti. Oleh karena itu, latihan ini membuktikan bahwa revisi Anda memang ditulis di lokasi jurnal ini relatif terhadap intisari.

# Bagaimana redaksi data memengaruhi verifikasi?

Di Amazon QLDB, `DELETE` sebuah pernyataan hanya secara logis menghapus dokumen dengan membuat revisi baru yang menandainya sebagai dihapus. QLDB juga mendukung operasi redaksi data yang memungkinkan Anda menghapus revisi dokumen yang tidak aktif secara permanen dalam riwayat tabel.

Operasi redaksi hanya menghapus data pengguna dalam revisi yang ditentukan, dan membiarkan urutan jurnal dan metadata dokumen tidak berubah. Setelah revisi disunting, data pengguna dalam revisi (diwakili oleh `data` struktur) digantikan oleh bidang baru. `dataHash` Nilai bidang ini adalah hash Amazon Ion dari `data` struktur yang dihapus. Untuk informasi lebih lanjut dan contoh operasi redaksi, lihatMenyunting revisi dokumen.

Akibatnya, buku besar mempertahankan integritas data secara keseluruhan dan tetap dapat diverifikasi secara kriptografis melalui operasi API verifikasi yang ada. Anda masih dapat menggunakan operasi API ini seperti yang diharapkan untuk meminta digest (GetDigest), meminta bukti (GetBlockatau GetRevision), dan kemudian menjalankan algoritme verifikasi menggunakan objek yang dikembalikan.

## Menghitung ulang hash revisi

Jika Anda berencana untuk memverifikasi revisi dokumen individual dengan menghitung ulang hash, Anda harus memeriksa secara kondisional apakah revisi telah dihapus. Jika revisi telah dihapus, Anda dapat menggunakan nilai hash yang disediakan di bidang. `dataHash` Jika tidak disunting, Anda dapat menghitung ulang hash dengan menggunakan bidang. `data`

Dengan melakukan pemeriksaan bersyarat ini, Anda dapat mengidentifikasi revisi yang disunting dan mengambil tindakan yang sesuai. Misalnya, Anda dapat mencatat peristiwa manipulasi data untuk tujuan pemantauan.

# Memulai dengan verifikasi

Sebelum Anda dapat memverifikasi data, Anda harus meminta intisari dari buku besar Anda dan menyimpannya untuk nanti. Setiap revisi dokumen yang dilakukan sebelum blok terbaru yang dicakup oleh intisari memenuhi syarat untuk verifikasi terhadap intisari tersebut.

Kemudian, Anda meminta bukti dari Amazon QLDB untuk revisi yang memenuhi syarat yang ingin Anda verifikasi. Dengan menggunakan bukti ini, Anda memanggil API sisi klien untuk menghitung

ulang intisari, dimulai dengan hash revisi Anda. Selama intisari yang disimpan sebelumnya diketahui dan dipercaya di luar QLDB, integritas dokumen Anda terbukti jika hash intisari yang dihitung ulang cocok dengan hash intisari yang disimpan.

> **Important**
>
> - Apa yang secara khusus Anda buktikan adalah bahwa revisi dokumen tidak diubah antara waktu Anda menyimpan intisari ini dan saat Anda menjalankan verifikasi. Anda dapat meminta dan menyimpan intisari segera setelah revisi yang ingin Anda verifikasi nanti dilakukan ke jurnal.
> - Sebagai praktik terbaik, kami menyarankan Anda meminta intisari secara teratur dan menyimpannya jauh dari buku besar. Tentukan frekuensi yang Anda minta intisari berdasarkan seberapa sering Anda melakukan revisi di buku besar Anda.
>
>   Untuk posting AWS blog terperinci yang membahas nilai verifikasi kriptografi dalam konteks kasus penggunaan yang realistis, lihat Verifikasi kriptografi dunia nyata dengan Amazon QLDB.

Untuk step-by-step panduan tentang cara meminta intisari dari buku besar Anda dan kemudian memverifikasi data Anda, lihat berikut ini:

- Langkah 1: Meminta intisari di QLDB
- Langkah 2: Memverifikasi data Anda di QLDB

# Langkah 1: Meminta intisari di QLDB

Amazon QLDB menyediakan API untuk meminta intisari yang mencakup ujung jurnal saat ini di buku besar Anda. Tip jurnal mengacu pada blok komitmen terbaru pada saat QLDB menerima permintaan Anda. Anda dapat menggunakan AWS Management Console, AWS SDK, atau AWS Command Line Interface (AWS CLI) untuk mendapatkan intisari.

Topik

- AWS Management Console
- QLDB API

# AWS Management Console

Ikuti langkah-langkah ini untuk meminta intisari menggunakan konsol QLDB.

Untuk meminta intisari (konsol)

1. Masuk ke AWS Management Console, dan buka konsol QLDB Amazon di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih Buku Besar.
3. Dalam daftar buku besar, pilih nama buku besar yang ingin Anda minta intisari.
4. Pilih Dapatkan intisari. Kotak dialog Get digest menampilkan detail intisari berikut:
   - Digest — Nilai hash SHA-256 dari intisari yang Anda minta.
   - Alamat tip intisari — Lokasi blok terbaru dalam jurnal yang dicakup oleh intisari yang Anda minta. Sebuah alamat memiliki dua bidang berikut:
     - `strandId`— ID unik untai jurnal yang berisi blok.
     - `sequenceNo`— Nomor indeks yang menentukan lokasi blok di dalam untai.
   - Buku Besar — Nama buku besar yang Anda minta intisari.
   - Tanggal — Stempel waktu saat Anda meminta intisari.
5. Tinjau informasi intisari. Lalu, pilih Simpan. Anda dapat menyimpan nama file default, atau memasukkan nama baru.

   > **Note**
   >
   > Anda mungkin memperhatikan bahwa nilai hash dan alamat tip intisari Anda berubah bahkan ketika Anda tidak memodifikasi data apa pun di buku besar Anda. Ini karena konsol mengambil katalog sistem buku besar setiap kali Anda menjalankan kueri di editor PartiQL. Ini adalah transaksi baca yang berkomitmen pada jurnal dan menyebabkan alamat blok terbaru berubah.

   Langkah ini menyimpan file plaintext dengan konten dalam format Amazon Ion. File ini memiliki ekstensi nama file `.ion.txt` dan berisi semua informasi intisari yang tercantum pada kotak dialog sebelumnya. Berikut ini adalah contoh isi file digest. Urutan bidang dapat bervariasi tergantung pada browser Anda.

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

6. Simpan file ini di mana Anda dapat mengaksesnya di masa depan. Kemudian, Anda dapat menggunakan file ini untuk memverifikasi revisi dokumen terhadap.

> **Important**
>
> Revisi dokumen yang Anda verifikasi nanti harus dicakup oleh intisari yang Anda simpan. Artinya, nomor urut alamat dokumen harus kurang dari atau sama dengan nomor urut alamat tip Digest.

## QLDB API

Anda juga dapat meminta intisari dari buku besar Anda dengan menggunakan Amazon QLDB API dengan SDK atau file. AWS AWS CLI QLDB API menyediakan operasi berikut untuk digunakan oleh program aplikasi:

- GetDigest— Mengembalikan intisari buku besar di blok komitmen terbaru dalam jurnal. Responsnya mencakup nilai hash 256-bit dan alamat blok.

Untuk informasi tentang meminta intisari menggunakan AWS CLI, lihat perintah get-digest di Referensi Perintah.AWS CLI

### Aplikasi sampel

Untuk contoh kode Java, lihat GitHub repositori amazon-qldb-dmv-sampleaws-samples/ -java. Untuk petunjuk tentang cara mengunduh dan menginstal aplikasi sampel ini, lihatMenginstal aplikasi sampel Amazon QLDB Java. Sebelum meminta intisari, pastikan Anda mengikuti Langkah 1-3 tutorial java untuk membuat buku besar sampel dan memuatnya dengan data sampel.

Kode tutorial di kelas GetDigestmemberikan contoh meminta intisari dari buku besar `vehicle-registration` sampel.

Untuk memverifikasi revisi dokumen menggunakan intisari yang Anda simpan, lanjutkan ke. Langkah 2: Memverifikasi data Anda di QLDB

# Langkah 2: Memverifikasi data Anda di QLDB

Amazon QLDB menyediakan API untuk meminta bukti ID dokumen tertentu dan blok terkaitnya. Anda juga harus memberikan alamat tip dari intisari yang sebelumnya Anda simpan, seperti yang dijelaskan dalamLangkah 1: Meminta intisari di QLDB. Anda dapat menggunakan AWS Management Console, AWS SDK, atau AWS CLI untuk mendapatkan bukti.

Kemudian, Anda dapat menggunakan bukti yang dikembalikan oleh QLDB untuk memverifikasi revisi dokumen terhadap intisari yang disimpan, menggunakan API sisi klien. Ini memberi Anda kontrol atas algoritme yang Anda gunakan untuk memverifikasi data Anda.

Topik

- AWS Management Console
- QLDB API

## AWS Management Console

Bagian ini menjelaskan langkah-langkah untuk memverifikasi revisi dokumen terhadap intisari yang disimpan sebelumnya menggunakan konsol QLDB Amazon.

Sebelum Anda mulai, pastikan Anda mengikuti langkah-langkahnyaLangkah 1: Meminta intisari di QLDB. Verifikasi memerlukan intisari yang disimpan sebelumnya yang mencakup revisi yang ingin Anda verifikasi.

Untuk memverifikasi revisi dokumen (konsol)

1. Buka konsol QLDB Amazon di https://console.aws.amazon.com/qldb.
2. Pertama, kueri buku besar Anda untuk `id` dan `blockAddress` revisi yang ingin Anda verifikasi. Bidang ini disertakan dalam metadata dokumen, yang dapat Anda kueri dalam tampilan komit.

   Dokumen `id` ini adalah string ID unik yang ditetapkan sistem. `blockAddress`Ini adalah struktur Ion yang menentukan lokasi blok tempat revisi dilakukan.

   Di panel navigasi, pilih editor PartiQL.
3. Pilih nama buku besar di mana Anda ingin memverifikasi revisi.

4. Di jendela editor kueri, masukkan `SELECT` pernyataan dalam sintaks berikut, lalu pilih Jalankan.

```
SELECT metadata.id, blockAddress FROM _ql_committed_table_name
WHERE criteria
```

Misalnya, query berikut mengembalikan dokumen dari `VehicleRegistration` tabel dalam contoh buku besar yang dibuat diMemulai dengan konsol Amazon QLDB.

```
SELECT r.metadata.id, r.blockAddress FROM _ql_committed_VehicleRegistration AS r
WHERE r.data.VIN = 'KM8SRDHF6EU074761'
```

5. Salin dan simpan `id` dan `blockAddress` nilai yang dikembalikan kueri Anda. Pastikan untuk menghilangkan tanda kutip ganda untuk `id` bidang tersebut. Di Amazon Ion, tipe data string dibatasi dengan tanda kutip ganda. Misalnya, Anda harus menyalin hanya teks alfanumerik dalam cuplikan berikut.

```
"LtMNJYNjSwzBLgf7sLifrG"
```

6. Sekarang setelah Anda memilih revisi dokumen, Anda dapat memulai proses memverifikasinya.

Di panel navigasi, pilih Verifikasi.

7. Pada formulir Verifikasi dokumen, di bawah Tentukan dokumen yang ingin Anda verifikasi, masukkan parameter input berikut:

- Buku besar — Buku besar di mana Anda ingin memverifikasi revisi.
- Alamat blok — `blockAddress` Nilai yang dikembalikan oleh kueri Anda di langkah 4.
- ID Dokumen - `id` Nilai yang dikembalikan oleh kueri Anda di langkah 4.

8. Di bawah Tentukan intisari yang akan digunakan untuk verifikasi, pilih intisari yang sebelumnya Anda simpan dengan memilih Pilih intisari. Jika file tersebut valid, ini secara otomatis mengisi semua bidang intisari di konsol Anda. Atau, Anda dapat menyalin dan menempelkan nilai berikut secara manual langsung dari file intisari Anda:

- Digest — `digest` Nilai dari file digest Anda.
- Alamat tip intisari — `digestTipAddress` Nilai dari file intisari Anda.

9. Tinjau dokumen Anda dan intisari parameter input, lalu pilih Verifikasi.

Konsol mengotomatiskan dua langkah untuk Anda:

a. Minta bukti dari QLDB untuk dokumen yang Anda tentukan.

b. Gunakan bukti yang dikembalikan oleh QLDB untuk memanggil API sisi klien, yang memverifikasi revisi dokumen Anda terhadap intisari yang disediakan. Untuk memeriksa algoritma verifikasi ini, lihat bagian berikut QLDB API untuk mengunduh contoh kode.

Konsol menampilkan hasil permintaan Anda di kartu hasil Verifikasi. Untuk informasi selengkapnya, lihat Hasil verifikasi.

## QLDB API

Anda juga dapat memverifikasi revisi dokumen dengan menggunakan Amazon QLDB API dengan AWS SDK atau file. AWS CLI QLDB API menyediakan operasi berikut untuk digunakan oleh program aplikasi:

- `GetDigest`— Mengembalikan intisari buku besar di blok komitmen terbaru dalam jurnal. Responsnya mencakup nilai hash 256-bit dan alamat blok.
- `GetBlock`— Mengembalikan objek blok pada alamat tertentu dalam jurnal. Juga mengembalikan bukti blok yang ditentukan untuk verifikasi jika `DigestTipAddress` disediakan.
- `GetRevision`— Mengembalikan objek data revisi untuk ID dokumen tertentu dan alamat blok. Juga mengembalikan bukti revisi yang ditentukan untuk verifikasi jika `DigestTipAddress` disediakan.

Untuk deskripsi lengkap tentang operasi API ini, lihat. Referensi API Amazon QLDB

Untuk informasi tentang memverifikasi data menggunakan AWS CLI, lihat Referensi AWS CLI Perintah.

### Aplikasi sampel

Untuk contoh kode Java, lihat GitHub repositori amazon-qldb-dmv-sampleaws-samples/ -java. Untuk petunjuk tentang cara mengunduh dan menginstal aplikasi sampel ini, lihatMenginstal aplikasi sampel Amazon QLDB Java. Sebelum melakukan verifikasi, pastikan Anda mengikuti Langkah 1-3 tutorial java untuk membuat buku besar sampel dan memuatnya dengan data sampel.

Kode tutorial di kelas GetRevisionmemberikan contoh meminta bukti untuk revisi dokumen dan kemudian memverifikasi revisi itu. Kelas ini menjalankan langkah-langkah berikut:

1. Meminta intisari baru dari buku besar `vehicle-registration` sampel.

2. Meminta bukti untuk revisi dokumen sampel dari `VehicleRegistration` tabel di `vehicle-registration` buku besar.
3. Memverifikasi revisi sampel menggunakan intisari dan bukti yang dikembalikan.

## Hasil verifikasi

Bagian ini menjelaskan hasil yang dikembalikan oleh permintaan verifikasi data QLDB Amazon pada. AWS Management Console Untuk langkah-langkah rinci tentang cara mengirimkan permintaan verifikasi, lihatLangkah 2: Memverifikasi data Anda di QLDB.

Pada halaman Verifikasi konsol QLDB, hasil permintaan Anda ditampilkan di kartu hasil Verifikasi. Tab Bukti menunjukkan isi bukti yang dikembalikan oleh QLDB untuk revisi dan intisari dokumen yang Anda tentukan. Ini mencakup rincian berikut:

- Revisi hash — Nilai SHA-256 yang secara unik mewakili revisi dokumen yang Anda verifikasi.
- Bukti hash - Daftar urutan hash yang disediakan oleh QLDB yang digunakan untuk menghitung ulang intisari yang ditentukan. Konsol dimulai dengan hash Revisi dan secara berurutan menggabungkannya dengan setiap hash bukti hingga berakhir dengan intisari yang dihitung ulang.

  Daftar ini diciutkan secara default, sehingga Anda dapat memperluasnya untuk mengungkapkan nilai hash. Secara opsional, Anda dapat mencoba perhitungan hash sendiri dengan mengikuti langkah-langkah seperti yang dijelaskan dalam. Menggunakan bukti untuk menghitung ulang intisari Anda
- Digest Calculated — Hash yang dihasilkan dari serangkaian perhitungan Hash yang dilakukan pada hash Revisi. Jika nilai ini cocok dengan Digest yang Anda simpan sebelumnya, verifikasi berhasil.

Tab Blokir menampilkan konten blok yang berisi revisi yang Anda verifikasi. Ini mencakup rincian berikut:

- ID Transaksi — ID unik dari transaksi yang melakukan blok ini.
- Waktu transaksi — Stempel waktu ketika blok ini berkomitmen pada untai.
- Block hash — Nilai SHA-256 yang secara unik mewakili blok ini dan semua isinya.
- Alamat blok — Lokasi di jurnal buku besar Anda tempat blok ini dilakukan. Sebuah alamat memiliki dua bidang berikut:
  - Strand ID — ID unik untai jurnal yang berisi blok ini.

  - Nomor urutan - Nomor indeks yang menentukan lokasi blok ini di dalam untai.
- Pernyataan — Pernyataan PartiQL yang dilakukan untuk melakukan entri di blok ini.

> **Note**
>
> Jika Anda menjalankan pernyataan berparameter secara terprogram, pernyataan tersebut direkam di blok jurnal Anda dengan parameter pengikat, bukan data literal. Misalnya, Anda mungkin melihat pernyataan berikut di blok jurnal, di mana tanda tanya (?) adalah placeholder variabel untuk isi dokumen.
>
> ```
> INSERT INTO Vehicle ?
> ```

- Entri dokumen — Revisi dokumen yang dilakukan di blok ini.

Jika permintaan Anda gagal memverifikasi revisi dokumen, lihat Kesalahan umum untuk verifikasi informasi tentang kemungkinan penyebabnya.

## Menggunakan bukti untuk menghitung ulang intisari Anda

Setelah QLDB mengembalikan bukti untuk permintaan verifikasi dokumen Anda, Anda dapat mencoba melakukan perhitungan hash sendiri. Bagian ini menjelaskan langkah-langkah tingkat tinggi untuk menghitung ulang intisari Anda menggunakan bukti yang disediakan.

Pertama, pasangkan hash Revisi Anda dengan hash pertama dalam daftar hash Proof. Kemudian, lakukan langkah-langkah berikut.

1. Urutkan dua hash. Bandingkan hash dengan nilai byte yang ditandatangani dalam urutan endian kecil.
2. Gabungkan dua hash dalam urutan yang diurutkan.
3. Hash pasangan gabungan dengan generator hash SHA-256.
4. Pasangkan hash baru Anda dengan hash berikutnya dalam bukti dan ulangi langkah 1-3. Setelah Anda memproses hash bukti terakhir, hash baru Anda adalah intisari yang dihitung ulang.

Jika intisari yang dihitung ulang cocok dengan intisari yang disimpan sebelumnya, dokumen Anda berhasil diverifikasi.

Untuk step-by-step tutorial dengan contoh kode yang menunjukkan langkah-langkah verifikasi ini, lanjutkan keTutorial: Memverifikasi data menggunakan AWS SDK.

# Tutorial: Memverifikasi data menggunakan AWS SDK

Dalam tutorial ini, Anda memverifikasi hash revisi dokumen dan hash blok jurnal di buku besar Amazon QLDB dengan menggunakan QLDB API melalui SDK. AWS Anda juga menggunakan driver QLDB untuk menanyakan revisi dokumen.

Pertimbangkan contoh di mana Anda memiliki revisi dokumen yang berisi data untuk kendaraan dengan nomor identifikasi kendaraan (VIN) dari. `KM8SRDHF6EU074761` Revisi dokumen ada dalam `VehicleRegistration` tabel yang ada di buku besar bernama. `vehicle-registration` Misalkan Anda ingin memverifikasi integritas revisi dokumen untuk kendaraan ini dan blok jurnal yang berisi revisi.

> **Note**
>
> Untuk posting AWS blog terperinci yang membahas nilai verifikasi kriptografi dalam konteks kasus penggunaan yang realistis, lihat Verifikasi kriptografi dunia nyata dengan Amazon QLDB.

Topik



## Prasyarat

Sebelum memulai, pastikan Anda melakukan hal berikut:

1. Siapkan driver QLDB untuk bahasa pilihan Anda dengan menyelesaikan prasyarat masing-masing di bawah. Memulai dengan driver Amazon QLDB Ini termasuk mendaftar AWS, memberikan akses terprogram untuk pengembangan, dan mengonfigurasi lingkungan pengembangan Anda.
2. Ikuti langkah 1-2 Memulai dengan konsol Amazon QLDB untuk membuat buku besar bernama `vehicle-registration` dan memuatnya dengan data sampel yang telah ditentukan.

Selanjutnya, tinjau langkah-langkah berikut untuk mempelajari cara kerja verifikasi, lalu jalankan contoh kode lengkap dari awal hingga akhir.

## Langkah 1: Minta intisari

Sebelum Anda dapat memverifikasi data, Anda harus terlebih dahulu meminta intisari dari buku besar Anda `vehicle-registration` untuk digunakan nanti.

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')
```

# Langkah 2: Kueri revisi dokumen

Gunakan driver QLDB untuk menanyakan alamat blok, hash, dan ID dokumen yang terkait dengan 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)
```

## Langkah 3: Minta bukti untuk revisi

Ulangi hasil kueri dan gunakan setiap alamat blok dan ID dokumen bersama dengan nama buku besar untuk mengirimkan permintaan. `GetRevision` Untuk mendapatkan bukti revisi, Anda juga harus memberikan alamat tip dari intisari yang disimpan sebelumnya. Operasi API ini mengembalikan objek yang menyertakan revisi dokumen dan bukti revisi.

Untuk informasi tentang struktur revisi dan isinya, lihatMelakukan Kueri Metadata Dokumen.

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)
```

Kemudian, ambil bukti untuk revisi yang diminta.

QLDB API mengembalikan bukti sebagai representasi string dari daftar urutan hash node. Untuk mengonversi string ini menjadi daftar representasi biner dari hash node, Anda dapat menggunakan pembaca Ion dari perpustakaan Amazon Ion. Untuk informasi selengkapnya tentang penggunaan pustaka Ion, lihat Buku Masak Amazon Ion.

Java

Dalam contoh ini, Anda gunakan `IonReader` untuk melakukan konversi biner.

```
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

Dalam contoh ini, Anda gunakan `IonLoader` untuk memuat bukti ke dalam datagram Ion.

```
string proofText = revisionResponse.Proof.IonText;
```

```
IIonDatagram proofValue = IonLoader.Default.Load(proofText);
```

Go

Dalam contoh ini, Anda menggunakan pembaca Ion untuk mengonversi bukti menjadi biner dan untuk mengulangi daftar hash simpul bukti.

```
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

Dalam contoh ini, Anda menggunakan `load` fungsi untuk melakukan konversi biner.

```
let proofValue: dom.Value = load(revisionResponse.Proof.IonText);
```

Python

Dalam contoh ini, Anda menggunakan `loads` fungsi untuk melakukan konversi biner.

```
proof_text = proof_response.get('Proof').get('IonText')
proof_hashes = loads(proof_text)
```

## Langkah 4: Hitung ulang intisari dari revisi

Gunakan daftar hash bukti untuk menghitung ulang intisari, dimulai dengan hash revisi. Selama intisari yang disimpan sebelumnya diketahui dan dipercaya di luar QLDB, integritas revisi dokumen terbukti jika hash intisari yang dihitung ulang cocok dengan hash intisari yang disimpan.

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))
```

## Langkah 5: Minta bukti untuk blok jurnal

Selanjutnya, Anda memverifikasi blok jurnal yang berisi revisi dokumen.

Gunakan alamat blokir dan alamat tip dari intisari yang Anda simpan di Langkah 1 untuk mengirimkan `GetBlock` permintaan. Mirip dengan `GetRevision` permintaan di Langkah 2, Anda harus kembali memberikan alamat tip dari intisari yang disimpan untuk mendapatkan bukti untuk blok tersebut. Operasi API ini mengembalikan objek yang menyertakan blok dan bukti untuk blok tersebut.

Untuk informasi tentang struktur blok jurnal dan isinya, lihatIsi jurnal di 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)
```

Kemudian, ambil hash blok dan bukti dari hasilnya.

Java

Dalam contoh ini, Anda gunakan `IonLoader` untuk memuat objek blok ke dalam `IonDatagram` wadah.

```
String blockText = getBlockResult.getBlock().getIonText();

IonDatagram datagram = SYSTEM.getLoader().load(blockText);
ionStruct = (IonStruct)datagram.get(0);

final byte[] blockHash = ((IonBlob)ionStruct.get("blockHash")).getBytes();
```

Anda juga menggunakan `IonLoader` untuk memuat bukti ke dalam`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

Dalam contoh ini, Anda gunakan `IonLoader` untuk memuat blok dan bukti ke dalam datagram Ion untuk masing-masing.

```
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

Dalam contoh ini, Anda menggunakan pembaca Ion untuk mengonversi bukti menjadi biner dan untuk mengulangi daftar hash simpul bukti.

```
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

Dalam contoh ini, Anda menggunakan `load` fungsi untuk mengonversi blok dan bukti menjadi biner.

```
const blockValue: dom.Value = load(getBlockResponse.Block.IonText)
let blockHash: Uint8Array = blockValue.get("blockHash").uInt8ArrayValue();

proofValue = load(getBlockResponse.Proof.IonText);
```

Python

Dalam contoh ini, Anda menggunakan `loads` fungsi untuk mengonversi blok dan bukti menjadi biner.

```
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)
```

## Langkah 6: Hitung ulang intisari dari blok

Gunakan daftar hash bukti untuk menghitung ulang intisari, dimulai dengan hash blok. Selama intisari yang disimpan sebelumnya diketahui dan dipercaya di luar QLDB, integritas blok terbukti jika hash intisari yang dihitung ulang cocok dengan hash intisari yang disimpan.

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))
```

Contoh kode sebelumnya menggunakan `dot` fungsi berikut saat menghitung ulang intisari. Fungsi ini mengambil masukan dari dua hash, mengurutkannya, menggabungkan mereka, dan kemudian mengembalikan hash dari array gabungan.

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
```

# Jalankan contoh kode lengkap

Jalankan contoh kode lengkap sebagai berikut untuk melakukan semua langkah sebelumnya dari awal hingga akhir.

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))
```

# Kesalahan umum untuk verifikasi

Bagian ini menjelaskan kesalahan runtime yang dilemparkan oleh Amazon QLDB untuk permintaan verifikasi.

Berikut ini adalah daftar pengecualian umum yang dikembalikan oleh layanan. Setiap pengecualian mencakup pesan kesalahan tertentu, diikuti oleh operasi API yang dapat membuangnya, deskripsi singkat, dan saran untuk solusi yang mungkin.

IllegalArgumentException

Pesan: Nilai Ion yang disediakan tidak valid dan tidak dapat diuraikan.

Operasi API: `GetDigest, GetBlock, GetRevision`

Pastikan Anda memberikan nilai Amazon Ion yang valid sebelum mencoba kembali permintaan Anda.

IllegalArgumentException

Pesan: Alamat blok yang diberikan tidak valid.

Operasi API: `GetDigest, GetBlock, GetRevision`

Pastikan Anda memberikan alamat blokir yang valid sebelum mencoba kembali permintaan Anda. Alamat blok adalah struktur Amazon Ion yang memiliki dua bidang: `strandId` dan`sequenceNo`.

Misalnya: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`

IllegalArgumentException

Pesan: Nomor urut alamat tip intisari yang diberikan berada di luar catatan komitmen terbaru untai.

Operasi API: `GetDigest, GetBlock, GetRevision`

Alamat tip intisari yang Anda berikan harus memiliki nomor urut kurang dari atau sama dengan nomor urut catatan komitmen terbaru untai jurnal. Sebelum mencoba kembali permintaan Anda, pastikan Anda memberikan alamat tip intisari dengan nomor urut yang valid.

IllegalArgumentException

Pesan: ID Strand dari alamat blok yang diberikan tidak valid.

Operasi API: `GetDigest, GetBlock, GetRevision`

Alamat blok yang Anda berikan harus memiliki ID untai yang cocok dengan ID untai jurnal. Sebelum mencoba kembali permintaan Anda, pastikan Anda memberikan alamat blok dengan ID untai yang valid.

IllegalArgumentException

Pesan: Nomor urut alamat blok yang disediakan berada di luar catatan komitmen terbaru untai.

Operasi API: `GetBlock, GetRevision`

Alamat blok yang Anda berikan harus memiliki nomor urut kurang dari atau sama dengan nomor urut catatan komit terbaru untai. Sebelum mencoba kembali permintaan Anda, pastikan Anda memberikan alamat blok dengan nomor urut yang valid.

IllegalArgumentException

Pesan: ID Strand dari alamat blok yang disediakan harus cocok dengan ID Strand dari alamat tip intisari yang disediakan.

Operasi API: `GetBlock, GetRevision`

Anda hanya dapat memverifikasi revisi atau pemblokiran dokumen jika ada di untai jurnal yang sama dengan intisari yang Anda berikan.

IllegalArgumentException

Pesan: Nomor urut alamat blok yang disediakan tidak boleh lebih besar dari nomor urut alamat tip intisari yang disediakan.

Operasi API: `GetBlock, GetRevision`

Anda hanya dapat memverifikasi revisi atau pemblokiran dokumen jika dilindungi oleh intisari yang Anda berikan. Ini berarti bahwa itu berkomitmen pada jurnal sebelum alamat tip intisari.

IllegalArgumentException

Pesan: ID Dokumen yang disediakan tidak ditemukan di blok di alamat blok yang ditentukan.

Operasi API: `GetRevision`

ID dokumen yang Anda berikan harus ada di alamat blok yang Anda berikan. Sebelum mencoba kembali permintaan Anda, pastikan kedua parameter ini konsisten.

# Mengekspor data jurnal dari Amazon QLDB

Amazon QLDB menggunakan log transaksional yang tidak dapat diubah, yang dikenal sebagai jurnal, untuk penyimpanan data. Jurnal melacak setiap perubahan pada data komitmen Anda dan mempertahankan riwayat perubahan yang lengkap dan dapat diverifikasi dari waktu ke waktu.

Anda dapat mengakses isi jurnal di buku besar Anda untuk berbagai tujuan termasuk analitik, audit, retensi data, verifikasi, dan ekspor ke sistem lain. Topik berikut menjelaskan cara mengekspor blok jurnal dari buku besar Anda ke bucket Amazon Simple Storage Service (Amazon S3) di bucket Anda. Akun AWS Pekerjaan ekspor jurnal menulis data Anda di Amazon S3 sebagai objek baik dalam teks atau representasi biner format Amazon Ion, atau dalam format teks JSON Lines.

Dalam format JSON Lines, setiap blok dalam objek data yang diekspor adalah objek JSON yang valid yang dibatasi oleh baris baru. Anda dapat menggunakan format ini untuk mengintegrasikan ekspor JSON secara langsung dengan alat analitik seperti Amazon Athena AWS Glue dan karena layanan ini dapat mengurai JSON yang dibatasi baris baru secara otomatis. Untuk informasi selengkapnya tentang format, lihat JSON Lines.

Untuk informasi tentang Amazon S3, lihat Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Jika Anda menentukan JSON sebagai format output dari pekerjaan ekspor Anda, QLDB akan mengubah data jurnal Ion ke JSON di objek data yang diekspor. Untuk informasi selengkapnya, lihat Down-convert ke JSON.

Topik

# Meminta ekspor jurnal di QLDB

Amazon QLDB menyediakan API untuk meminta ekspor blok jurnal Anda untuk rentang tanggal dan waktu tertentu serta tujuan bucket Amazon S3 yang ditentukan. Pekerjaan ekspor jurnal dapat menulis objek data baik dalam teks atau representasi biner format Amazon Ion, atau dalam format teks JSON Lines. Anda dapat menggunakan AWS Management Console, AWS SDK, atau AWS Command Line Interface (AWS CLI) untuk membuat pekerjaan ekspor.

Topik



## AWS Management Console

Ikuti langkah-langkah berikut untuk mengirimkan permintaan ekspor jurnal di QLDB menggunakan konsol QLDB.

Untuk meminta ekspor (konsol)

1. Masuk ke AWS Management Console, dan buka konsol QLDB Amazon di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih Ekspor.
3. Pilih Buat pekerjaan ekspor.
4. Pada halaman Create export job, masukkan pengaturan ekspor berikut:
   - Ledger — Buku besar yang jurnalnya memblokir Anda ingin mengekspor.
   - Tanggal dan waktu mulai — Stempel waktu mulai inklusif di Coordinated Universal Time (UTC) dari rentang blok jurnal yang akan diekspor. Stempel waktu ini harus lebih awal dari tanggal dan waktu Akhir. Jika Anda memberikan stempel waktu awal yang lebih awal dari buku besar, `CreationDateTime` QLDB mendefaultkannya ke buku besar. `CreationDateTime`
   - Tanggal dan waktu akhir - Stempel waktu akhir eksklusif (UTC) dari rentang blok jurnal yang akan diekspor. Tanggal dan waktu ini tidak mungkin di masa depan.
   - Tujuan untuk blok jurnal — Bucket Amazon S3 dan nama awalan tempat pekerjaan ekspor Anda menulis objek data. Gunakan format URI Amazon S3 berikut.

```
s3://DOC-EXAMPLE-BUCKET/prefix/
```

Anda harus menentukan nama bucket S3 dan nama awalan opsional untuk objek keluaran. Berikut adalah contohnya.

```
s3://DOC-EXAMPLE-BUCKET/journalExport/
```

Nama bucket dan awalan harus mematuhi aturan dan konvensi penamaan Amazon S3. Untuk informasi selengkapnya tentang penamaan bucket, lihat Pembatasan dan batasan bucket di Panduan Pengembang Amazon S3. Untuk informasi selengkapnya tentang awalan nama kunci, lihat Kunci objek dan metadata.

> **Note**
>
> Ekspor lintas wilayah tidak didukung. Bucket Amazon S3 yang ditentukan harus Wilayah AWS sama dengan buku besar Anda.

- Enkripsi S3 - Pengaturan enkripsi yang digunakan oleh pekerjaan ekspor Anda untuk menulis data dalam bucket Amazon S3. Untuk mempelajari lebih lanjut tentang opsi enkripsi sisi server di Amazon S3, lihat Melindungi data menggunakan enkripsi sisi server di Panduan Pengembang Amazon S3.
  - Enkripsi default bucket — Gunakan pengaturan enkripsi default bucket Amazon S3 yang ditentukan.
  - AES-256 - Gunakan enkripsi sisi server dengan kunci terkelola Amazon S3 (SSE-S3).
  - AWS-KMS — Gunakan enkripsi sisi server dengan AWS KMS kunci terkelola (SSE-KMS).

    Jika Anda memilih jenis ini bersama dengan AWS KMS key opsi Pilih yang berbeda, Anda juga harus menentukan kunci KMS enkripsi simetris dalam format Amazon Resource Name (ARN) berikut.

    ```
    arn:aws:kms:aws-region:account-id:key/key-id
    ```
- Akses layanan — Peran IAM yang memberikan izin menulis QLDB di bucket Amazon S3 Anda. Jika berlaku, peran IAM juga harus memberikan izin QLDB untuk menggunakan kunci KMS Anda.

Untuk meneruskan peran ke QLDB saat meminta ekspor jurnal, Anda harus memiliki izin untuk melakukan *iam:PassRole* tindakan pada sumber daya peran IAM.

- Buat dan gunakan peran layanan baru — Biarkan konsol membuat peran baru untuk Anda dengan izin yang diperlukan untuk bucket Amazon S3 yang ditentukan.
- Gunakan peran layanan yang ada — Untuk mempelajari cara membuat peran ini secara manual di IAM, lihatIzin ekspor.

- Format output - Format output dari data jurnal yang diekspor
  - Teks ion - (Default) Representasi teks Amazon Ion
  - Biner ion — Representasi biner dari Amazon Ion
  - JSON - Format teks JSON yang dibatasi Newline

    Jika Anda memilih JSON, QLDB menurunkan data jurnal Ion menjadi JSON di objek data yang diekspor. Untuk informasi selengkapnya, lihat Down-convert ke JSON.

5. Ketika pengaturan seperti yang Anda inginkan, pilih Buat pekerjaan ekspor.

   Jumlah waktu yang dibutuhkan untuk menyelesaikan pekerjaan ekspor Anda bervariasi tergantung pada ukuran data. Jika pengiriman permintaan Anda berhasil, konsol akan kembali ke halaman Ekspor utama dan mencantumkan pekerjaan ekspor Anda dengan statusnya saat ini.

6. Anda dapat melihat objek ekspor Anda di konsol Amazon S3.

   Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.

   Untuk mempelajari lebih lanjut tentang format objek keluaran ini, lihatOutput ekspor jurnal dalam QLDB.

> **Note**
>
> Pekerjaan ekspor berakhir tujuh hari setelah selesai. Untuk informasi selengkapnya, lihat Kedaluwarsa pekerjaan ekspor.

## QLDB API

Anda juga dapat meminta ekspor jurnal dengan menggunakan Amazon QLDB API dengan SDK AWS atau file. AWS CLI QLDB API menyediakan operasi berikut untuk digunakan oleh program aplikasi:

- `ExportJournalToS3`— Mengekspor konten jurnal dalam rentang tanggal dan waktu dari buku besar tertentu ke ember Amazon S3 tertentu. Pekerjaan ekspor dapat menulis data sebagai objek baik dalam teks atau representasi biner format Amazon Ion, atau dalam format teks JSON Lines.
- `DescribeJournalS3Export`— Mengembalikan informasi rinci tentang pekerjaan ekspor jurnal. Outputnya mencakup statusnya saat ini, waktu pembuatan, dan parameter permintaan ekspor asli Anda.
- `ListJournalS3Exports`— Mengembalikan daftar deskripsi pekerjaan ekspor jurnal untuk semua buku besar yang terkait dengan saat ini Akun AWS dan Wilayah. Output dari setiap deskripsi pekerjaan ekspor mencakup detail yang sama yang dikembalikan oleh`DescribeJournalS3Export`.
- `ListJournalS3ExportsForLedger`— Mengembalikan daftar deskripsi pekerjaan ekspor jurnal untuk buku besar yang diberikan. Output dari setiap deskripsi pekerjaan ekspor mencakup detail yang sama yang dikembalikan oleh`DescribeJournalS3Export`.

Untuk deskripsi lengkap tentang operasi API ini, lihat. Referensi API Amazon QLDB

Untuk informasi tentang mengekspor data jurnal menggunakan AWS CLI, lihat Referensi AWS CLI Perintah.

### Contoh aplikasi (Java)

Untuk contoh kode Java dari operasi ekspor dasar, lihat GitHub repositori amazon-qldb-dmv-sampleaws-samples/ -java. Untuk petunjuk tentang cara mengunduh dan menginstal aplikasi sampel ini, lihatMenginstal aplikasi sampel Amazon QLDB Java. Sebelum meminta ekspor, pastikan Anda mengikuti Langkah 1-3 tutorial java untuk membuat buku besar sampel dan memuatnya dengan data sampel.

Kode tutorial di kelas berikut memberikan contoh pembuatan ekspor, memeriksa status ekspor, dan memproses output ekspor.

| Kelas | Deskripsi |
| --- | --- |
| ExportJournal | Mengekspor blok jurnal dari buku besar `vehicle-registration` sampel untuk rentang waktu 10 menit yang lalu hingga sekarang. Menulis objek keluaran dalam bucket S3 tertentu, atau buat bucket unik jika tidak disediakan. |
| DescribeJournalExport | Menjelaskan pekerjaan ekspor jurnal untuk yang ditentukan `exportId` dalam buku besar `vehicle-registration` sampel. |
| ListJournalExports | Mengembalikan daftar deskripsi pekerjaan ekspor jurnal untuk buku besar `vehicle-registration` sampel. |
| ValidateQldbHashChain | Memvalidasi rantai hash dari buku besar `vehicle-registration` sampel menggunakan yang diberikan. `exportId` Jika tidak disediakan, minta ekspor baru untuk digunakan untuk validasi rantai hash. |

## Kedaluwarsa pekerjaan ekspor

Pekerjaan ekspor jurnal yang diselesaikan tunduk pada periode retensi 7 hari. Mereka secara otomatis dihapus setelah batas ini kedaluwarsa. Periode kedaluwarsa ini adalah batas yang sulit dan tidak dapat diubah.

Setelah tugas ekspor selesai dihapus, Anda tidak dapat lagi menggunakan konsol QLDB atau operasi API berikut untuk mengambil metadata tentang pekerjaan:

- `DescribeJournalS3Export`
- `ListJournalS3Exports`
- `ListJournalS3ExportsForLedger`

Namun, kedaluwarsa ini tidak berdampak pada data yang diekspor itu sendiri. Semua metadata disimpan dalam file manifes yang ditulis oleh ekspor Anda. Kedaluwarsa ini dirancang untuk memberikan pengalaman yang lebih lancar bagi operasi API yang mencantumkan pekerjaan ekspor jurnal. QLDB menghapus pekerjaan ekspor lama untuk memastikan bahwa Anda hanya melihat ekspor terbaru tanpa harus mengurai beberapa halaman pekerjaan.

## Output ekspor jurnal dalam QLDB

Pekerjaan ekspor jurnal QLDB Amazon menulis dua file manifes selain objek data yang berisi blok jurnal Anda. Ini semua disimpan di bucket Amazon S3 yang Anda berikan dalam permintaan ekspor Anda. Bagian berikut menjelaskan format dan isi dari setiap objek output.

> **Note**
>
> Jika Anda menentukan JSON sebagai format output dari pekerjaan ekspor Anda, QLDB akan mengubah data jurnal Amazon Ion ke JSON di objek data yang diekspor. Untuk informasi lebih lanjut, lanjutkan keDown-convert ke JSON.

Topik

- File manifes
- Obyek data
- Down-convert ke JSON
- Ekspor pustaka prosesor (Java)

## File manifes

Amazon QLDB membuat dua file manifes dalam bucket S3 yang disediakan untuk setiap permintaan ekspor. File manifes awal dibuat segera setelah Anda mengirimkan permintaan ekspor. File manifes akhir ditulis setelah ekspor selesai. Anda dapat menggunakan file-file ini untuk memeriksa status pekerjaan ekspor Anda di Amazon S3.

Format untuk isi file manifes sesuai dengan format output yang diminta untuk ekspor.

## Manifes awal

Manifes awal menunjukkan bahwa pekerjaan ekspor Anda telah dimulai. Ini berisi parameter input yang Anda berikan ke permintaan. Selain tujuan Amazon S3 dan parameter waktu mulai dan berakhir untuk ekspor, file ini juga berisi file. `exportId` `exportId`Ini adalah ID unik yang diberikan QLDB untuk setiap pekerjaan ekspor.

Konvensi penamaan file adalah sebagai berikut.

```
s3://DOC-EXAMPLE-BUCKET/prefix/exportId.started.manifest
```

Berikut ini adalah contoh file manifes awal dan isinya dalam format teks 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"
}
```

Manifes awal mencakup `outputFormat` satu-satunya jika ditentukan dalam permintaan ekspor. Jika Anda tidak menentukan format output, data yang diekspor default ke format. `ION_TEXT`

Operasi DescribeJournalS3Export API dan jenis konten objek Amazon S3 yang diekspor juga menunjukkan format output.

## Manifes akhir

Manifes terakhir menunjukkan bahwa pekerjaan ekspor Anda untuk untaian jurnal tertentu telah selesai. Pekerjaan ekspor menulis file manifes akhir terpisah untuk setiap untai.

> **Note**
>
> Di Amazon QLDB, untai adalah partisi jurnal buku besar Anda. QLDB saat ini mendukung jurnal dengan untai tunggal saja.

Manifes akhir mencakup daftar urutan kunci objek data yang ditulis selama ekspor. Konvensi penamaan file adalah sebagai berikut.

```
s3://DOC-EXAMPLE-BUCKET/prefix/exportId.strandId.completed.manifest
```

`strandId`Ini adalah ID unik yang diberikan QLDB ke untai. Berikut ini adalah contoh file manifes akhir dan isinya dalam format teks 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"
  ]
}
```

## Obyek data

Amazon QLDB menulis objek data jurnal dalam bucket Amazon S3 yang disediakan baik dalam teks atau representasi biner format Amazon Ion, atau dalam format teks JSON Lines.

Dalam format JSON Lines, setiap blok dalam objek data yang diekspor adalah objek JSON yang valid yang dibatasi oleh baris baru. Anda dapat menggunakan format ini untuk mengintegrasikan ekspor JSON secara langsung dengan alat analitik seperti Amazon Athena AWS Glue dan karena layanan ini dapat mengurai JSON yang dibatasi baris baru secara otomatis. Untuk informasi selengkapnya tentang format, lihat JSON Lines.

## Nama objek data

Pekerjaan ekspor jurnal menulis objek data ini dengan konvensi penamaan berikut.

```
s3://DOC-EXAMPLE-BUCKET/prefix/yyyy/mm/dd/hh/strandId.startSn-endSn.ion|.json
```

- Data output dari setiap pekerjaan ekspor dipecah menjadi beberapa bagian.
- `yyyy/mm/dd/hh`— Tanggal dan waktu ketika Anda mengajukan permintaan ekspor. Objek yang diekspor dalam jam yang sama dikelompokkan di bawah awalan Amazon S3 yang sama.
- `strandId`— ID unik dari untai tertentu yang berisi blok jurnal yang sedang diekspor.
- `startSn-endSn`— Rentang nomor urut yang termasuk dalam objek. Nomor urut menentukan lokasi blok dalam untai.

Misalnya, anggaplah Anda menentukan jalur berikut.

```
s3://DOC-EXAMPLE-BUCKET/journalExport/
```

Pekerjaan ekspor Anda membuat objek data Amazon S3 yang terlihat mirip dengan yang berikut ini. Contoh ini menunjukkan nama objek dalam format Ion.

```
s3://DOC-EXAMPLE-BUCKET/journalExport/2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.1-5.ion
```

## Isi objek data

Setiap objek data berisi objek blok jurnal dengan format berikut.

```
{
  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
    }
  ]
}
```

Blok adalah objek yang berkomitmen pada jurnal selama transaksi. Sebuah blok berisi metadata transaksi bersama dengan entri yang mewakili revisi dokumen yang dilakukan dalam transaksi dan pernyataan PartiQL yang melakukannya.

Berikut ini adalah contoh blok dengan data sampel dalam format teks Ion. Untuk informasi tentang bidang dalam objek blok, lihatIsi jurnal di Amazon QLDB.

> **Note**
>
> Contoh blok ini disediakan hanya untuk tujuan informasi. Hash yang ditampilkan bukanlah nilai hash yang dihitung secara nyata.

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

Di `revisions` lapangan, beberapa objek revisi mungkin hanya berisi `hash` nilai dan tidak ada atribut lainnya. Ini adalah revisi sistem internal saja yang tidak berisi data pengguna. Pekerjaan ekspor mencakup revisi ini di blok masing-masing karena hash dari revisi ini adalah bagian dari rantai hash penuh jurnal. Rantai hash penuh diperlukan untuk verifikasi kriptografi.

## Down-convert ke JSON

Jika Anda menentukan JSON sebagai format output dari pekerjaan ekspor Anda, QLDB akan mengubah data jurnal Amazon Ion ke JSON di objek data yang diekspor. Namun, mengonversi Ion ke JSON hilang dalam kasus tertentu di mana data Anda menggunakan tipe Ion kaya yang tidak ada di JSON.

Untuk detail tentang aturan konversi Ion ke JSON, lihat Mengonversi ke bawah ke JSON di Buku Masak Amazon Ion.

## Ekspor pustaka prosesor (Java)

QLDB menyediakan kerangka kerja yang dapat diperluas untuk Java yang merampingkan pemrosesan ekspor di Amazon S3. Pustaka kerangka kerja ini menangani pekerjaan membaca

output ekspor dan iterasi melalui blok yang diekspor dalam urutan berurutan. Untuk menggunakan prosesor ekspor ini, lihat GitHub repositori amazon-qldb-export-processorawslabs/ -java.

## Izin ekspor jurnal di QLDB

Sebelum mengirimkan permintaan ekspor jurnal di Amazon QLDB, Anda harus memberikan izin menulis kepada QLDB di bucket Amazon S3 yang Anda tentukan. Jika Anda memilih pelanggan yang dikelola AWS KMS key sebagai jenis enkripsi objek untuk bucket Amazon S3, Anda juga harus memberikan izin kepada QLDB untuk menggunakan kunci enkripsi simetris yang ditentukan. Amazon S3 tidak mendukung kunci KMS asimetris.

Untuk memberikan izin yang diperlukan kepada pekerjaan ekspor Anda, Anda dapat membuat QLDB mengambil peran layanan IAM dengan kebijakan izin yang sesuai. Peran layanan adalah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Note**
>
> Untuk meneruskan peran ke QLDB saat meminta ekspor jurnal, Anda harus memiliki izin untuk melakukan `iam:PassRole` tindakan pada sumber daya peran IAM. Ini selain `qldb:ExportJournalToS3` izin pada sumber daya buku besar QLDB.
> Untuk mempelajari cara mengontrol akses ke QLDB menggunakan IAM, lihat. Bagaimana Amazon QLDB bekerja dengan IAM Untuk contoh kebijakan QLDB, lihat. Contoh kebijakan berbasis identitas untuk Amazon QLDB

Dalam contoh ini, Anda membuat peran yang memungkinkan QLDB menulis objek ke dalam bucket Amazon S3 atas nama Anda. Untuk informasi selengkapnya, lihat Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

Jika Anda mengekspor jurnal QLDB di jurnal Akun AWS Anda untuk pertama kalinya, Anda harus terlebih dahulu membuat peran IAM dengan kebijakan yang sesuai dengan melakukan hal berikut. Atau, Anda dapat menggunakan konsol QLDB untuk secara otomatis membuat peran untuk Anda. Jika tidak, Anda dapat memilih peran yang sebelumnya Anda buat.

Topik

- Membuat kebijakan izin
- Membuat peran IAM

## Membuat kebijakan izin

Selesaikan langkah-langkah berikut untuk membuat kebijakan izin untuk pekerjaan ekspor jurnal QLDB. Contoh ini menunjukkan kebijakan bucket Amazon S3 yang memberikan izin QLDB untuk menulis objek ke dalam bucket yang Anda tentukan. Jika berlaku, contoh ini juga menunjukkan kebijakan kunci yang memungkinkan QLDB menggunakan kunci KMS enkripsi simetris Anda.

Untuk informasi selengkapnya tentang kebijakan bucket Amazon S3, lihat Menggunakan kebijakan bucket dan kebijakan pengguna di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Untuk mempelajari lebih lanjut tentang kebijakan AWS KMS utama, lihat Menggunakan kebijakan utama AWS KMS di Panduan AWS Key Management Service Pengembang.

> **Note**
>
> Bucket Amazon S3 dan kunci KMS Anda harus sama dengan buku besar QLDB Wilayah AWS Anda.

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di kolom navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul laman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.
4. Pilih tab JSON.
5. Masukkan dokumen kebijakan JSON.
   - Jika Anda menggunakan kunci KMS yang dikelola pelanggan untuk enkripsi objek Amazon S3, gunakan contoh dokumen kebijakan berikut. *Untuk menggunakan kebijakan ini, ganti DOC-EXAMPLE-BUCKET, us-east-1, 123456789012, dan 1234abcd-12ab-34cd-56ef-1234567890ab dalam contoh dengan informasi Anda sendiri.*

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

- Untuk jenis enkripsi lainnya, gunakan contoh dokumen kebijakan berikut. Untuk menggunakan kebijakan ini, ganti *DOC-EXAMPLE-BUCKET dalam contoh dengan nama bucket* Amazon S3 Anda sendiri.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permission",
            "Action": [
                "s3:PutObjectAcl",
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

6. Pilih Tinjau kebijakan.

> **Note**
>
> Anda dapat berpindah antara tab Editor visual dan JSON kapan pun. Namun, apabila Anda melakukan perubahan atau memilih Tinjau kebijakan pada tab Editor visual, IAM dapat merestrukturisasi kebijakan Anda untuk menjadikannya optimal bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. Pada halaman Peninjauan Kebijakan, ketikkan Nama dan Deskripsi opsional untuk kebijakan yang sedang Anda buat. Tinjau Summary (Ringkasan) kebijakan untuk melihat izin yang diberikan oleh kebijakan Anda. Kemudian pilih Buat kebijakan untuk menyimpan pekerjaan Anda.

## Membuat peran IAM

Setelah membuat kebijakan izin untuk pekerjaan ekspor jurnal QLDB, Anda kemudian dapat membuat peran IAM dan melampirkan kebijakan Anda padanya.

Untuk membuat peran layanan untuk QLDB (konsol IAM)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran, dan lalu pilih Buat peran.
3. Untuk jenis entitas Tepercaya, pilih Layanan AWS.
4. Untuk kasus Layanan atau penggunaan, pilih QLDB, lalu pilih kasus penggunaan QLDB.
5. Pilih Selanjutnya.
6. Pilih kotak di samping kebijakan yang Anda buat di langkah sebelumnya.
7. (Opsional) Tetapkan batas izin. Ini adalah fitur lanjutan yang tersedia untuk peran layanan, tetapi bukan peran tertaut layanan.

    a. Buka bagian Setel batas izin, lalu pilih Gunakan batas izin untuk mengontrol izin peran maksimum.

    IAM menyertakan daftar kebijakan yang AWS dikelola dan dikelola pelanggan di akun Anda.

    b. Pilih kebijakan yang akan digunakan untuk batas izin.

8. Pilih Selanjutnya.
9. Masukkan nama peran atau akhiran nama peran untuk membantu Anda mengidentifikasi tujuan peran.

> **Important**
>
> Saat Anda memberi nama peran, perhatikan hal berikut:
>
> - Nama peran harus unik di dalam diri Anda Akun AWS, dan tidak dapat dibuat unik berdasarkan kasus.
>
>  Misalnya, jangan membuat peran bernama keduanya **`PRODROLE`** dan**`prodrole`**. Ketika nama peran digunakan dalam kebijakan atau sebagai bagian dari ARN, nama peran tersebut peka huruf besar/kecil, namun ketika nama peran muncul kepada pelanggan di konsol, seperti selama proses masuk, nama peran tersebut tidak peka huruf besar/kecil.
> - Anda tidak dapat mengedit nama peran setelah dibuat karena entitas lain mungkin mereferensikan peran tersebut.

10. (Opsional) Untuk Deskripsi, masukkan deskripsi untuk peran tersebut.
11. (Opsional) Untuk mengedit kasus penggunaan dan izin untuk peran, di Langkah 1: Pilih entitas tepercaya atau Langkah 2: Tambahkan izin, pilih Edit.
12. (Opsional) Untuk membantu mengidentifikasi, mengatur, atau mencari peran, tambahkan tag sebagai pasangan nilai kunci. Untuk informasi selengkapnya tentang penggunaan tanda di IAM, lihat Menandai sumber daya IAM di Panduan Pengguna IAM.
13. Tinjau peran lalu pilih Buat peran.

Dokumen JSON berikut adalah contoh kebijakan kepercayaan yang memungkinkan QLDB untuk mengambil peran IAM dengan izin khusus yang melekat padanya.

```
{
    "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**
>
> Contoh kebijakan kepercayaan ini menunjukkan bagaimana Anda dapat menggunakan kunci konteks kondisi `aws:SourceAccount` global `aws:SourceArn` dan global untuk mencegah masalah wakil yang membingungkan. Dengan kebijakan kepercayaan ini, QLDB dapat mengambil peran untuk sumber daya QLDB apa pun di akun saja. `123456789012` Untuk informasi selengkapnya, lihat Pencegahan confused deputy lintas layanan.

Setelah membuat peran IAM Anda, kembali ke konsol QLDB dan segarkan halaman pekerjaan Buat ekspor sehingga dapat menemukan peran baru Anda.

## Kesalahan umum untuk ekspor jurnal

Bagian ini menjelaskan kesalahan runtime yang dilemparkan oleh Amazon QLDB untuk permintaan ekspor jurnal.

Berikut ini adalah daftar pengecualian umum yang dikembalikan oleh layanan. Setiap pengecualian mencakup pesan kesalahan tertentu, diikuti dengan deskripsi singkat dan saran untuk solusi yang mungkin.

AccessDeniedException

*Pesan: Pengguna: UserARN tidak berwenang untuk melakukan: iam: PassRole on resource: roLearn*

Anda tidak memiliki izin untuk meneruskan peran IAM ke layanan QLDB. QLDB memerlukan peran untuk semua permintaan ekspor jurnal, dan Anda harus memiliki izin untuk meneruskan peran ini ke QLDB. Peran ini memberi QLDB izin menulis di bucket Amazon S3 yang Anda tentukan.

Verifikasi bahwa Anda menentukan kebijakan IAM yang memberikan izin untuk menjalankan operasi `PassRole` API pada sumber daya peran IAM yang ditentukan untuk layanan QLDB (). `qldb.amazonaws.com` Untuk contoh kebijakan, lihatContoh kebijakan berbasis identitas untuk Amazon QLDB.

IllegalArgumentException

*Pesan: QLDB mengalami kesalahan memvalidasi konfigurasi S3: ErrorCode ErrorMessage*

Kemungkinan penyebab kesalahan ini adalah bucket Amazon S3 yang disediakan tidak ada di Amazon S3. Atau, QLDB tidak memiliki izin yang cukup untuk menulis objek ke dalam bucket Amazon S3 yang Anda tentukan.

Verifikasi bahwa nama bucket S3 yang Anda berikan dalam permintaan pekerjaan ekspor sudah benar. Untuk informasi selengkapnya tentang penamaan bucket, lihat Pembatasan dan batasan bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Selain itu, verifikasi bahwa Anda menentukan kebijakan untuk bucket yang ditentukan yang memberikan `PutObject` dan `PutObjectAcl` izin ke layanan QLDB (). `qldb.amazonaws.com` Untuk mempelajari selengkapnya, lihat Izin ekspor.

IllegalArgumentException

Pesan: Respons tak terduga dari Amazon S3 saat memvalidasi konfigurasi S3. *Tanggapan dari S3: ErrorCode ErrorMessage*

Upaya untuk menulis data ekspor jurnal ke dalam bucket S3 yang disediakan gagal dengan respons kesalahan Amazon S3 yang disediakan. Untuk informasi selengkapnya tentang kemungkinan penyebab, lihat Memecahkan Masalah Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

IllegalArgumentException

Pesan: Awalan bucket Amazon S3 tidak boleh melebihi 128 karakter

Awalan yang disediakan dalam permintaan ekspor jurnal berisi lebih dari 128 karakter.

IllegalArgumentException

Pesan: Tanggal mulai tidak boleh lebih besar dari tanggal akhir

Keduanya `InclusiveStartTime` dan `ExclusiveEndTime` harus dalam format tanggal dan waktu ISO 8601 dan dalam Coordinated Universal Time (UTC).

IllegalArgumentException

Pesan: Tanggal akhir tidak bisa di masa depan

Keduanya `InclusiveStartTime` dan `ExclusiveEndTime` harus dalam format `ISO 8601` tanggal dan waktu dan dalam UTC.

IllegalArgumentException

Pesan: Pengaturan enkripsi objek yang disediakan (S3EncryptionConfiguration) tidak kompatibel dengan kunci AWS Key Management Service (AWS KMS)

Anda menyediakan `KMSKeyArn` dengan `ObjectEncryptionType` salah satu `NO_ENCRYPTION` atau`SSE_S3`. Anda hanya dapat menyediakan pelanggan yang dikelola AWS KMS key untuk jenis enkripsi objek`SSE_KMS`. Untuk mempelajari lebih lanjut tentang opsi enkripsi sisi server di Amazon S3, lihat Melindungi data menggunakan enkripsi sisi server di Panduan Pengembang Amazon S3.

LimitExceededException

Pesan: Melebihi batas 2 pekerjaan ekspor Jurnal yang berjalan secara bersamaan

QLDB memberlakukan batas default dari dua pekerjaan ekspor jurnal bersamaan.

# Streaming data jurnal dari Amazon QLDB

Amazon QLDB menggunakan log transaksional yang tidak dapat diubah, yang dikenal sebagai jurnal, untuk penyimpanan data. Jurnal melacak setiap perubahan pada data komitmen Anda dan mempertahankan riwayat perubahan yang lengkap dan dapat diverifikasi dari waktu ke waktu.

Anda dapat membuat aliran di QLDB yang menangkap setiap revisi dokumen yang berkomitmen pada jurnal Anda dan mengirimkan data ini ke Amazon Kinesis Data Streams secara nyaris real time. Aliran QLDB adalah aliran data yang berkelanjutan dari jurnal buku besar Anda ke sumber daya aliran data Kinesis.

Kemudian, Anda menggunakan platform streaming Kinesis atau Perpustakaan Klien Kinesis untuk menggunakan aliran Anda, memproses catatan data, dan menganalisis konten data. Aliran QLDB menulis data Anda ke Kinesis Data Streams dalam tiga jenis catatan: kontrol, ringkasan blok, dan detail revisi. Untuk informasi selengkapnya, lihat Catatan aliran QLDB dalam Kinesis.

Topik



## Kasus penggunaan umum

Streaming memungkinkan Anda menggunakan QLDB sebagai sumber kebenaran tunggal yang dapat diverifikasi sambil mengintegrasikan data jurnal Anda dengan layanan lain. Berikut ini adalah beberapa kasus penggunaan umum yang didukung oleh aliran jurnal QLDB:

- Arsitektur berbasis peristiwa — Bangun aplikasi dalam gaya arsitektur berbasis peristiwa dengan komponen terpisah. Misalnya, bank dapat menggunakan AWS Lambda fungsi untuk menerapkan sistem notifikasi yang memberi tahu pelanggan ketika saldo akun mereka turun di bawah ambang batas. Dalam sistem seperti itu, saldo akun dipertahankan dalam buku besar QLDB, dan setiap perubahan saldo dicatat dalam jurnal. AWS Lambda Fungsi ini dapat memicu logika notifikasi setelah mengkonsumsi peristiwa pembaruan saldo yang berkomitmen ke jurnal dan dikirim ke aliran data Kinesis.
- Analitik real-time — Membangun aplikasi konsumen Kinesis yang menjalankan analisis real-time pada data peristiwa. Dengan kemampuan ini, Anda dapat memperoleh wawasan dalam waktu nyaris nyata dan merespons dengan cepat terhadap lingkungan bisnis yang berubah. Misalnya, situs web e-niaga dapat menganalisis data penjualan produk dan menghentikan iklan untuk produk diskon segera setelah penjualan mencapai batas.
- Analisis historis — Manfaatkan arsitektur berorientasi jurnal Amazon QLDB dengan memutar ulang data peristiwa historis. Anda dapat memilih untuk memulai aliran QLDB pada setiap titik waktu di masa lalu, di mana semua revisi sejak saat itu dikirim ke Kinesis Data Streams. Dengan menggunakan fitur ini, Anda dapat membangun aplikasi konsumen Kinesis yang menjalankan pekerjaan analitik pada data historis. Misalnya, situs web e-niaga dapat menjalankan analitik sesuai kebutuhan untuk menghasilkan metrik penjualan sebelumnya yang sebelumnya tidak ditangkap.
- Replikasi ke database yang dibuat khusus - Hubungkan buku besar QLDB ke penyimpanan data lain yang dibuat khusus menggunakan aliran jurnal QLDB. Misalnya, gunakan platform data streaming Kinesis untuk berintegrasi dengan Amazon OpenSearch Service, yang dapat menyediakan kemampuan pencarian teks lengkap untuk dokumen QLDB. Anda juga dapat membuat aplikasi konsumen Kinesis khusus untuk mereplikasi data jurnal Anda ke database lain yang dibuat khusus yang memberikan tampilan terwujud yang berbeda. Misalnya, replikasi ke Amazon Aurora untuk data relasional atau ke Amazon Neptunus untuk data berbasis grafik.

## Mengkonsumsi streaming Anda

Gunakan Kinesis Data Streams untuk terus mengkonsumsi, memproses, dan menganalisis aliran besar catatan data. Selain Kinesis Data Streams, platform data streaming Kinesis mencakup Amazon Data Firehose dan Amazon Managed Service untuk Apache Flink. Anda dapat menggunakan platform ini untuk mengirim catatan data langsung ke layanan seperti Amazon OpenSearch Service, Amazon Redshift, Amazon S3, atau Splunk. Untuk informasi selengkapnya, lihat Konsumen Kinesis Data Streams di Panduan Pengembang Amazon Kinesis Data Streams.

Anda juga dapat menggunakan Kinesis Client Library (KCL) untuk membangun aplikasi konsumen streaming untuk memproses catatan data dengan cara khusus. KCL menyederhanakan pengodean dengan menyediakan abstraksi yang berguna di atas Kinesis Data Streams API tingkat rendah. Untuk mempelajari lebih lanjut tentang KCL, lihat Menggunakan Perpustakaan Klien Kinesis di Panduan Pengembang Amazon Kinesis Data Streams.

## Jaminan pengiriman

Aliran QLDB memberikan jaminan pengiriman. at-least-once Setiap catatan data yang dihasilkan oleh aliran QLDB dikirim ke Kinesis Data Streams setidaknya sekali. Catatan yang sama dapat muncul dalam aliran data Kinesis beberapa kali. Jadi, Anda harus memiliki logika deduplikasi di lapisan aplikasi konsumen jika kasus penggunaan Anda memerlukannya.

Juga tidak ada jaminan pemesanan. Dalam beberapa keadaan, blok dan revisi QLDB dapat diproduksi dalam aliran data Kinesis yang rusak. Untuk informasi selengkapnya, lihat Menangani duplikat dan catatan out-of-order.

## Pertimbangan latensi pengiriman

Aliran QLDB biasanya mengirimkan pembaruan ke Kinesis Data Streams dalam waktu nyaris nyata. Namun, skenario berikut mungkin menciptakan latensi tambahan sebelum data QLDB yang baru berkomitmen dipancarkan ke aliran data Kinesis:

- Kinesis dapat membatasi data yang dialirkan dari QLDB, tergantung pada penyediaan Kinesis Data Streams Anda. Misalnya, ini mungkin terjadi jika Anda memiliki beberapa aliran QLDB yang menulis ke satu aliran data Kinesis, dan tingkat permintaan QLDB melebihi kapasitas sumber daya aliran Kinesis. Pelambatan pada Kinesis juga dapat terjadi saat menggunakan penyediaan sesuai permintaan jika throughput tumbuh lebih dari dua kali lipat puncak sebelumnya dalam waktu kurang dari 15 menit.

  Anda dapat mengukur throughput yang terlampaui ini dengan memantau metrik Kinesis. `WriteProvisionedThroughputExceeded` Untuk informasi selengkapnya dan solusi yang mungkin, lihat Bagaimana cara memecahkan masalah error throttling di Kinesis Data Streams? .
- Dengan aliran QLDB, Anda dapat membuat aliran tidak terbatas dengan tanggal dan waktu mulai di masa lalu dan tanpa tanggal dan waktu akhir. Secara desain, QLDB mulai memancarkan data yang baru berkomitmen ke Kinesis Data Streams hanya setelah semua data sebelumnya dari tanggal dan waktu mulai yang ditentukan berhasil dikirim. Jika Anda merasakan latensi tambahan

dalam skenario ini, Anda mungkin perlu menunggu data sebelumnya dikirimkan, atau Anda dapat memulai streaming dari tanggal dan waktu mulai nanti.

## Memulai dengan aliran

Berikut ini adalah ikhtisar tingkat tinggi dari langkah-langkah yang diperlukan untuk memulai streaming data jurnal ke Kinesis Data Streams:

1. Buat sumber daya Kinesis Data Streams. Untuk petunjuknya, lihat Membuat dan memperbarui aliran data di Panduan Pengembang Amazon Kinesis Data Streams.
2. Buat peran IAM yang memungkinkan QLDB untuk mengasumsikan izin menulis untuk aliran data Kinesis. Untuk petunjuk, lihat Izin streaming di QLDB.
3. Buat aliran jurnal QLDB. Untuk petunjuk, lihat Membuat dan mengelola aliran di QLDB.
4. Konsumsi aliran data Kinesis, seperti yang dijelaskan di bagian sebelumnya. Mengkonsumsi streaming Anda Untuk contoh kode yang menunjukkan cara menggunakan Kinesis Client Library atau AWS Lambda, lihat. Berkembang dengan aliran di QLDB

## Membuat dan mengelola aliran di QLDB

Amazon QLDB menyediakan operasi API untuk membuat dan mengelola aliran data jurnal dari buku besar Anda ke Amazon Kinesis Data Streams. Aliran QLDB menangkap setiap revisi dokumen yang berkomitmen pada jurnal Anda dan mengirimkannya ke aliran data Kinesis.

Anda dapat menggunakan AWS Management Console, AWS SDK, atau AWS Command Line Interface (AWS CLI) untuk membuat aliran jurnal. Selain itu, Anda juga dapat menggunakan AWS CloudFormationtemplate untuk membuat aliran. Untuk informasi selengkapnya, lihat AWS::QLDB::Streamsumber daya di Panduan AWS CloudFormation Pengguna.

Topik

- Parameter aliran
- ARN Streaming
- AWS Management Console
- Negara aliran
- Menangani aliran yang terganggu

# Parameter aliran

Untuk membuat aliran jurnal QLDB, Anda harus memberikan parameter konfigurasi berikut:

Nama buku besar

Buku besar QLDB yang data jurnalnya ingin Anda streaming ke Kinesis Data Streams.

Nama aliran

Nama yang ingin Anda tetapkan ke pengaliran jurnal QLDB. Nama yang ditentukan pengguna dapat membantu mengidentifikasi dan menunjukkan tujuan pengaliran.

Nama pengaliran Anda harus unik di antara pengaliran aktif lainnya untuk buku besar yang ditentukan. Nama aliran memiliki batasan penamaan yang sama dengan nama buku besar, seperti yang didefinisikan dalam. Kuota dan batasan di Amazon QLDB

Selain nama streaming, QLDB menetapkan ID aliran ke setiap aliran QLDB yang Anda buat. ID aliran unik di antara semua aliran untuk buku besar tertentu, terlepas dari statusnya.

Tanggal dan waktu mulai

Tanggal dan waktu untuk memulai streaming data jurnal. Nilai ini bisa berupa tanggal dan waktu di masa lalu tetapi tidak bisa di masa depan.

Tanggal dan waktu akhir

(Opsional) Tanggal dan waktu yang menentukan kapan aliran berakhir.

Jika Anda membuat aliran tidak terbatas tanpa waktu akhir, Anda harus membatalkannya secara manual untuk mengakhiri aliran. Anda juga dapat membatalkan aliran aktif dan terbatas yang belum mencapai tanggal dan waktu akhir yang ditentukan.

Aliran data Kinesis Tujuan

Kinesis Data Streams menargetkan sumber daya tempat aliran Anda menulis catatan data. Untuk mempelajari cara membuat aliran data Kinesis, lihat Membuat dan memperbarui aliran data di Panduan Pengembang Amazon Kinesis Data Streams.

> **Important**
>
> - Stream lintas wilayah dan lintas-akun tidak didukung. Aliran data Kinesis yang ditentukan harus sama Wilayah AWS dan akun sebagai buku besar Anda.

- Rekaman agregasi di Kinesis Data Streams diaktifkan secara default. Opsi ini memungkinkan QLDB mempublikasikan beberapa catatan data dalam satu catatan Kinesis Data Streams, meningkatkan jumlah catatan yang dikirim per panggilan API.

  Agregasi rekaman memiliki implikasi penting untuk memproses catatan dan memerlukan de-agregasi di konsumen aliran Anda. Untuk mempelajari lebih lanjut, lihat konsep kunci KPL dan de-agregasi Konsumen di Panduan Pengembang Amazon Kinesis Data Streams.

Peran IAM

Peran IAM yang memungkinkan QLDB untuk mengasumsikan izin menulis ke aliran data Kinesis Anda. Anda dapat menggunakan konsol QLDB untuk membuat peran ini secara otomatis, atau Anda dapat membuatnya secara manual di IAM. Untuk mempelajari cara membuatnya secara manual, lihatIzin streaming.

Untuk meneruskan peran ke QLDB saat meminta aliran jurnal, Anda harus memiliki izin untuk melakukan `iam:PassRole` tindakan pada sumber daya peran IAM.

## ARN Streaming

Setiap aliran jurnal QLDB adalah subsumber daya dari buku besar dan diidentifikasi secara unik oleh Amazon Resource Name (ARN). Berikut ini adalah contoh ARN dari aliran QLDB dengan ID aliran untuk buku besar bernama. `IiPT4brpZCqCq3f4MTHbYy` `exampleLedger`

```
arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/IiPT4brpZCqCq3f4MTHbYy
```

Bagian berikut menjelaskan cara membuat dan membatalkan aliran QLDB menggunakan. AWS Management Console

## AWS Management Console

Ikuti langkah-langkah ini untuk membuat atau membatalkan aliran QLDB menggunakan konsol QLDB.

Untuk membuat aliran (konsol)

1. Masuk ke AWS Management Console, dan buka konsol QLDB Amazon di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih Pengaliran.
3. Pilih Buat aliran QLDB.
4. Pada halaman Create QLDB stream, masukkan pengaturan berikut:

   - Nama aliran — Nama yang ingin Anda tetapkan ke aliran QLDB.
   - Ledger — Buku besar yang data jurnalnya ingin Anda streaming.
   - Tanggal dan waktu mulai - Stempel waktu inklusif dalam Coordinated Universal Time (UTC) untuk memulai streaming data jurnal. Stempel waktu ini default ke tanggal dan waktu saat ini. Itu tidak bisa di masa depan dan harus lebih awal dari tanggal dan waktu Akhir.
   - Tanggal dan waktu berakhir — (Opsional) Stempel waktu eksklusif (UTC) yang menentukan kapan aliran berakhir. Jika Anda membiarkan parameter ini kosong, aliran berjalan tanpa batas hingga Anda membatalkannya.
   - Aliran tujuan - Kinesis Data Streams menargetkan sumber daya tempat aliran Anda menulis catatan data. Gunakan format ARN berikut.

     ```
     arn:aws:kinesis:aws-region:account-id:stream/kinesis-stream-name
     ```

     Berikut adalah contohnya.

     ```
     arn:aws:kinesis:us-east-1:123456789012:stream/stream-for-qldb
     ```

     Stream lintas wilayah dan lintas-akun tidak didukung. Aliran data Kinesis yang ditentukan harus sama Wilayah AWS dan akun sebagai buku besar Anda.
   - Aktifkan agregasi rekaman di Kinesis Data Streams — (Diaktifkan secara default) Memungkinkan QLDB mempublikasikan beberapa catatan data dalam satu catatan Kinesis Data Streams, meningkatkan jumlah rekaman yang dikirim per panggilan API.
   - Akses layanan — Peran IAM yang memberikan izin menulis QLDB ke aliran data Kinesis Anda.

     Untuk meneruskan peran ke QLDB saat meminta aliran jurnal, Anda harus memiliki izin untuk melakukan *iam:PassRole* tindakan pada sumber daya peran IAM.

- Buat dan gunakan peran layanan baru — Biarkan konsol membuat peran baru untuk Anda dengan izin yang diperlukan untuk aliran data Kinesis yang ditentukan.
- Gunakan peran layanan yang ada — Untuk mempelajari cara membuat peran ini secara manual di IAM, lihatIzin streaming.

- Tag - (Opsional) Tambahkan metadata ke aliran dengan melampirkan tag sebagai pasangan nilai kunci. Anda dapat menambahkan tag ke aliran Anda untuk membantu mengatur dan mengidentifikasi mereka. Untuk informasi selengkapnya, lihat Pemberian tag pada sumber daya Amazon QLDB.

  Pilih Tambahkan tag, lalu masukkan pasangan nilai kunci apa pun yang sesuai.

5. Ketika pengaturan seperti yang Anda inginkan, pilih Buat aliran QLDB.

   Jika pengiriman permintaan Anda berhasil, konsol akan kembali ke halaman Streams utama dan mencantumkan aliran QLDB Anda dengan statusnya saat ini.

6. Setelah streaming Anda aktif, gunakan Kinesis untuk memproses data streaming Anda dengan aplikasi konsumen.

   Buka konsol Kinesis Data Streams di https://console.aws.amazon.com/kinesis/.

Untuk informasi tentang format rekaman data aliran, lihatCatatan aliran QLDB dalam Kinesis.

Untuk mempelajari cara menangani aliran yang menghasilkan kesalahan, lihatMenangani aliran yang terganggu.

Untuk membatalkan streaming (konsol)

Anda tidak dapat memulai ulang aliran QLDB setelah Anda membatalkannya. Untuk melanjutkan pengiriman data ke Kinesis Data Streams, Anda dapat membuat aliran QLDB baru.

1. Buka konsol QLDB Amazon di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih Pengaliran.
3. Dalam daftar aliran QLDB, pilih aliran aktif yang ingin Anda batalkan.
4. Pilih Batalkan aliran. Konfirmasikan ini **`cancel stream`** dengan memasukkan kotak yang disediakan.

Untuk informasi tentang penggunaan QLDB API dengan AWS SDK atau untuk membuat dan mengelola AWS CLI aliran jurnal, lihat. Berkembang dengan aliran di QLDB

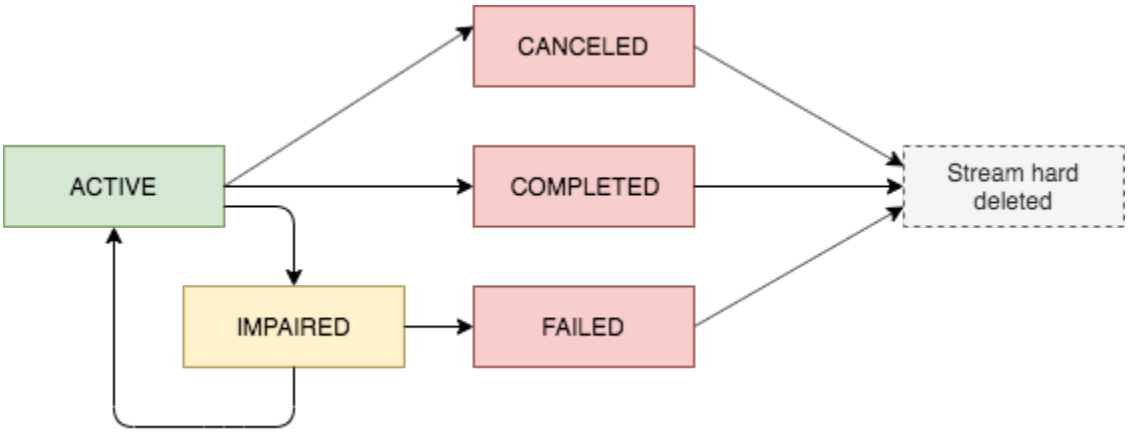
# Negara aliran

Status aliran QLDB dapat menjadi salah satu dari berikut ini:

- `ACTIVE`— Saat ini streaming atau menunggu untuk melakukan streaming data (untuk aliran tidak terbatas tanpa waktu akhir).
- `COMPLETED`— Telah berhasil menyelesaikan streaming semua blok jurnal dalam rentang waktu yang ditentukan. Ini adalah status terakhir.
- `CANCELED`— Diakhiri oleh permintaan pengguna sebelum waktu akhir yang ditentukan dan tidak lagi aktif streaming data. Ini adalah status terakhir.
- `IMPAIRED`— Tidak dapat menulis catatan ke Kinesis karena kesalahan yang memerlukan tindakan Anda. Ini adalah keadaan non-terminal yang dapat dipulihkan.

  Jika Anda menyelesaikan kesalahan dalam satu jam, aliran secara otomatis bergerak ke `ACTIVE` status. Jika kesalahan tetap tidak terselesaikan setelah satu jam, aliran secara otomatis bergerak ke `FAILED` status.
- `FAILED`— Tidak dapat menulis catatan ke Kinesis karena kesalahan dan dalam keadaan terminal yang tidak dapat dipulihkan.

Diagram berikut menggambarkan bagaimana sumber daya aliran QLDB dapat bertransisi antar negara.




## Kedaluwarsa untuk aliran terminal

Sumber daya aliran yang berada dalam status terminal (`CANCELED`,`COMPLETED`, dan`FAILED`) tunduk pada periode retensi 7 hari. Mereka secara otomatis dihapus setelah batas ini kedaluwarsa.

Setelah aliran terminal dihapus, Anda tidak dapat lagi menggunakan konsol QLDB atau QLDB API untuk mendeskripsikan atau mencantumkan sumber daya aliran.

## Menangani aliran yang terganggu

Jika streaming Anda mengalami kesalahan, streaming akan beralih ke `IMPAIRED` status terlebih dahulu. QLDB terus `IMPAIRED` mencoba lagi streaming hingga satu jam.

Jika Anda menyelesaikan kesalahan dalam satu jam, aliran secara otomatis bergerak ke `ACTIVE` status. Jika kesalahan tetap tidak terselesaikan setelah satu jam, aliran secara otomatis bergerak ke `FAILED` status.

Aliran yang terganggu atau gagal dapat memiliki salah satu penyebab kesalahan berikut:

- `KINESIS_STREAM_NOT_FOUND`— Sumber daya Kinesis Data Streams tujuan tidak ada. Verifikasi bahwa aliran data Kinesis yang Anda berikan dalam permintaan aliran QLDB Anda sudah benar. Kemudian, pergi ke Kinesis dan membuat aliran data yang Anda tentukan.
- `IAM_PERMISSION_REVOKED`— QLDB tidak memiliki izin yang cukup untuk menulis catatan data ke aliran data Kinesis yang Anda tentukan. Verifikasi bahwa Anda menetapkan kebijakan untuk aliran data Kinesis tertentu yang memberikan izin layanan QLDB () `qldb.amazonaws.com` untuk tindakan berikut:
  - `kinesis:PutRecord`
  - `kinesis:PutRecords`
  - `kinesis:DescribeStream`
  - `kinesis:ListShards`

### Memantau aliran yang terganggu

Jika aliran menjadi terganggu, konsol QLDB menampilkan spanduk yang menampilkan detail tentang aliran dan kesalahan yang ditemuinya. Anda juga dapat menggunakan operasi `DescribeJournalKinesisStream` API untuk mendapatkan status aliran dan penyebab kesalahan yang mendasarinya.

Selain itu, Anda dapat menggunakan Amazon CloudWatch untuk membuat alarm yang memantau `IsImpaired` metrik aliran. Untuk informasi tentang memantau metrik CloudWatch QLDB dengan, lihat. Dimensi dan metrik Amazon QLDB

# Berkembang dengan aliran di QLDB

Bagian ini merangkum operasi API yang dapat Anda gunakan dengan AWS SDK atau AWS CLI untuk membuat dan mengelola aliran jurnal di Amazon QLDB. Ini juga menjelaskan contoh aplikasi yang mendemonstrasikan operasi ini dan menggunakan Kinesis Client Library (KCL) atau AWS Lambda untuk mengimplementasikan konsumen aliran.

Anda dapat menggunakan KCL untuk membangun aplikasi konsumen untuk Amazon Kinesis Data Streams. KCL menyederhanakan pengodean dengan menyediakan abstraksi yang berguna di atas Kinesis Data Streams API tingkat rendah. Untuk mempelajari lebih lanjut tentang KCL, lihat Menggunakan Perpustakaan Klien Kinesis di Panduan Pengembang Amazon Kinesis Data Streams.

Daftar Isi



## API aliran jurnal QLDB

QLDB API menyediakan operasi aliran jurnal berikut untuk digunakan oleh program aplikasi:

- `StreamJournalToKinesis`— Membuat aliran jurnal untuk buku besar QLDB tertentu. Aliran menangkap setiap revisi dokumen yang berkomitmen pada jurnal buku besar dan mengirimkan data ke sumber daya Kinesis Data Streams tertentu.
  - Rekaman agregasi di Kinesis Data Streams diaktifkan secara default. Opsi ini memungkinkan QLDB mempublikasikan beberapa catatan data dalam satu catatan Kinesis Data Streams, meningkatkan jumlah catatan yang dikirim per panggilan API.

    Agregasi rekaman memiliki implikasi penting untuk memproses catatan dan memerlukan de-agregasi di konsumen aliran Anda. Untuk mempelajari lebih lanjut, lihat konsep kunci KPL dan de-agregasi Konsumen di Panduan Pengembang Amazon Kinesis Data Streams.
- `DescribeJournalKinesisStream`— Mengembalikan informasi rinci tentang aliran jurnal QLDB yang diberikan. Outputnya mencakup ARN, nama aliran, status saat ini, waktu pembuatan, dan parameter permintaan pembuatan aliran asli Anda.

- `ListJournalKinesisStreamsForLedger`— Mengembalikan daftar semua deskriptor aliran jurnal QLDB untuk buku besar yang diberikan. Output dari setiap deskriptor aliran mencakup detail yang sama yang dikembalikan oleh`DescribeJournalKinesisStream`.
- `CancelJournalKinesisStream`— Mengakhiri aliran jurnal QLDB yang diberikan. Sebelum streaming dapat dibatalkan, statusnya saat ini harus`ACTIVE`.

  Anda tidak dapat memulai ulang aliran setelah Anda membatalkannya. Untuk melanjutkan pengiriman data ke Kinesis Data Streams, Anda dapat membuat aliran QLDB baru.

Untuk deskripsi lengkap tentang operasi API ini, lihat. Referensi API Amazon QLDB

Untuk informasi tentang membuat dan mengelola aliran jurnal menggunakan AWS CLI, lihat Referensi AWS CLI Perintah.

## Aplikasi sampel

QLDB menyediakan contoh aplikasi yang menunjukkan berbagai operasi menggunakan aliran jurnal. Aplikasi ini bersifat open source di GitHub situs AWS Sampel.

Topik

- Operasi dasar (Java)
- Integrasi dengan OpenSearch Layanan (Python)
- Integrasi dengan Amazon SNS dan Amazon SQS (Python)

### Operasi dasar (Java)

Untuk contoh kode Java yang menunjukkan operasi dasar untuk aliran jurnal QLDB, lihat repositori aws-samples/ -java. GitHub amazon-qldb-dmv-sample Untuk petunjuk tentang cara mengunduh dan menginstal aplikasi sampel ini, lihatMenginstal aplikasi sampel Amazon QLDB Java.

> **Note**
>
> Setelah Anda menginstal aplikasi, jangan lanjutkan ke Langkah 1 dari tutorial Java untuk membuat buku besar. Contoh aplikasi untuk streaming ini membuat `vehicle-registration` buku besar untuk Anda.

Contoh aplikasi ini mengemas kode sumber lengkap dari tutorial java dan dependensinya, termasuk modul berikut:

- AWS SDK for Java— Untuk membuat dan menghapus sumber daya QLDB dan Kinesis Data Streams, termasuk buku besar, aliran jurnal QLDB, dan aliran data Kinesis.
- Driver QLDB Amazon untuk JavaUntuk menjalankan transaksi data pada buku besar menggunakan pernyataan PartiQL, termasuk membuat tabel dan memasukkan dokumen.
- Perpustakaan Klien Kinesis — Untuk mengkonsumsi dan memproses data dari aliran data Kinesis.

Menjalankan kode

StreamJournalKelas berisi kode tutorial yang menunjukkan operasi berikut:

1. Buat buku besar bernama`vehicle-registration`, buat tabel, dan muat dengan data sampel.

   > **Note**
   >
   > Sebelum menjalankan kode ini, pastikan Anda belum memiliki buku besar aktif bernama`vehicle-registration`.

2. Buat aliran data Kinesis, peran IAM yang memungkinkan QLDB mengasumsikan izin menulis untuk aliran data Kinesis, dan aliran jurnal QLDB.
3. Gunakan KCL untuk memulai pembaca aliran yang memproses aliran data Kinesis dan mencatat setiap catatan data QLDB.
4. Gunakan data aliran untuk memvalidasi rantai hash dari buku besar `vehicle-registration` sampel.
5. Bersihkan semua sumber daya dengan menghentikan pembaca aliran, membatalkan aliran jurnal QLDB, menghapus buku besar, dan menghapus aliran data Kinesis.

Untuk menjalankan kode `StreamJournal` tutorial, masukkan perintah Gradle berikut dari direktori root proyek Anda.

```
./gradlew run -Dtutorial=streams.StreamJournal
```

### Integrasi dengan OpenSearch Layanan (Python)

Untuk contoh aplikasi Python yang menunjukkan cara mengintegrasikan aliran QLDB dengan Amazon OpenSearch Service, lihat repositori aws-samples/ -. GitHub amazon-qldb-streaming-amazon opensearch-service-sample-python Aplikasi ini menggunakan AWS Lambda fungsi untuk mengimplementasikan konsumen Kinesis Data Streams.

Untuk mengkloning repositori, masukkan perintah berikut. `git`

```
git clone https://github.com/aws-samples/amazon-qldb-streaming-amazon-opensearch-service-sample-python.git
```

Untuk menjalankan aplikasi sampel, lihat README aktif GitHub untuk instruksi.

### Integrasi dengan Amazon SNS dan Amazon SQS (Python)

Untuk contoh aplikasi Python yang menunjukkan cara mengintegrasikan aliran QLDB dengan Amazon Simple Notification Service (Amazon SNS), lihat repositori aws-samples/ -. GitHub amazon-qldb-streams-dmv sample-lambda-python

Aplikasi ini menggunakan AWS Lambda fungsi untuk mengimplementasikan konsumen Kinesis Data Streams. Ini mengirim pesan ke topik Amazon SNS, yang memiliki antrian Amazon Simple Queue Service (Amazon SQS) berlangganan.

Untuk mengkloning repositori, masukkan perintah berikut. `git`

```
git clone https://github.com/aws-samples/amazon-qldb-streams-dmv-sample-lambda-python.git
```

Untuk menjalankan aplikasi sampel, lihat README aktif GitHub untuk instruksi.

## Catatan aliran QLDB dalam Kinesis

Aliran QLDB Amazon menulis tiga jenis catatan data ke sumber daya Amazon Kinesis Data Streams yang diberikan: kontrol, ringkasan blokir, dan detail revisi. Ketiga jenis rekaman ditulis dalam representasi biner dari format Amazon Ion.

Catatan kontrol menunjukkan awal dan penyelesaian aliran QLDB Anda. Setiap kali revisi dilakukan pada jurnal Anda, aliran QLDB menulis semua data blok jurnal terkait dalam ringkasan blok dan catatan detail revisi.

Tiga jenis rekaman adalah polimorfik. Semuanya terdiri dari catatan tingkat atas umum yang berisi ARN aliran QLDB, jenis catatan, dan muatan catatan. Catatan tingkat atas ini memiliki format berikut.

```
{
  qldbStreamArn: string,
  recordType: string,
  payload: {
    //control | block summary | revision details record
  }
}
```

`recordType`Bidang dapat memiliki salah satu dari tiga nilai:

- `CONTROL`
- `BLOCK_SUMMARY`
- `REVISION_DETAILS`

Bagian berikut menjelaskan format dan isi setiap catatan muatan individu.

> **Note**
>
> QLDB menulis semua catatan aliran ke Kinesis Data Streams dalam representasi biner Amazon Ion. Contoh berikut disediakan dalam representasi teks Ion untuk mengilustrasikan isi rekaman dalam format yang dapat dibaca.

Topik



## Catatan kontrol

Aliran QLDB menulis catatan kontrol untuk menunjukkan peristiwa awal dan penyelesaiannya. Berikut ini adalah contoh catatan kontrol dengan data sampel untuk masing-masing`controlRecordType`:

- `CREATED`— Catatan pertama yang ditulis oleh aliran QLDB ke Kinesis untuk menunjukkan bahwa aliran yang baru Anda buat aktif.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
    controlRecordType:"CREATED"
  }
}
```

- `COMPLETED`— Catatan terakhir yang ditulis oleh aliran QLDB ke Kinesis untuk menunjukkan bahwa aliran Anda telah mencapai tanggal dan waktu akhir yang ditentukan. Catatan ini tidak ditulis jika Anda membatalkan streaming.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
    controlRecordType:"COMPLETED"
  }
}
```

## Blokir catatan ringkasan

Catatan ringkasan blok mewakili blok jurnal tempat revisi dokumen Anda dilakukan. Blok adalah objek yang berkomitmen pada jurnal QLDB Anda selama transaksi.

Muatan catatan ringkasan blok berisi alamat blok, stempel waktu, dan metadata lain dari transaksi yang melakukan blok tersebut. Ini juga mencakup atribut ringkasan dari revisi di blok dan pernyataan PartiQL yang melakukannya. Berikut ini adalah contoh catatan ringkasan blok dengan data sampel.

> **Note**
>
> Contoh ringkasan blok ini disediakan hanya untuk tujuan informasi. Hash yang ditampilkan bukanlah nilai hash yang dihitung secara nyata.

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

Di `revisionSummaries` lapangan, beberapa revisi mungkin tidak memiliki a`documentId`. Ini adalah revisi sistem internal saja yang tidak berisi data pengguna. Aliran QLDB menyertakan revisi ini dalam catatan ringkasan blok masing-masing karena hash dari revisi ini adalah bagian dari rantai hash lengkap jurnal. Rantai hash penuh diperlukan untuk verifikasi kriptografi.

Hanya revisi yang memiliki ID dokumen yang diterbitkan dalam catatan rincian revisi terpisah, seperti yang dijelaskan di bagian berikut.

## Revisi rincian catatan

Catatan rincian revisi mewakili revisi dokumen yang berkomitmen untuk jurnal Anda. Payload berisi semua atribut dari tampilan komited revisi, bersama dengan nama tabel terkait dan ID tabel. Berikut ini adalah contoh catatan revisi dengan data sampel.

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

## Menangani duplikat dan catatan out-of-order

Aliran QLDB dapat mempublikasikan duplikat dan catatan ke out-of-order Kinesis Data Streams. Jadi, aplikasi konsumen mungkin perlu menerapkan logikanya sendiri untuk mengidentifikasi dan menangani skenario tersebut. Ringkasan blok dan catatan detail revisi mencakup bidang yang dapat Anda gunakan untuk tujuan ini. Dikombinasikan dengan fitur layanan hilir, bidang ini dapat menunjukkan identitas unik dan urutan ketat untuk catatan.

Misalnya, pertimbangkan aliran yang mengintegrasikan QLDB dengan indeks untuk memberikan kemampuan OpenSearch pencarian teks lengkap melalui dokumen. Dalam kasus penggunaan ini, Anda harus menghindari pengindeksan revisi basi (out-of-order) dokumen. Untuk menerapkan

pengurutan dan deduplikasi, Anda dapat menggunakan ID dokumen dan bidang versi eksternal di OpenSearch, bersama dengan ID dokumen dan bidang versi dalam catatan detail revisi.

Untuk contoh logika deduplikasi dalam contoh aplikasi yang mengintegrasikan QLDB dengan OpenSearch Amazon Service, lihat repositori aws-samples/ -. GitHub amazon-qldb-streaming-amazon opensearch-service-sample-python

## Izin streaming di QLDB

Sebelum membuat aliran QLDB Amazon, Anda harus memberikan QLDB dengan izin menulis ke sumber daya Amazon Kinesis Data Streams yang Anda tentukan. Jika Anda menggunakan pelanggan yang dikelola AWS KMS key untuk enkripsi sisi server dari aliran Kinesis Anda, Anda juga harus memberikan izin kepada QLDB untuk menggunakan kunci enkripsi simetris yang Anda tentukan. Kinesis Data Streams tidak mendukung kunci KMS asimetris.

Untuk menyediakan aliran QLDB Anda dengan izin yang diperlukan, Anda dapat membuat QLDB mengambil peran layanan IAM dengan kebijakan izin yang sesuai. Peran layanan adalah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Note**
>
> Untuk meneruskan peran ke QLDB saat meminta aliran jurnal, Anda harus memiliki izin untuk melakukan `iam:PassRole` tindakan pada sumber daya peran IAM. Ini merupakan tambahan dari `qldb:StreamJournalToKinesis` izin pada subsumber daya aliran QLDB.
> Untuk mempelajari cara mengontrol akses ke QLDB menggunakan IAM, lihat. Bagaimana Amazon QLDB bekerja dengan IAM Untuk contoh kebijakan QLDB, lihat. Contoh kebijakan berbasis identitas untuk Amazon QLDB

Dalam contoh ini, Anda membuat peran yang memungkinkan QLDB menulis catatan data ke aliran data Kinesis atas nama Anda. Untuk informasi selengkapnya, lihat Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

Jika Anda melakukan streaming jurnal QLDB untuk pertama kalinya, Akun AWS Anda harus terlebih dahulu membuat peran IAM dengan kebijakan yang sesuai dengan melakukan hal berikut. Atau,

Anda dapat menggunakan konsol QLDB untuk secara otomatis membuat peran untuk Anda. Jika tidak, Anda dapat memilih peran yang sebelumnya Anda buat.

Topik

- Membuat kebijakan izin
- Membuat peran IAM

## Membuat kebijakan izin

Selesaikan langkah-langkah berikut untuk membuat kebijakan izin untuk aliran QLDB. Contoh ini menunjukkan kebijakan Kinesis Data Streams yang memberikan izin QLDB untuk menulis catatan data ke aliran data Kinesis yang Anda tentukan. Jika berlaku, contoh ini juga menunjukkan kebijakan kunci yang memungkinkan QLDB menggunakan kunci KMS enkripsi simetris Anda.

Untuk informasi selengkapnya tentang kebijakan Kinesis Data Streams, lihat Mengontrol akses ke resource Amazon Kinesis Data Streams menggunakan IAM dan Izin untuk menggunakan kunci KMS buatan pengguna di Panduan Pengembang Amazon Kinesis Data Streams. Untuk mempelajari lebih lanjut tentang kebijakan AWS KMS utama, lihat Menggunakan kebijakan utama AWS KMS di Panduan AWS Key Management Service Pengembang.

> **Note**
>
> Aliran data Kinesis dan kunci KMS Anda harus sama Wilayah AWS dan akun dengan buku besar QLDB Anda.

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di kolom navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul laman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.
4. Pilih tab JSON.
5. Masukkan dokumen kebijakan JSON.

- Jika Anda menggunakan kunci KMS yang dikelola pelanggan untuk enkripsi sisi server aliran Kinesis Anda, gunakan contoh dokumen kebijakan berikut. *Untuk menggunakan kebijakan ini, ganti us-east-1, 123456789012,, dan 1234abcd-12ab-34cd-56ef-1234567890ab dalam contoh dengan informasi Anda kinesis-stream-namesendiri.*

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

- Jika tidak, gunakan contoh dokumen kebijakan berikut. Untuk menggunakan kebijakan ini, ganti *us-east-1*, 123456789012*,* dan dalam contoh dengan informasi Anda sendiri. *kinesis-stream-name*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBStreamKinesisPermissions",
            "Action": [ "kinesis:PutRecord*", "kinesis:DescribeStream",
 "kinesis:ListShards" ],
            "Effect": "Allow",
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/kinesis-
stream-name"
        }
```

```
    ]
}
```

6. Pilih Tinjau kebijakan.

> **Note**
>
> Anda dapat berpindah antara tab Editor visual dan JSON kapan pun. Namun, apabila Anda melakukan perubahan atau memilih Tinjau kebijakan pada tab Editor visual, IAM dapat merestrukturisasi kebijakan Anda untuk menjadikannya optimal bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. Pada halaman Peninjauan Kebijakan, ketikkan Nama dan Deskripsi opsional untuk kebijakan yang sedang Anda buat. Tinjau Summary (Ringkasan) kebijakan untuk melihat izin yang diberikan oleh kebijakan Anda. Kemudian pilih Buat kebijakan untuk menyimpan pekerjaan Anda.

## Membuat peran IAM

Setelah membuat kebijakan izin untuk aliran QLDB, Anda kemudian dapat membuat peran IAM dan melampirkan kebijakan Anda padanya.

Untuk membuat peran layanan untuk QLDB (konsol IAM)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran, dan lalu pilih Buat peran.
3. Untuk jenis entitas Tepercaya, pilih Layanan AWS.
4. Untuk kasus Layanan atau penggunaan, pilih QLDB, lalu pilih kasus penggunaan QLDB.
5. Pilih Selanjutnya.
6. Pilih kotak di samping kebijakan yang Anda buat di langkah sebelumnya.
7. (Opsional) Tetapkan batas izin. Ini adalah fitur lanjutan yang tersedia untuk peran layanan, tetapi bukan peran tertaut layanan.

    a. Buka bagian Setel batas izin, lalu pilih Gunakan batas izin untuk mengontrol izin peran maksimum.

IAM menyertakan daftar kebijakan yang AWS dikelola dan dikelola pelanggan di akun Anda.

b. Pilih kebijakan yang akan digunakan untuk batas izin.

8. Pilih Selanjutnya.
9. Masukkan nama peran atau akhiran nama peran untuk membantu Anda mengidentifikasi tujuan peran.

> **Important**
>
> Saat Anda memberi nama peran, perhatikan hal berikut:
>
> - Nama peran harus unik di dalam diri Anda Akun AWS, dan tidak dapat dibuat unik berdasarkan kasus.
>
>  Misalnya, jangan membuat peran bernama keduanya **`PRODROLE`** dan**`prodrole`**. Ketika nama peran digunakan dalam kebijakan atau sebagai bagian dari ARN, nama peran tersebut peka huruf besar/kecil, namun ketika nama peran muncul kepada pelanggan di konsol, seperti selama proses masuk, nama peran tersebut tidak peka huruf besar/kecil.
> - Anda tidak dapat mengedit nama peran setelah dibuat karena entitas lain mungkin mereferensikan peran tersebut.

10. (Opsional) Untuk Deskripsi, masukkan deskripsi untuk peran tersebut.
11. (Opsional) Untuk mengedit kasus penggunaan dan izin untuk peran, di Langkah 1: Pilih entitas tepercaya atau Langkah 2: Tambahkan izin, pilih Edit.
12. (Opsional) Untuk membantu mengidentifikasi, mengatur, atau mencari peran, tambahkan tag sebagai pasangan nilai kunci. Untuk informasi selengkapnya tentang penggunaan tanda di IAM, lihat Menandai sumber daya IAM di Panduan Pengguna IAM.
13. Tinjau peran lalu pilih Buat peran.

Dokumen JSON berikut adalah contoh kebijakan kepercayaan yang memungkinkan QLDB untuk mengambil peran IAM dengan izin khusus yang melekat padanya.

```
{
    "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**
>
> Contoh kebijakan kepercayaan ini menunjukkan bagaimana Anda dapat menggunakan kunci konteks kondisi `aws:SourceAccount` global `aws:SourceArn` dan global untuk mencegah masalah wakil yang membingungkan. Dengan kebijakan kepercayaan ini, QLDB dapat mengambil peran untuk aliran QLDB apa pun di akun untuk buku besar saja. `123456789012` `myExampleLedger`
>
> Untuk informasi selengkapnya, lihat Pencegahan confused deputy lintas layanan.

Setelah membuat peran IAM Anda, kembali ke konsol QLDB dan segarkan halaman aliran Buat QLDB sehingga dapat menemukan peran baru Anda.

## Kesalahan umum untuk aliran jurnal di QLDB

Bagian ini menjelaskan kesalahan runtime yang dilemparkan oleh Amazon QLDB untuk permintaan aliran jurnal.

Berikut ini adalah daftar pengecualian umum yang dikembalikan oleh layanan. Setiap pengecualian mencakup pesan kesalahan tertentu, diikuti dengan deskripsi singkat dan saran untuk solusi yang mungkin.

AccessDeniedException

*Pesan: Pengguna: UserARN tidak berwenang untuk melakukan: iam: PassRole on resource: roLearn*

Anda tidak memiliki izin untuk meneruskan peran IAM ke layanan QLDB. QLDB memerlukan peran untuk semua permintaan aliran jurnal, dan Anda harus memiliki izin untuk meneruskan peran ini ke QLDB. Peran ini menyediakan QLDB dengan izin menulis di sumber daya Amazon Kinesis Data Streams yang Anda tentukan.

Verifikasi bahwa Anda menentukan kebijakan IAM yang memberikan izin untuk menjalankan operasi `PassRole` API pada sumber daya peran IAM yang ditentukan untuk layanan QLDB (). `qldb.amazonaws.com` Untuk contoh kebijakan, lihatContoh kebijakan berbasis identitas untuk Amazon QLDB.

IllegalArgumentException

*Pesan: QLDB mengalami kesalahan saat memvalidasi Kinesis Data Streams: Respons dari Kinesis: ErrorCode ErrorMessage*

Kemungkinan penyebab kesalahan ini adalah sumber daya Kinesis Data Streams yang disediakan tidak ada. Atau, QLDB tidak memiliki izin yang cukup untuk menulis catatan data ke aliran data Kinesis yang Anda tentukan.

Verifikasi bahwa aliran data Kinesis yang Anda berikan dalam permintaan aliran Anda sudah benar. Untuk informasi selengkapnya, lihat Membuat dan memperbarui aliran data di Panduan Pengembang Amazon Kinesis Data Streams.

Selain itu, verifikasi bahwa Anda menentukan kebijakan untuk aliran data Kinesis yang ditentukan yang memberikan izin layanan QLDB () `qldb.amazonaws.com` untuk tindakan berikut. Untuk informasi selengkapnya, lihat Izin streaming.

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

Pesan: Respon tak terduga dari Kinesis Data Streams saat memvalidasi konfigurasi Kinesis. *Tanggapan dari Kinesis: ErrorCode ErrorMessage*

Upaya untuk menulis catatan data ke aliran data Kinesis yang disediakan gagal dengan respons kesalahan Kinesis yang disediakan. Untuk informasi selengkapnya tentang kemungkinan penyebab, lihat Memecahkan Masalah produsen Amazon Kinesis Data Streams di Panduan Pengembang Amazon Kinesis Data Streams.

IllegalArgumentException

Pesan: Tanggal mulai tidak boleh lebih besar dari tanggal akhir.

Keduanya `InclusiveStartTime` dan `ExclusiveEndTime` harus dalam format tanggal dan waktu ISO 8601 dan dalam Coordinated Universal Time (UTC).

IllegalArgumentException

Pesan: Tanggal mulai tidak bisa di masa depan.

Keduanya `InclusiveStartTime` dan `ExclusiveEndTime` harus dalam format `ISO 8601` tanggal dan waktu dan dalam UTC.

LimitExceededException

Pesan: Melebihi batas 5 aliran Jurnal yang berjalan secara bersamaan ke Kinesis Data Streams

QLDB memberlakukan batas default lima aliran jurnal bersamaan.

# Manajemen buku besar di Amazon QLDB

Bab ini menjelaskan cara menggunakan QLDB API,AWS Command Line Interface (AWS CLI), danAWS CloudFormation untuk melakukan operasi manajemen buku besar di Amazon QLDB.

Anda juga dapat melakukan tugas-tugas yang sama ini menggunakan AWS Management Console. Untuk informasi selengkapnya, lihat Mengakses Amazon QLDB menggunakan konsol.

Topik



## Operasi dasar untuk buku besar Amazon QLDB

Anda dapat menggunakan QLDB API atauAWS Command Line Interface (AWS CLI) untuk membuat, memperbarui, dan menghapus buku besar di Amazon QLDB. Anda juga dapat mendaftar semua buku besar di akun Anda, atau mendapatkan informasi tentang buku besar tertentu.

Topik berikut memberikan contoh kode singkat yang menunjukkan langkah-langkah umum untuk operasi buku besar menggunakanAWS SDK for Java danAWS CLI.

Topik



Untuk contoh kode yang menunjukkan operasi ini dalam aplikasi sampel lengkap, lihatMemulai dengan driver tutorial dan GitHub repositori berikut:

- Java: Tutorial | GitHub repositori
- Node.js: Tutorial | GitHub repositori

- Python: Tutorial | GitHub repositori

## Membuat buku besar

GunakanCreateLedger operasi untuk membuat buku besar di AndaAkun AWS. Anda harus memberikan informasi berikut ini:

- Nama buku besar — Nama buku besar yang ingin Anda buat di akun Anda. Nama harus unik di antara semua buku besar Anda di saat iniWilayah AWS.

  Penamaan kendala untuk nama buku besar didefinisikan dalamKuota dan batasan di Amazon QLDB.
- Mode izin — Mode perizinan untuk ditetapkan ke buku besar. Pilih salah satu opsi berikut:
  - Izinkan semua — Mode izin warisan yang memungkinkan kontrol akses dengan rincian tingkat API untuk buku besar.

    Mode ini memungkinkan pengguna yang memiliki izin API SendCommand untuk buku besar ini untuk menjalankan semua perintah PartiQL (maka, ALLOW_ALL) pada setiap tabel dalam buku besar yang ditentukan. Mode ini mengabaikan setiap kebijakan izin IAM tingkat tabel atau tingkat perintah yang Anda buat untuk buku besar.
  - Standard - (Direkomendasikan) Mode perizinan yang memungkinkan kontrol akses dengan rincian yang lebih halus untuk buku besar, tabel, dan perintah PartiQL. Kami sangat merekomendasikan untuk menggunakan mode izin ini untuk memaksimalkan keamanan data buku besar Anda.

    Secara default, mode ini menyangkal semua permintaan untuk menjalankan perintah PartiQL pada setiap tabel dalam buku besar ini. Untuk mengizinkan perintah PartiQL, Anda harus membuat kebijakan izin IAM untuk sumber daya tabel tertentu dan tindakan PartiQL, selain izinSendCommand API untuk buku besar. Untuk informasi, lihat Memulai dengan mode izin standar di Amazon QLDB.
- Perlindungan penghapusan — (Opsional) Bendera yang mencegah buku besar dihapus oleh pengguna manapun. Jika Anda tidak menentukan selama pembuatan buku besar, fitur ini diaktifkan (true) secara default.

  Jika perlindungan penghapusan diaktifkan, Anda harus menonaktifkannya terlebih dahulu sebelum dapat menghapus buku besar. Anda dapat menonaktifkannya dengan menggunakanUpdateLedger operasi untuk mengatur bendera kefalse.

- AWS KMS key- (Opsional) Kunci diAWS Key Management Service (AWS KMS) digunakan untuk enkripsi data saat istirahat. Pilih salah satu jenis berikutAWS KMS keys:
  - AWSdimiliki kunci KMS - Gunakan kunci KMS yang dimiliki dan dikelola olehAWS atas nama Anda.

    Jika Anda tidak menentukan parameter ini selama pembuatan buku besar, buku besar menggunakan jenis kunci ini secara default. Anda juga dapat menggunakan string`AWS_OWNED_KMS_KEY` untuk menentukan jenis kunci ini. Opsi ini tidak memerlukan pengaturan tambahan.
  - Kunci KMS yang dikelola pelanggan — Gunakan kunci KMS enkripsi simetris di akun Anda yang Anda buat, miliki, dan kelola. QLDB tidak mendukung kunci asimetris.

    Opsi ini mengharuskan Anda untuk membuat kunci KMS atau menggunakan kunci yang ada di akun Anda. Untuk petunjuk tentang cara membuat kunci yang dikelola pelanggan, lihat Membuat kunci KMS enkripsi simetris di PanduanAWS Key Management Service Pengembang.

    Anda dapat menentukan kunci KMS yang dikelola pelanggan dengan menggunakan ID, alias, atau Amazon Resource Name (ARN). Untuk mempelajari lebih lanjut, lihat Pengidentifikasi kunci (KeyId) di PanduanAWS Key Management Service Developer.

    > **Note**
    >
    > Kunci lintas wilayah tidak didukung. Kunci KMS tertentu harusWilayah AWS sama dengan buku besar Anda.

  Untuk informasi selengkapnya, lihat Enkripsi saat istirahat di Amazon QLDB.
- Tag — (Opsional) Tambahkan metadata ke buku besar dengan melampirkan tag sebagai pasangan nilai kunci. Anda dapat menambahkan tanda ke buku besar Anda untuk membantu mengatur dan mengidentifikasi buku besar tersebut. Untuk informasi selengkapnya, lihat Pemberian tag pada sumber daya Amazon QLDB.

Buku besar tidak siap digunakan sampai QLDB membuat dan menetapkan statusnya ke`ACTIVE`.

Membuat buku besar (Java)

Untuk membuat buku besar menggunakanAWS SDK for Java

1. Buat instans dari kelas `AmazonQLDB`.

2. Buat instans dari kelas `CreateLedgerRequest` untuk memberikan informasi permintaan.

   Anda harus memberikan nama buku besar dan mode izin.

3. Jalankan metode `createLedger` dengan menyediakan objek permintaan sebagai parameter.

`createLedger`Permintaan mengembalikan`CreateLedgerResult` objek yang memiliki informasi tentang buku besar. Lihat bagian berikutnya untuk contoh penggunaan`DescribeLedger` operasi untuk memeriksa status buku besar Anda setelah Anda membuatnya.

Contoh berikut mendemonstrasikan langkah sebelumnya.

Example — Gunakan pengaturan konfigurasi default

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
CreateLedgerRequest request = new CreateLedgerRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
CreateLedgerResult result = client.createLedger(request);
```

> **Note**
>
> Buku besar menggunakan pengaturan default berikut jika Anda tidak menentukannya:
>
> - Perlindungan penghapusan - Diaktifkan (`true`).
> - Kunci KMS -AWS dimiliki kunci KMS.

Example - Nonaktifkan perlindungan penghapusan, gunakan kunci KMS yang dikelola pelanggan, dan lampirkan 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);
```

Membuat buku besar (AWS CLI)

Buat buku besar baru bernama`vehicle-registration` menggunakan pengaturan konfigurasi default.

Example

```
aws qldb create-ledger --name vehicle-registration --permissions-mode STANDARD
```

> **Note**
>
> Buku besar menggunakan pengaturan default berikut jika Anda tidak menentukannya:
>
> - Perlindungan penghapusan - Diaktifkan (`true`).
> - Kunci KMS -AWS dimiliki kunci KMS.

Atau, buat buku besar baru bernama`vehicle-registration` dengan perlindungan penghapusan dinonaktifkan, dengan kunci KMS yang dikelola pelanggan tertentu, dan dengan tag yang ditentukan.

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
```

Membuat buku besar (AWS CloudFormation)

Anda juga dapat menggunakan AWS CloudFormationtemplat untuk membuat buku besar. Untuk informasi selengkapnya, lihat AWS::QLDB::Ledgersumber daya di PanduanAWS CloudFormation Pengguna.

# Menjelaskan buku besar

GunakanDescribeLedger operasi untuk melihat detail tentang buku besar. Anda harus memberikan nama buku besar. Output dari DescribeLedger adalah dalam format yang sama seperti dari CreateLedger. Hal ini mencakup informasi berikut:

- Nama buku besar — Nama buku besar yang ingin Anda jelaskan.
- ARN — Amazon Resource Name (ARN) untuk buku besar dalam format berikut.

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

- Perlindungan penghapusan - Bendera yang menunjukkan apakah fitur perlindungan penghapusan diaktifkan.
- Tanggal dan waktu pembuatan — Tanggal dan waktu, dalam format waktu epoch, saat buku besar dibuat.
- Negara - Status buku besar saat ini. Ini dapat berupa salah satu dari nilai berikut:
  - CREATING
  - ACTIVE
  - DELETING
  - DELETED
- Modus izin - Mode izin yang ditugaskan ke buku besar. Ini dapat berupa salah satu dari nilai berikut:
  - ALLOW_ALL— Mode izin warisan yang memungkinkan kontrol akses dengan rincian tingkat API untuk buku besar.
  - STANDARD— Mode perizinan yang memungkinkan kontrol akses dengan rincian yang lebih halus untuk buku besar, tabel, dan perintah PartiQL.
- Deskripsi enkripsi - Informasi tentang enkripsi data saat istirahat di buku besar. Hal ini termasuk item berikut:
  - AWS KMS keyARN - ARN pelanggan mengelola kunci KMS yang digunakan buku besar untuk enkripsi saat istirahat. Jika ini tidak terdefinisi, buku besar menggunakan kunci KMS yangAWS dimiliki untuk enkripsi.
  - Status enkripsi - Status enkripsi saat ini saat istirahat untuk buku besar. Ini dapat berupa salah satu dari nilai berikut:
    - ENABLED— Enkripsi sepenuhnya diaktifkan menggunakan kunci tertentu.

- `UPDATING`- Perubahan kunci yang ditentukan secara aktif sedang diproses.

  Perubahan kunci dalam QLDB bersifat asinkron. Buku besar dapat diakses sepenuhnya tanpa dampak kinerja saat perubahan kunci sedang diproses. Jumlah waktu yang dibutuhkan untuk memperbarui kunci berbeda-beda tergantung pada ukuran buku besar.

- `KMS_KEY_INACCESSIBLE`- Kunci KMS yang dikelola pelanggan yang ditentukan tidak dapat diakses, dan buku besar terganggu. Entah kunci dinonaktifkan atau dihapus, atau hibah pada kunci dicabut. Ketika buku besar terganggu, buku besar tidak dapat diakses dan tidak menerima permintaan baca atau tulis apa pun.

  Buku besar yang terganggu secara otomatis kembali ke status aktif setelah Anda memulihkan hibah pada kunci, atau setelah Anda mengaktifkan kembali kunci yang dinonaktifkan. Namun, menghapus kunci KMS yang dikelola pelanggan tidak dapat diubah. Setelah kunci dihapus, Anda tidak dapat lagi mengakses buku besar yang dilindungi dengan kunci tersebut, dan data menjadi tidak dapat dipulihkan secara permanen.

- Tidak dapat diaksesAWS KMS key - Tanggal dan waktu, dalam format waktu epoch, ketika kunci KMS pertama kali menjadi tidak dapat diakses, dalam kasus kesalahan.

  Ini tidak terdefinisi jika kunci KMS dapat diakses.

Untuk informasi selengkapnya, lihat Enkripsi saat istirahat di Amazon QLDB.

> **Note**
>
> Setelah Anda membuat buku besar QLDB, itu menjadi siap untuk digunakan ketika statusnya berubah dari`CREATING` ke`ACTIVE`.

Menggambarkan buku besar (Java)

Untuk menggambarkan buku besar menggunakanAWS SDK for Java

1. Ciptakan contoh dari kelas `AmazonQLDB`. Atau, Anda dapat menggunakan contoh yang sama dari`AmazonQLDB` klien yang Anda instantiated untuk`CreateLedger` permintaan.
2. Buat instans dari`DescribeLedgerRequest` kelas dan berikan nama buku besar yang ingin Anda jelaskan.
3. Jalankan metode `describeLedger` dengan menyediakan objek permintaan sebagai parameter.

4. `describeLedger`Permintaan mengembalikan`DescribeLedgerResult` objek yang memiliki informasi terkini tentang buku besar.

Contoh kode berikut mendemonstrasikan langkah sebelumnya. Anda dapat memanggil`describeLedger` metode klien untuk mendapatkan informasi buku besar kapan saja.

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());
```

Menggambarkan buku besar (AWS CLI)

Jelaskan`vehicle-registration` buku besar yang baru Anda buat.

Example

```
aws qldb describe-ledger --name vehicle-registration
```

## Memperbarui buku besar

`UpdateLedger`Operasi saat ini memungkinkan Anda mengubah pengaturan konfigurasi berikut untuk buku besar yang ada:

- Perlindungan penghapusan — Bendera yang mencegah buku besar dihapus oleh pengguna manapun. Jika fitur ini diaktifkan, Anda harus menonaktifkannya terlebih dahulu dengan mengatur bendera ke`false` sebelum dapat menghapus buku besar.

  Jika Anda tidak menentukan parameter ini, tidak ada perubahan yang dibuat pada pengaturan perlindungan penghapusan buku besar.

- AWS KMS key- Kunci diAWS Key Management Service (AWS KMS) untuk digunakan untuk enkripsi data saat istirahat. Jika Anda tidak menentukan parameter ini, tidak ada perubahan yang dibuat ke kunci KMS buku besar.

  > **Note**
  >
  > Amazon QLDB meluncurkan dukungan untuk pelanggan yang dikelolaAWS KMS keys pada 22 Juli 2021. Buku besar apa pun yang dibuat sebelum peluncuran dilindungi secaraKunci milik AWS default, tetapi saat ini tidak memenuhi syarat untuk enkripsi saat istirahat menggunakan kunci yang dikelola pelanggan.
  > Anda dapat melihat waktu pembuatan buku besar Anda di konsol QLDB.

  Gunakan salah satu opsi berikut:

  - AWSdimiliki kunci KMS - Gunakan kunci KMS yang dimiliki dan dikelola olehAWS atas nama Anda. Untuk menggunakan jenis kunci ini, tentukan string`AWS_OWNED_KMS_KEY` untuk parameter ini. Opsi ini tidak memerlukan pengaturan tambahan.
  - Kunci KMS yang dikelola pelanggan — Gunakan kunci KMS enkripsi simetris di akun Anda yang Anda buat, miliki, dan kelola. QLDB tidak mendukung kunci asimetris.

    Opsi ini mengharuskan Anda untuk membuat kunci KMS atau menggunakan kunci yang ada di akun Anda. Untuk petunjuk tentang cara membuat kunci yang dikelola pelanggan, lihat Membuat kunci KMS enkripsi simetris di PanduanAWS Key Management Service Pengembang.

    Anda dapat menentukan kunci KMS yang dikelola pelanggan dengan menggunakan ID, alias, atau Amazon Resource Name (ARN). Untuk mempelajari lebih lanjut, lihat Pengidentifikasi kunci (KeyId) di PanduanAWS Key Management Service Developer.

    > **Note**
    >
    > Kunci lintas wilayah tidak didukung. Kunci KMS tertentu harusWilayah AWS sama dengan buku besar Anda.

  Perubahan kunci dalam QLDB bersifat asinkron. Buku besar dapat diakses sepenuhnya tanpa dampak kinerja saat perubahan kunci sedang diproses.

Anda dapat mengganti kunci sesering yang diperlukan, tetapi jumlah waktu yang diperlukan untuk memperbarui kunci bervariasi tergantung pada ukuran buku besar. Anda dapat menggunakan`DescribeLedger` operasi untuk memeriksa enkripsi saat status istirahat.

Untuk informasi selengkapnya, lihat Enkripsi saat istirahat di Amazon QLDB.

Output dari `UpdateLedger` adalah dalam format yang sama seperti dari `CreateLedger`.

Memperbarui buku besar (Java)

Untuk memperbarui buku besar menggunakanAWS SDK for Java

1. Buat instans dari kelas `AmazonQLDB`.
2. Buat instans dari kelas `UpdateLedgerRequest` untuk memberikan informasi permintaan.

   Anda harus memberikan nama ledger bersama dengan nilai Boolean baru untuk perlindungan penghapusan atau nilai string baru untuk kunci KMS.
3. Jalankan metode `updateLedger` dengan menyediakan objek permintaan sebagai parameter.

Contoh kode berikut mendemonstrasikan langkah sebelumnya. `updateLedger`Permintaan mengembalikan`UpdateLedgerResult` objek yang telah memperbarui informasi tentang buku besar.

Example — Nonaktifkan perlindungan penghapusan

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withDeletionProtection(false);
UpdateLedgerResult result = client.updateLedger(request);
```

Example - Gunakan kunci KMS yang dikelola pelanggan

```
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 - Gunakan kunci KMS yangAWS dimiliki

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withKmsKey("AWS_OWNED_KMS_KEY")
UpdateLedgerResult result = client.updateLedger(request);
```

Memperbarui buku besar (AWS CLI)

Jika`vehicle-registration` buku besar Anda mengaktifkan perlindungan penghapusan, Anda harus menonaktifkannya terlebih dahulu sebelum dapat menghapusnya.

Example

```
aws qldb update-ledger --name vehicle-registration --no-deletion-protection
```

Anda juga dapat mengubah enkripsi buku besar saat istirahat pengaturan untuk menggunakan kunci KMS dikelola pelanggan.

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Atau, Anda dapat mengubah enkripsi saat istirahat pengaturan untuk menggunakan kunci KMS yangAWS dimiliki.

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key AWS_OWNED_KMS_KEY
```

## Memperbarui mode izin buku besar

`UpdateLedgerPermissionsMode`Operasi ini memungkinkan Anda mengubah mode izin buku besar yang ada. Pilih salah satu opsi berikut:

- Izinkan semua — Mode izin warisan yang memungkinkan kontrol akses dengan rincian tingkat API untuk buku besar.

Mode ini memungkinkan pengguna yang memiliki izin API `SendCommand` untuk buku besar ini untuk menjalankan semua perintah PartiQL (maka, `ALLOW_ALL`) pada setiap tabel dalam buku besar yang ditentukan. Mode ini mengabaikan setiap kebijakan izin IAM tingkat tabel atau tingkat perintah yang Anda buat untuk buku besar.

- Standard - (Direkomendasikan) Mode perizinan yang memungkinkan kontrol akses dengan rincian yang lebih halus untuk buku besar, tabel, dan perintah PartiQL. Kami sangat merekomendasikan untuk menggunakan mode izin ini untuk memaksimalkan keamanan data buku besar Anda.

  Secara default, mode ini menyangkal semua permintaan untuk menjalankan perintah PartiQL pada setiap tabel dalam buku besar ini. Untuk mengizinkan perintah PartiQL, Anda harus membuat kebijakan izin IAM untuk sumber daya tabel tertentu dan tindakan PartiQL, selain izin`SendCommand` API untuk buku besar. Untuk informasi, lihat Memulai dengan mode izin standar di Amazon QLDB.

> **Important**
>
> Sebelum beralih ke mode`STANDARD` izin, Anda harus terlebih dahulu membuat semua kebijakan IAM dan tag tabel yang diperlukan untuk menghindari gangguan pada pengguna Anda. Untuk mempelajari lebih lanjut, lanjutkan keMigrasi ke mode izin standar.

Memperbarui mode izin buku besar (Java)

Untuk memperbarui mode izin buku besar menggunakanAWS SDK for Java

1. Buat instans dari kelas `AmazonQLDB`.
2. Buat instans dari kelas `UpdateLedgerPermissionsModeRequest` untuk memberikan informasi permintaan.

   Anda harus memberikan nama buku beserta nilai string baru untuk mode perizinan.
3. Jalankan metode `updateLedgerPermissionsMode` dengan menyediakan objek permintaan sebagai parameter.

Contoh kode berikut mendemonstrasikan langkah sebelumnya.
`updateLedgerPermissionsMode`Permintaan

mengembalikan`UpdateLedgerPermissionsModeResult` objek yang telah memperbarui informasi tentang buku besar.

Example - Tetapkan mode izin standar

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerPermissionsModeRequest request = new UpdateLedgerPermissionsModeRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
UpdateLedgerPermissionsModeResult result = client.updateLedgerPermissionsMode(request);
```

Memperbarui mode izin buku besar (AWS CLI)

Tetapkan mode`STANDARD` izin ke`vehicle-registration` buku besar Anda.

Example

```
aws qldb update-ledger-permissions-mode --name vehicle-registration --permissions-mode
 STANDARD
```

## Migrasi ke mode izin standar

Untuk bermigrasi ke mode`STANDARD` izin, sebaiknya analisis pola akses QLDB Anda dan menambahkan kebijakan IAM yang memberi pengguna izin yang sesuai untuk mengakses sumber daya mereka.

Sebelum beralih ke mode`STANDARD` izin, Anda harus terlebih dahulu membuat semua kebijakan IAM dan tag tabel yang diperlukan. Jika tidak, mengganti mode izin dapat mengganggu pengguna hingga Anda membuat kebijakan IAM yang benar, atau mengembalikan mode izin kembali ke`ALLOW_ALL`. Untuk informasi tentang membuat kebijakan ini, lihat[Memulai dengan mode izin standar di Amazon QLDB](#).

Anda juga dapat menggunakan kebijakan[AWS terkelola untuk memberikan akses penuh ke semua sumber daya QLDB. Kebijakan yang`AmazonQLDBConsoleFullAccess` dikelola`AmazonQLDBFullAccess` dan dikelola mencakup semua tindakan QLDB, termasuk semua tindakan PartiQL. Melampirkan salah satu kebijakan ini ke prinsipal setara dengan mode`ALLOW_ALL` izin untuk prinsipal tersebut. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk Amazon QLDB](#).

# Menghapus buku besar

GunakanDeleteLedger operasi untuk menghapus buku besar dan semua isinya. Menghapus buku besar adalah operasi yang tidak dapat dipulihkan.

Jika perlindungan penghapusan diaktifkan untuk buku besar Anda, Anda harus menonaktifkannya terlebih dahulu sebelum dapat menghapus buku besar.

Saat Anda mengeluarkanDeleteLedger permintaan, status buku besar berubah dariACTIVE keDELETING. Mungkin perlu beberapa saat untuk menghapus buku besar, tergantung pada jumlah penyimpanan yang digunakannya. KetikaDeleteLedger operasi menyimpulkan, buku besar tidak lagi ada di QLDB.

Menghapus buku besar (Java)

Untuk menghapus buku besar menggunakanAWS SDK for Java

1. Ciptakan contoh dari kelas AmazonQLDB.
2. Buat instans dariDeleteLedgerRequest kelas dan berikan nama buku besar yang ingin Anda hapus.
3. Jalankan metode deleteLedger dengan menyediakan objek permintaan sebagai parameter.

Contoh kode berikut mendemonstrasikan langkah sebelumnya.

Example

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
DeleteLedgerRequest request = new DeleteLedgerRequest().withName(ledgerName);
DeleteLedgerResult result = client.deleteLedger(request);
```

Menghapus buku besar (AWS CLI)

Hapusvehicle-registration buku besar Anda.

Example

```
aws qldb delete-ledger --name vehicle-registration
```

# Daftar buku besar

`ListLedgers`Operasi mengembalikan informasi ringkasan dari semua buku besar QLDB untuk saat iniAkun AWS dan Wilayah.

Listing buku besar (Java)

Untuk mencantumkan buku besar di akun Anda menggunakanAWS SDK for Java

1. Buat instans dari kelas `AmazonQLDB`.
2. Buat instans dari kelas `ListLedgersRequest`.

   Jika Anda menerima nilai untuk`NextToken` respons dari`ListLedgers` panggilan sebelumnya, Anda harus memberikan nilai tersebut dalam permintaan ini untuk mendapatkan halaman hasil berikutnya.
3. Jalankan metode `listLedgers` dengan menyediakan objek permintaan sebagai parameter.
4. `listLedgers`Permintaan mengembalikan sebuah`ListLedgersResult` objek. Objek ini memiliki daftar`LedgerSummary` objek dan token pagination yang menunjukkan apakah ada lebih banyak hasil yang tersedia:

   - Jika`NextToken` kosong, halaman terakhir hasil telah diproses dan tidak ada lagi hasil.
   - Jika`NextToken` tidak kosong, ada lebih banyak hasil yang tersedia. Untuk mengambil halaman hasil berikutnya, gunakan nilai`NextToken` dalam`ListLedgers` panggilan berikutnya.

Contoh kode berikut mendemonstrasikan langkah sebelumnya.

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);
```

Listing buku besar (AWS CLI)

Daftar semua buku besar di saat iniAkun AWS dan Wilayah.

Example

```
aws qldb list-ledgers
```

# Membuat sumber daya Amazon QLDB denganAWS CloudFormation

Amazon QLDB terintegrasi denganAWS CloudFormation, yaitu layanan yang membantu Anda memodelkan dan mengaturAWS sumber daya agar Anda dapat menghemat waktu untuk membuat dan mengelola sumber daya dan infrastruktur Anda. Anda membuat templat yang menggambarkanAWS sumber daya yang Anda inginkan (seperti buku besar QLDB), dan memasokAWS CloudFormation persediaan dan mengonfigurasi sumber daya tersebut untuk Anda.

Saat menggunakanAWS CloudFormation, Anda dapat menggunakan kembali templat Anda untuk menyiapkan sumber daya QLDB secara konsisten dan berulang kali. Jelaskan sumber daya Anda satu kali, lalu sediakan sumber daya yang sama berulang kali dalam beberapa Akun AWS dan Wilayah.

## QLDB danAWS CloudFormation template

Untuk menyediakan dan mengonfigurasi sumber daya untuk QLDB dan layanan terkait, Anda harus memahami AWS CloudFormationtemplat. Templat adalah file teks dengan format JSON atau YAML. Templat ini menjelaskan sumber daya yang ingin Anda sediakan di tumpukan AWS CloudFormation Anda. Jika Anda tidak terbiasa dengan JSON atau YAML, Anda dapat menggunakan AWS CloudFormation Designer untuk membantu Anda memulai dengan templat AWS CloudFormation. Untuk informasi selengkapnya, lihat Apa yang dimaksud dengan AWS CloudFormation Designer? dalam Panduan Pengguna AWS CloudFormation.

QLDB mendukung menciptakan buku besar dan jurnal sungai diAWS CloudFormation. Untuk informasi selengkapnya, termasuk contoh templat JSON dan YAKL untuk buku besar dan streaming, lihat referensi tipe sumber daya berikut di PanduanAWS CloudFormation Pengguna Baris Perintah

- AWS: :QLDB: :Referensi Ledger
- AWS: :QLDB: :Stream Referensi

## Pelajari selengkapnya tentang AWS CloudFormation

Untuk mempelajari selengkapnya tentang AWS CloudFormation, lihat sumber daya berikut:

- AWS CloudFormation
- AWS CloudFormationPanduan Pengguna
- AWS CloudFormationReferensi API
- AWS CloudFormationPanduan Pengguna Baris Perintah

# Pemberian tag pada sumber daya Amazon QLDB

Tag merupakan label atribut kustom yang Anda atau AWS tetapkan ke sumber daya AWS. Setiap tanda memiliki dua bagian:

- Sebuah kunci tag (misalnya, `CostCenter`, `Environment`, atau `Project`). Kunci tag peka terhadap huruf besar dan kecil.
- Bidang opsional yang dikenal sebagai nilai tanda (misalnya, `111122223333` atau `Production`). Mengabaikan nilai tag sama saja dengan menggunakan string kosong. Seperti kunci tag, nilai tag peka huruf besar dan kecil.

Tag membantu Anda melakukan hal berikut:

- Identifikasi dan organisir sumber daya AWS Anda. Banyak penandaanLayanan AWS dukungan, sehingga Anda dapat menugaskan tanda yang sama ke sumber daya dari berbagai layanan untuk menunjukkan bahwa sumber daya tersebut terkait. Misalnya, Anda dapat menugaskan tanda yang sama ke buku besar Amazon QLDB yang Anda tetapkan ke sebuah bucket Amazon S3.
- Telusuri biaya AWS Anda. Anda mengaktifkan tag ini pada AWS Billing and Cost Management dasbor.AWS menggunakan tag untuk mengategorikan biaya Anda lalu mengirimkan laporan alokasi biaya bulanan kepada Anda. Untuk informasi selengkapnya, lihat Menggunakan tag alokasi biaya dalam Panduan Pengguna AWS Billing.
- Kontrol akses keAWS sumber daya Anda denganAWS Identity and Access Management (IAM). Untuk informasi, lihatKontrol akses berbasis atribut (ABAC) dengan QLDB di panduan pengembang ini dan Kontrol akses menggunakan tag IAM di Panduan Pengguna IAM.

Untuk tips menggunakan tanda, lihat postingan AWS Strategi Penandaan di blog AWS Jawaban.

Bagian berikut menyediakan informasi selengkapnya tentang tanda untuk Amazon QLDB.

Topik



## Sumber daya yang didukung di Amazon QLDB

Sumber daya berikut di Amazon QLDB mendukung penandaan:

- buku besar
- tabel
- aliran jurnal

Untuk informasi tentang menambahkan dan mengelola tag, lihat Mengola.

## Konvensi penamaan dan penggunaan tag

Kesepakatan penamaan dan penggunaan dasar berikut berlaku untuk menggunakan tanda dengan sumber daya Amazon QLDB:

- Setiap sumber daya dapat memiliki maksimum tanda 50.
- Untuk setiap sumber daya, setiap tanda kunci harus unik, dan setiap tanda kunci hanya dapat memiliki satu nilai.
- Panjang tanda kunci maksimum adalah 128 karakter Unicode dalam UTF-8.
- Panjang tanda nilai maksimum adalah 256 karakter Unicode dalam UTF-8.
- Karakter yang diperbolehkan adalah huruf, angka, spasi yang dapat ditampilkan di UTF-8, serta karakter berikut: . : + = @ _ / - (tanda hubung).
- Kunci dan nilai tag peka huruf besar dan kecil. Sebagai praktik terbaik, putuskan strategi untuk memanfaatkan tag dan terapkan strategi tersebut secara konsisten di semua jenis sumber daya. Misalnya, putuskan apakah akan menggunakan `Costcenter`, `costcenter`, atau `CostCenter` dan menggunakan kesepakatan yang sama untuk semua tag. Hindari penggunaan tag yang serupa dengan perlakuan kasus yang tidak konsisten.

- Prefiks `aws:` disimpan untuk penggunaan AWS. Anda tidak dapat mengedit atau menghapus kunci atau nilai tanda jika tanda memiliki kunci dengan`aws:` prefiks. Tanda dengan prefiks ini tidak dihitung terhadap tanda Anda per batas sumber daya.

## Mengola

Tag terdiri dari`Key` dan`Value` properti pada sumber daya. Anda dapat menggunakan konsol Amazon QLDBAWS CLI, atau API QLDB untuk menambahkan, mengedit, atau menghapus nilai untuk properti ini. Anda juga dapat menggunakan Editor untuk mengelola tanda.AWS Resource Groups

Untuk informasi tentang bekerja dengan tanda, lihat operasi API:

- ListTagsForResourcedalam Referensi API Amazon QLDB
- TagResourcedalam Referensi API Amazon QLDB
- UntagResourcedalam Referensi API Amazon QLDB

Untuk menggunakan panel penandaan QLDB (konsol)

1. Masuk keAWS Management Console, dan buka konsol Amazon QLDB di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih Buku besar.
3. Dalam daftar Buku Besar, pilih nama buku besar yang tagnya ingin Anda kelola.
4. Pada halaman detail buku besar, cari kartu Tag dan pilih Kelola tag.
5. Di halaman Kelola tanda, Anda dapat menambahkan, mengedit, atau menghapus tanda apa pun yang sesuai untuk buku besar Anda. Bila kunci dan nilai sesuai keinginan Anda, pilih Simpan.

## Pemberian tag pada sumber daya

Untuk sumber daya QLDB yang mendukung penandaan, Anda dapat menentukan tag saat Anda membuat sumber daya dengan menggunakanAWS CLI,, atau QLDB API.AWS Management Console Dengan menandai sumber daya saat sedang dibuat, Anda dapat menghilangkan kebutuhan untuk menjalankan skrip penandaan khusus setelah pembuatan sumber daya.

Setelah sumber daya ditandai, Anda dapat mengontrol akses ke sumber daya berdasarkan tag tersebut. Misalnya, Anda dapat memberikan akses penuh hanya ke sumber daya yang memiliki tanda tertentu. Untuk contoh kebijakan JSON, lihatAkses penuh ke semua tindakan berdasarkan tag tabel.

> **Note**
>
> Sumber daya tabel dan aliran tidak mewarisi tag sumber daya buku besar root mereka.

Anda juga dapat menentukan tag tabel dengan menentukan mereka dalam pernyataan`CREATE TABLE` PartiQL. Untuk mempelajari informasi lebih lanjut, lihat Menandai tabel.

# Keamanan di Amazon QLDB

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan  cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang berjalan Layanan AWS di dalamnya AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara berkala menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari Program kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku untuk Amazon QLDB, AWS lihat Layanan dalam Lingkup berdasarkan Program Kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh Layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, termasuk sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan QLDB. Topik berikut menunjukkan cara mengonfigurasi QLDB untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan Layanan AWS yang lain yang membantu Anda memantau dan mengamankan sumber daya QLDB Anda.

Topik

# Perlindungan data di Amazon QLDB

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di Amazon QLDB. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam Pertanyaan Umum Privasi Data. Lihat informasi tentang perlindungan data di Eropa di pos blog Model Tanggung Jawab Bersama dan GDPR AWS di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensil dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya. AWS Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi yang lebih lengkap tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-2.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan QLDB atau Layanan AWS lainnya menggunakan konsol, API AWS CLI, atau SDK. AWS Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan

supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

> **Note**
>
> Panduan tentang menghindari tag atau bidang bentuk bebas untuk informasi sensitif ini mengacu pada metadata sumber daya buku besar QLDB, bukan data yang disimpan dalam buku besar. Data Anda yang disimpan dalam sumber daya buku besar QLDB tidak digunakan untuk penagihan atau log diagnostik.

Topik



## Enkripsi saat istirahat di Amazon QLDB

Semua data yang disimpan di Amazon QLDB sepenuhnya dienkripsi saat istirahat secara default. Enkripsi QLDB saat istirahat memberikan keamanan yang ditingkatkan dengan mengenkripsi semua data buku besar saat istirahat menggunakan kunci enkripsi di (). AWS Key Management Service AWS KMS Fungsi ini membantu mengurangi beban operasional dan kompleksitas yang terlibat dalam melindungi data sensitif. Dengan enkripsi saat istirahat, Anda dapat membangun aplikasi buku besar yang sensitif terhadap keamanan yang memenuhi kepatuhan enkripsi dan persyaratan peraturan yang ketat.

Enkripsi saat istirahat terintegrasi dengan AWS KMS untuk mengelola kunci enkripsi yang digunakan untuk melindungi buku besar QLDB Anda. Untuk informasi selengkapnya AWS KMS, lihat AWS Key Management Service konsep di Panduan AWS Key Management Service Pengembang.

Di QLDB, Anda dapat menentukan jenis untuk setiap sumber daya buku AWS KMS key besar. Saat Anda membuat buku besar baru atau memperbarui buku besar yang ada, Anda dapat memilih salah satu dari jenis kunci KMS berikut untuk melindungi data buku besar Anda:

- Kunci milik AWS— Jenis enkripsi default. Kuncinya dimiliki oleh QLDB (tanpa biaya tambahan).
- Kunci yang dikelola pelanggan — Kunci disimpan di dalam Anda Akun AWS dan dibuat, dimiliki, dan dikelola oleh Anda. Anda memiliki kendali penuh atas kunci (AWS KMS dikenakan biaya).

> **Note**
>
> Amazon QLDB meluncurkan dukungan untuk pelanggan yang AWS KMS keys dikelola pada 22 Juli 2021. Buku besar apa pun yang dibuat sebelum peluncuran dilindungi secara Kunci milik AWS default, tetapi saat ini tidak memenuhi syarat untuk enkripsi saat istirahat menggunakan kunci yang dikelola pelanggan.
> Anda dapat melihat waktu pembuatan buku besar Anda di konsol QLDB.

Saat Anda mengakses buku besar, QLDB mendekripsi data secara transparan. Anda dapat beralih antara kunci yang dikelola pelanggan Kunci milik AWS dan pada waktu tertentu. Anda tidak perlu mengubah kode atau aplikasi apa pun untuk menggunakan atau mengelola data terenkripsi.

Anda dapat menentukan kunci enkripsi saat membuat buku besar baru atau mengubah kunci enkripsi pada buku besar yang ada dengan menggunakan AWS Management Console, QLDB API, atau (). AWS Command Line Interface AWS CLI Untuk informasi selengkapnya, lihat Menggunakan kunci yang dikelola pelanggan di Amazon QLDB.

> **Note**
>
> Secara default, Amazon QLDB secara otomatis mengaktifkan enkripsi saat istirahat Kunci milik AWS menggunakan tanpa biaya tambahan. Namun, AWS KMS biaya berlaku untuk menggunakan kunci yang dikelola pelanggan. Untuk informasi lebih lanjut tenngenai harga, lihat harga AWS Key Management Service.
> Enkripsi QLDB saat istirahat tersedia di semua Wilayah AWS tempat QLDB tersedia.

Topik

- Enkripsi saat istirahat: Cara kerjanya di Amazon QLDB
- Menggunakan kunci yang dikelola pelanggan di Amazon QLDB

## Enkripsi saat istirahat: Cara kerjanya di Amazon QLDB

Enkripsi QLDB saat istirahat mengenkripsi data Anda menggunakan Standar Enkripsi Lanjutan 256-bit (AES-256). Ini membantu mengamankan data Anda dari akses tidak sah ke penyimpanan yang mendasarinya. Semua data yang disimpan dalam buku besar QLDB dienkripsi saat istirahat secara default. Enkripsi sisi server transparan, yang berarti bahwa perubahan pada aplikasi tidak diperlukan.

Enkripsi saat istirahat terintegrasi dengan AWS Key Management Service (AWS KMS) untuk mengelola kunci enkripsi yang digunakan untuk melindungi buku besar QLDB Anda. Saat membuat buku besar baru atau memperbarui buku besar yang ada, Anda dapat memilih salah satu dari jenis AWS KMS kunci berikut:

- Kunci milik AWS— Jenis enkripsi default. Kuncinya dimiliki oleh QLDB (tanpa biaya tambahan).
- Kunci yang dikelola pelanggan — Kunci disimpan di dalam Anda Akun AWS dan dibuat, dimiliki, dan dikelola oleh Anda. Anda memiliki kendali penuh atas kunci (AWS KMS dikenakan biaya).

Topik



## Kunci milik AWS

Kunci milik AWS tidak disimpan di Anda Akun AWS. Mereka adalah bagian dari kumpulan kunci KMS yang AWS memiliki dan mengelola untuk digunakan dalam beberapa. Akun AWS Layanan AWS dapat digunakan Kunci milik AWS untuk melindungi data Anda.

Anda tidak perlu membuat atau mengelola Kunci milik AWS. Namun, Anda tidak dapat melihat atau melacak Kunci milik AWS, atau mengaudit penggunaannya. Anda tidak dikenakan biaya bulanan atau biaya penggunaan untuk Kunci milik AWS, dan mereka tidak dihitung terhadap AWS KMS kuota untuk akun Anda.

Untuk informasi lebih lanjut, lihat Kunci milik AWS dalam Panduan Pengembang AWS Key Management Service .

## Kunci yang dikelola pelanggan

Kunci yang dikelola pelanggan adalah kunci KMS Akun AWS yang Anda buat, miliki, dan kelola. Anda memiliki kontrol penuh atas kunci KMS ini. QLDB hanya mendukung kunci KMS enkripsi simetris.

Gunakan kunci yang dikelola pelanggan untuk mendapatkan fitur berikut:

- Menetapkan dan memelihara kebijakan utama, kebijakan IAM, dan hibah untuk mengontrol akses ke kunci
- Mengaktifkan dan menonaktifkan kunci
- Memutar bahan kriptografi untuk kuncinya
- Membuat tag kunci dan alias
- Menjadwalkan kunci untuk dihapus
- Mengimpor materi kunci Anda sendiri atau menggunakan toko kunci khusus yang Anda miliki dan kelola
- Menggunakan AWS CloudTrail dan Amazon CloudWatch Logs untuk melacak permintaan yang dikirim AWS KMS QLDB atas nama Anda

Untuk informasi selengkapnya, lihat Kunci terkelola pelanggan di Panduan AWS Key Management Service Pengembang.

Kunci yang dikelola pelanggan dikenakan biaya untuk setiap panggilan API, dan AWS KMS kuota berlaku untuk kunci KMS ini. Untuk informasi selengkapnya, lihat AWS KMS sumber daya atau permintaan kuota.

Saat Anda menentukan kunci yang dikelola pelanggan sebagai kunci KMS untuk buku besar, semua data buku besar di penyimpanan jurnal dan penyimpanan yang diindeks dilindungi dengan kunci yang dikelola pelanggan yang sama.

Kunci terkelola pelanggan yang tidak dapat diakses

Jika Anda menonaktifkan kunci terkelola pelanggan, menjadwalkan kunci untuk dihapus, atau mencabut hibah pada kunci, status enkripsi buku besar Anda menjadi. `KMS_KEY_INACCESSIBLE` Dalam keadaan ini, buku besar terganggu dan tidak menerima permintaan baca atau tulis apa pun. Kunci yang tidak dapat diakses mencegah semua pengguna dan layanan QLDB mengenkripsi atau mendekripsi data — dan melakukan operasi baca dan tulis di buku besar. QLDB harus memiliki akses ke kunci KMS Anda untuk memastikan bahwa Anda dapat terus mengakses buku besar Anda dan untuk mencegah kehilangan data.

> **Important**
>
> Buku besar yang rusak secara otomatis kembali ke status aktif setelah Anda mengembalikan hibah pada kunci, atau setelah Anda mengaktifkan kembali kunci yang dinonaktifkan.

Namun, menghapus kunci yang dikelola pelanggan tidak dapat diubah. Setelah kunci dihapus, Anda tidak dapat lagi mengakses buku besar yang dilindungi dengan kunci itu, dan data menjadi tidak dapat dipulihkan secara permanen.

Untuk memeriksa status enkripsi buku besar, gunakan AWS Management Console atau operasi DescribeLedgerAPI.

Bagaimana Amazon QLDB menggunakan hibah di AWS KMS

QLDB membutuhkan hibah untuk menggunakan kunci yang dikelola pelanggan Anda. Saat Anda membuat buku besar yang dilindungi dengan kunci yang dikelola pelanggan, QLDB membuat hibah atas nama Anda dengan mengirimkan permintaan ke. CreateGrant AWS KMS Hibah AWS KMS digunakan untuk memberikan akses QLDB ke kunci KMS pada pelanggan. Akun AWS Untuk informasi selengkapnya, lihat Menggunakan Hibah di Panduan AWS Key Management Service Pengembang.

QLDB mewajibkan hibah untuk menggunakan kunci yang dikelola pelanggan Anda untuk operasi berikut: AWS KMS

- DescribeKey— Verifikasi bahwa kunci KMS enkripsi simetris yang ditentukan valid.
- GenerateDataKey— Hasilkan kunci data simetris unik yang digunakan QLDB untuk mengenkripsi data saat istirahat di buku besar Anda.
- Dekripsi — Dekripsi kunci data yang dienkripsi oleh kunci yang dikelola pelanggan Anda.
- Enkripsi - Enkripsi plaintext ke dalam ciphertext menggunakan kunci yang dikelola pelanggan Anda.

Anda dapat mencabut hibah untuk menghapus akses layanan ke kunci yang dikelola pelanggan kapan saja. Jika Anda melakukannya, kunci menjadi tidak dapat diakses, dan QLDB kehilangan akses ke salah satu data buku besar yang dilindungi oleh kunci yang dikelola pelanggan. Dalam keadaan ini, buku besar terganggu dan tidak menerima permintaan baca atau tulis apa pun sampai Anda mengembalikan hibah pada kunci.

Memulihkan hibah di AWS KMS

Untuk memulihkan hibah pada kunci yang dikelola pelanggan dan memulihkan akses ke buku besar di QLDB, Anda dapat memperbarui buku besar dan menentukan kunci KMS yang sama. Untuk petunjuk, lihat Memperbarui buku besar yang ada AWS KMS key.

### Enkripsi saat istirahat pertimbangan

Pertimbangkan hal berikut saat Anda menggunakan enkripsi saat istirahat di QLDB:

- Enkripsi sisi server saat istirahat diaktifkan secara default pada semua data buku besar QLDB dan tidak dapat dinonaktifkan. Anda tidak dapat mengenkripsi hanya sebagian data dalam buku besar.
- Enkripsi saat diam hanya mengenkripsi data saat statis (saat diam) pada media penyimpanan persisten. Jika keamanan data menjadi perhatian untuk data dalam perjalanan atau data yang digunakan, Anda mungkin perlu mengambil langkah-langkah tambahan sebagai berikut:
  - Data dalam perjalanan: Semua data Anda di QLDB dienkripsi saat transit. Secara default, komunikasi ke dan dari QLDB menggunakan protokol HTTPS, yang melindungi lalu lintas jaringan dengan menggunakan enkripsi Secure Sockets Layer (SSL) /Transport Layer Security (TLS).
  - Data yang digunakan: Lindungi data Anda sebelum mengirimnya ke QLDB dengan menggunakan enkripsi sisi klien.

Untuk mempelajari cara menerapkan kunci yang dikelola pelanggan untuk buku besar, lanjutkan keMenggunakan kunci yang dikelola pelanggan di Amazon QLDB.

## Menggunakan kunci yang dikelola pelanggan di Amazon QLDB

Anda dapat menggunakan AWS Management Console, AWS Command Line Interface (AWS CLI), atau QLDB API untuk menentukan buku besar baru dan buku besar AWS KMS key yang ada di Amazon QLDB. Topik berikut menjelaskan cara mengelola dan memantau penggunaan kunci terkelola pelanggan Anda di QLDB.

### Topik



### Prasyarat

Sebelum Anda dapat melindungi buku besar QLDB dengan kunci yang dikelola pelanggan, Anda harus terlebih dahulu membuat kunci di (). AWS Key Management Service AWS KMS Anda juga

harus menentukan kebijakan kunci yang memungkinkan QLDB membuat hibah atas AWS KMS key nama Anda.

Membuat kunci yang dikelola pelanggan

Untuk membuat kunci terkelola pelanggan, ikuti langkah-langkah dalam Membuat kunci KMS enkripsi simetris di Panduan AWS Key Management Service Pengembang. QLDB tidak mendukung kunci asimetris.

Mengatur kebijakan kunci

Kebijakan utama adalah cara utama untuk mengontrol akses ke kunci yang dikelola pelanggan AWS KMS. Setiap kunci yang dikelola pelanggan harus memiliki persis satu kebijakan utama. Pernyataan dalam dokumen kebijakan kunci menentukan siapa yang memiliki izin untuk menggunakan kunci KMS dan bagaimana mereka dapat menggunakannya. Untuk informasi selengkapnya, lihat Menggunakan kebijakan utama di AWS KMS.

Anda dapat menentukan kebijakan kunci saat membuat kunci terkelola pelanggan. Untuk mengubah kebijakan kunci untuk kunci terkelola pelanggan yang sudah ada, lihat Mengubah kebijakan kunci.

Untuk mengizinkan QLDB menggunakan kunci terkelola pelanggan Anda, kebijakan utama harus menyertakan izin untuk tindakan berikut: AWS KMS

- kms: CreateGrant — Menambahkan hibah ke kunci yang dikelola pelanggan. Memberikan akses kontrol ke kunci KMS tertentu.

  Saat Anda membuat atau memperbarui buku besar dengan kunci terkelola pelanggan tertentu, QLDB membuat hibah yang memungkinkan akses ke operasi hibah yang diperlukan. Operasi hibah meliputi:

  - GenerateDataKey
  - Dekripsi
  - Enkripsi
- kms: DescribeKey — Mengembalikan informasi rinci tentang kunci yang dikelola pelanggan. QLDB menggunakan informasi ini untuk memvalidasi kunci.

Contoh kebijakan utama

Berikut ini adalah contoh kebijakan utama yang dapat Anda gunakan untuk QLDB. Kebijakan ini memungkinkan prinsipal yang berwenang untuk

menggunakan QLDB dari akun 111122223333 untuk memanggil dan operasi pada sumber daya. `DescribeKey` `CreateGrant` `arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`

*Untuk menggunakan kebijakan ini, ganti us-east-1, 111122223333, dan 1234abcd-12ab-34cd-56ef-1234567890ab dalam contoh dengan informasi Anda sendiri.*

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

Menentukan AWS KMS key untuk buku besar baru

Ikuti langkah-langkah ini untuk menentukan kunci KMS saat Anda membuat buku besar baru menggunakan konsol QLDB atau. AWS CLI

Anda dapat menentukan kunci yang dikelola pelanggan menggunakan ID, alias, atau Nama Sumber Daya Amazon (ARN). Untuk mempelajari selengkapnya, lihat Pengidentifikasi kunci (KeyId) di Panduan AWS Key Management Service Pengembang.

> **Note**
>
> Kunci Lintas Wilayah tidak didukung. Kunci KMS yang ditentukan harus Wilayah AWS sama dengan buku besar Anda.

Membuat buku besar (konsol)

1. Masuk ke AWS Management Console, dan buka konsol QLDB Amazon di https://console.aws.amazon.com/qldb.
2. Pilih Buat Buku Besar.
3. Pada halaman Create Ledger, lakukan hal berikut:
   - Informasi buku besar — Masukkan nama Buku Besar yang unik di antara semua buku besar di saat ini Akun AWS dan Wilayah.
   - Mode izin - Pilih mode izin untuk ditetapkan ke buku besar:
     - Izinkan semua
     - Standar (Direkomendasikan)
   - Enkripsi data saat istirahat - Pilih jenis kunci KMS yang akan digunakan untuk enkripsi saat istirahat:
     - Gunakan kunci KMS yang AWS dimiliki — Gunakan kunci KMS yang dimiliki dan dikelola oleh AWS atas nama Anda. Ini adalah opsi default dan tidak memerlukan pengaturan tambahan.
     - Pilih kunci yang berbeda — Gunakan AWS KMS kunci KMS enkripsi simetris di akun yang Anda buat, miliki, dan kelola.

       Untuk membuat kunci baru dengan menggunakan AWS KMS konsol, pilih Buat AWS KMS kunci. Untuk informasi selengkapnya, lihat Membuat kunci enkripsi simetris KMS di Panduan Developer AWS Key Management Service .

       Untuk menggunakan kunci KMS yang ada, pilih salah satu dari daftar dropdown atau tentukan ARN kunci KMS.
4. Ketika pengaturan seperti yang Anda inginkan, pilih Buat buku besar.

   Anda dapat mengakses buku besar QLDB Anda ketika statusnya menjadi Aktif. Ini dapat memakan waktu beberapa menit.

Membuat buku besar ()AWS CLI

Gunakan AWS CLI untuk membuat buku besar di QLDB dengan Kunci milik AWS default atau kunci yang dikelola pelanggan.

Example — Untuk membuat buku besar dengan default Kunci milik AWS

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

Example — Untuk membuat buku besar dengan kunci yang dikelola pelanggan

```
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
```

Memperbarui buku besar yang ada AWS KMS key

Anda juga dapat menggunakan konsol QLDB atau AWS CLI untuk memperbarui kunci KMS buku besar yang ada ke atau kunci Kunci milik AWS yang dikelola pelanggan kapan saja.

> **Note**
>
> Amazon QLDB meluncurkan dukungan untuk pelanggan yang AWS KMS keys dikelola pada 22 Juli 2021. Buku besar apa pun yang dibuat sebelum peluncuran dilindungi secara Kunci milik AWS default, tetapi saat ini tidak memenuhi syarat untuk enkripsi saat istirahat menggunakan kunci yang dikelola pelanggan.
> Anda dapat melihat waktu pembuatan buku besar Anda di konsol QLDB.

Perubahan utama dalam QLDB bersifat asinkron. Buku besar dapat diakses sepenuhnya tanpa dampak kinerja apa pun saat perubahan kunci sedang diproses. Jumlah waktu yang diperlukan untuk memperbarui kunci bervariasi tergantung pada ukuran buku besar.

Anda dapat menentukan kunci yang dikelola pelanggan menggunakan ID, alias, atau Nama Sumber Daya Amazon (ARN). Untuk mempelajari selengkapnya, lihat Pengidentifikasi kunci (KeyId) di Panduan AWS Key Management Service Pengembang.

> **Note**
>
> Kunci Lintas Wilayah tidak didukung. Kunci KMS yang ditentukan harus Wilayah AWS sama dengan buku besar Anda.

Memperbarui buku besar (konsol)

1. Masuk ke AWS Management Console, dan buka konsol QLDB Amazon di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih Buku Besar.
3. Dalam daftar buku besar, pilih buku besar yang ingin Anda perbarui, lalu pilih Edit buku besar.
4. Pada halaman Edit buku besar, pilih jenis kunci KMS yang akan digunakan untuk enkripsi saat istirahat:

   - Gunakan kunci KMS yang AWS dimiliki — Gunakan kunci KMS yang dimiliki dan dikelola oleh AWS atas nama Anda. Ini adalah opsi default dan tidak memerlukan pengaturan tambahan.
   - Pilih kunci yang berbeda — Gunakan AWS KMS kunci KMS enkripsi simetris di akun yang Anda buat, miliki, dan kelola.

     Untuk membuat kunci baru dengan menggunakan AWS KMS konsol, pilih Buat AWS KMS kunci. Untuk informasi selengkapnya, lihat Membuat kunci enkripsi simetris KMS di Panduan Developer AWS Key Management Service .

     Untuk menggunakan kunci KMS yang ada, pilih salah satu dari daftar dropdown atau tentukan ARN kunci KMS.
5. Pilih Konfirmasi perubahan.

Memperbarui buku besar ()AWS CLI

Gunakan AWS CLI untuk memperbarui buku besar yang ada di QLDB dengan Kunci milik AWS default atau kunci yang dikelola pelanggan.

Example — Untuk memperbarui buku besar dengan default Kunci milik AWS

```
aws qldb update-ledger --name my-example-ledger --kms-key AWS_OWNED_KMS_KEY
```

Example — Untuk memperbarui buku besar dengan kunci yang dikelola pelanggan

```
aws qldb update-ledger \
    --name my-example-ledger \
    --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Memantau Anda AWS KMS keys

Jika Anda menggunakan kunci yang dikelola pelanggan untuk melindungi buku besar QLDB Amazon Anda, Anda dapat menggunakan AWS CloudTrailatau CloudWatch Amazon Logs untuk melacak permintaan yang dikirimkan QLDB atas nama Anda. AWS KMS Untuk informasi selengkapnya, lihat Pemantauan AWS KMS keys di Panduan AWS Key Management Service Pengembang.

Contoh berikut adalah entri CloudTrail log untuk operasi`CreateGrant`,,`GenerateDataKey`, `DecryptEncrypt`, dan`DescribeKey`.

CreateGrant

Saat Anda menentukan kunci yang dikelola pelanggan untuk melindungi buku besar Anda, QLDB `CreateGrant` mengirimkan permintaan AWS KMS ke atas nama Anda untuk mengizinkan akses ke kunci KMS Anda. Selain itu, QLDB menggunakan `RetireGrant` operasi untuk menghapus hibah saat Anda menghapus buku besar.

Hibah yang dibuat QLDB khusus untuk buku besar. Prinsipal dalam `CreateGrant` permintaan adalah pengguna yang membuat tabel.

Peristiwa yang mencatat operasi `CreateGrant` serupa dengan peristiwa contoh berikut. Parameter termasuk Nama Sumber Daya Amazon (ARN) dari kunci yang dikelola pelanggan, pokok penerima hibah dan kepala pensiun (layanan QLDB), dan operasi yang dicakup oleh hibah.

```
{
    "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

Saat Anda menentukan kunci yang dikelola pelanggan untuk melindungi buku besar Anda, QLDB membuat kunci data yang unik. Ini mengirimkan `GenerateDataKey` permintaan ke AWS KMS yang menentukan kunci yang dikelola pelanggan untuk buku besar.

Peristiwa yang mencatat operasi `GenerateDataKey` serupa dengan peristiwa contoh berikut. Pengguna adalah akun layanan QLDB. Parameter termasuk ARN dari kunci yang dikelola pelanggan, penentu kunci data yang memerlukan panjang 32-byte, dan konteks enkripsi yang mengidentifikasi simpul hierarki kunci internal.

```
{
    "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"
}
```

### Dekripsi

Saat Anda mengakses buku besar, QLDB memanggil `Decrypt` operasi untuk mendekripsi kunci data tersimpan buku besar sehingga dapat mengakses data terenkripsi dalam buku besar.

Peristiwa yang mencatat operasi `Decrypt` serupa dengan peristiwa contoh berikut. Pengguna adalah akun layanan QLDB. Parameter termasuk ARN dari kunci yang dikelola pelanggan dan konteks enkripsi yang mengidentifikasi simpul hierarki kunci internal.

```
{
    "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"
}
```

### Enkripsi

QLDB memanggil operasi untuk mengenkripsi plaintext ke `Encrypt` dalam ciphertext menggunakan kunci yang dikelola pelanggan Anda.

Peristiwa yang mencatat operasi `Encrypt` serupa dengan peristiwa contoh berikut. Pengguna adalah akun layanan QLDB. Parameter termasuk ARN dari kunci yang dikelola pelanggan dan konteks enkripsi yang menentukan ID unik internal buku besar.

```
{
    "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 memanggil `DescribeKey` operasi untuk menentukan apakah kunci KMS yang Anda tentukan ada di dan Wilayah. Akun AWS

Peristiwa yang mencatat operasi `DescribeKey` serupa dengan peristiwa contoh berikut. Prinsipal adalah pengguna di Anda Akun AWS yang menentukan kunci KMS. Parameter termasuk ARN dari kunci yang dikelola pelanggan.

```
{
    "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"
```

```
}
```

## Enkripsi dalam perjalanan di Amazon QLDB

Amazon QLDB hanya menerima koneksi aman yang menggunakan protokol HTTPS, yang melindungi lalu lintas jaringan dengan menggunakan Secure Sockets Layer (SSL) /Transport Layer Security (TLS). Enkripsi dalam perjalanan menyediakan lapisan perlindungan data tambahan dengan mengenkripsi data Anda saat melakukan perjalanan ke dan dari QLDB. Kebijakan organisasi, peraturan industri atau pemerintah, dan persyaratan kepatuhan sering kali memerlukan penggunaan enkripsi dalam perjalanan untuk meningkatkan keamanan data aplikasi Anda ketika mereka mengirimkan data melalui jaringan.

QLDB juga menawarkan titik akhir FIPS di Wilayah tertentu. Tidak seperti titik akhir AWS standar, titik akhir FIPS menggunakan pustaka perangkat lunak TLS yang sesuai dengan Standar Pemrosesan Informasi Federal (FIPS) 140-2. Titik akhir ini mungkin diperlukan oleh korporasi yang berinteraksi dengan pemerintah Amerika Serikat. Untuk informasi selengkapnya, lihat titik akhir FIPS di. Referensi Umum AWS Untuk daftar lengkap Wilayah dan titik akhir yang tersedia untuk QLDB, lihat titik akhir dan kuota Amazon QLDB.

# Identity and Access Management untuk Amazon QLDB

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya QLDB. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- Audiens
- Mengautentikasi dengan identitas
- Mengelola akses menggunakan kebijakan
- Bagaimana Amazon QLDB bekerja dengan IAM
- Memulai dengan mode izin standar di Amazon QLDB
- Contoh kebijakan berbasis identitas untuk Amazon QLDB
- Pencegahan confused deputy lintas layanan

- AWS kebijakan terkelola untuk Amazon QLDB
- Memecahkan masalah identitas dan akses QLDB Amazon

## Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di QLDB.

Pengguna layanan - Jika Anda menggunakan layanan QLDB untuk melakukan pekerjaan Anda, maka administrator Anda memberi Anda kredensyal dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur QLDB untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di QLDB, lihat. Memecahkan masalah identitas dan akses QLDB Amazon

Administrator layanan - Jika Anda bertanggung jawab atas sumber daya QLDB di perusahaan Anda, Anda mungkin memiliki akses penuh ke QLDB. Tugas Anda adalah menentukan fitur dan sumber daya QLDB mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep Basic IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM dengan QLDB, lihat. Bagaimana Amazon QLDB bekerja dengan IAM

Administrator IAM - Jika Anda seorang administrator IAM, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke QLDB. Untuk melihat contoh kebijakan berbasis identitas QLDB yang dapat Anda gunakan di IAM, lihat. Contoh kebijakan berbasis identitas untuk Amazon QLDB

## Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensil yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensyal Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya

menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensyal Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan AWS API di Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Menggunakan autentikasi multi-faktor (MFA) dalam AWS dalam Panduan Pengguna IAM.

### Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

### Identitas gabungan

Sebagai praktik terbaik, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensyal sementara.

Identitas federasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, direktori Pusat Identitas AWS Directory Service, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensyal yang disediakan melalui sumber identitas. Ketika

identitas federasi mengakses Akun AWS, mereka mengambil peran, dan peran memberikan kredensyal sementara.

Untuk manajemen akses terpusat, kami sarankan Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua aplikasi Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat Apakah itu Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

## Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial sementara, bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin ke grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat Kapan harus membuat pengguna IAM (bukan peran) dalam Panduan Pengguna IAM.

## Peran IAM

Peran IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara AWS Management Console dengan beralih peran. Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat Menggunakan peran IAM dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat Set izin dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Sebagai contoh, ketika Anda memanggil suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses maju.
  - Peran layanan – Peran layanan adalah peran IAM yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan

menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensyal sementara untuk aplikasi yang berjalan pada instans EC2 dan membuat atau permintaan API. AWS CLI AWS Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan AWS peran ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan dalam instans Amazon EC2 dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat Kapan harus membuat peran IAM (bukan pengguna) dalam Panduan Pengguna IAM.

## Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasinya. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

### Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan yang dikelola atau kebijakan inline, lihat Memilih antara kebijakan yang dikelola dan kebijakan inline dalam Panduan Pengguna IAM.

### Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

### Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, lihat Gambaran umum daftar kontrol akses (ACL) dalam Panduan Developer Amazon Simple Storage Service.

### Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM (pengguna IAM atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) — SCP adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di. AWS Organizations AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur di organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations .
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

### Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

# Bagaimana Amazon QLDB bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke QLDB, pelajari fitur IAM apa yang tersedia untuk digunakan dengan QLDB.

Fitur IAM yang dapat Anda gunakan dengan Amazon QLDB

| Fitur IAM | Dukungan QLDB |
|---|---|
| Kebijakan berbasis identitas | Ya |
| Kebijakan berbasis sumber daya | Tidak |
| Tindakan kebijakan | Ya |
| Sumber daya kebijakan | Ya |
| Kunci kondisi kebijakan | Ya |
| ACL | Tidak |
| ABAC (tanda dalam kebijakan) | Ya |
| Kredensial sementara | Ya |
| Izin prinsipal | Tidak |
| Peran layanan | Ya |
| Peran terkait layanan | Tidak |

Untuk mendapatkan tampilan tingkat tinggi tentang bagaimana QLDB dan Layanan AWS lainnya bekerja dengan sebagian besar fitur IAM, Layanan AWS lihat yang bekerja dengan IAM di Panduan Pengguna IAM.

## Kebijakan berbasis identitas untuk QLDB

| | |
|---|---|
| Mendukung kebijakan berbasis identitas | Ya |

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan secara spesifik apakah tindakan dan sumber daya diizinkan atau ditolak, serta kondisi yang menjadi dasar dikabulkan atau ditolaknya tindakan tersebut. Anda tidak dapat menentukan secara spesifik prinsipal dalam sebuah kebijakan berbasis identitas karena prinsipal berlaku bagi pengguna atau peran yang melekat kepadanya. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk QLDB

Untuk melihat contoh kebijakan berbasis identitas QLDB, lihat. Contoh kebijakan berbasis identitas untuk Amazon QLDB

## Kebijakan berbasis sumber daya dalam QLDB

| | |
|---|---|
| Mendukung kebijakan berbasis sumber daya | Tidak |

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan secara spesifik seluruh akun atau entitas IAM di akun lain sebagai prinsipal dalam kebijakan berbasis sumber daya. Menambahkan prinsipal akun silang ke kebijakan berbasis sumber daya hanya setengah dari membangun hubungan kepercayaan. Ketika prinsipal dan sumber daya berbeda Akun AWS, administrator IAM di akun tepercaya juga harus memberikan izin entitas utama (pengguna atau peran) untuk mengakses sumber daya. Mereka memberikan izin dengan melampirkan kebijakan berbasis identitas kepada

entitas. Namun, jika kebijakan berbasis sumber daya memberikan akses ke prinsipal dalam akun yang sama, tidak diperlukan kebijakan berbasis identitas tambahan. Untuk informasi selengkapnya, lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.

## Tindakan kebijakan untuk QLDB

| | |
|---|---|
| Mendukung tindakan kebijakan | Ya |

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama dengan operasi AWS API terkait. Ada beberapa pengecualian, misalnya tindakan hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Menyertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Untuk melihat daftar tindakan QLDB, lihat Tindakan yang ditentukan oleh Amazon QLDB di Referensi Otorisasi Layanan.

Tindakan kebijakan di QLDB menggunakan awalan berikut sebelum tindakan:

```
qldb
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan tersebut dengan koma.

```
"Action": [
      "qldb:action1",
      "qldb:action2"
]
```

Anda juga dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut:

```
"Action": "qldb:Describe*"
```

Untuk berinteraksi dengan API data transaksional QLDB (Sesi QLDB) dengan menjalankan pernyataan PartiQL pada buku besar, Anda harus memberikan izin untuk tindakan sebagai berikut. `SendCommand`

```
"Action": "qldb:SendCommand"
```

Untuk buku besar dalam mode `STANDARD` izin, lihat untuk izin tambahan yang diperlukan Referensi izin PartiQL untuk setiap perintah PartiQL.

Untuk melihat contoh kebijakan berbasis identitas QLDB, lihat. Contoh kebijakan berbasis identitas untuk Amazon QLDB

## Sumber daya kebijakan untuk QLDB

| | |
|---|---|
| Mendukung sumber daya kebijakan | Ya |

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar jenis sumber daya QLDB dan ARNnya, lihat Sumber daya yang ditentukan oleh Amazon QLDB di Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang ditentukan oleh Amazon QLDB.

Di QLDB, sumber daya utama adalah buku besar. QLDB juga mendukung jenis sumber daya tambahan: tabel dan aliran. Namun, Anda dapat membuat tabel dan aliran hanya dalam konteks buku besar yang ada.

Tabel QLDB adalah pandangan terwujud dari koleksi revisi dokumen yang tidak teratur dari jurnal buku besar. Dalam mode `STANDARD` izin buku besar, Anda harus membuat kebijakan IAM yang memberikan izin untuk menjalankan pernyataan PartiQL pada sumber daya tabel ini. Dengan izin pada sumber daya tabel, Anda dapat menjalankan pernyataan yang mengakses status tabel saat ini. Anda juga dapat menanyakan riwayat revisi tabel dengan menggunakan `history()` fungsi bawaan. Untuk mempelajari selengkapnya, lihat Memulai dengan mode izin standar di Amazon QLDB.

> **Note**
>
> `CREATE TABLE`Pernyataan membuat tabel dengan ID unik dan nama tabel yang disediakan. Nama tabel yang disediakan harus unik di antara semua tabel aktif. Namun, QLDB memungkinkan Anda menonaktifkan tabel, jadi mungkin ada beberapa tabel tidak aktif yang berbagi nama tabel yang sama. Oleh karena itu, ARN sumber daya tabel merujuk ke ID unik yang ditetapkan sistem daripada nama tabel yang ditentukan pengguna.

Setiap buku besar juga menyediakan sumber daya katalog yang ditentukan sistem yang dapat Anda kueri untuk mencantumkan semua tabel dan indeks dalam buku besar. Untuk informasi selengkapnya tentang model objek data QLDB, lihat. Konsep inti dan terminologi di Amazon QLDB

Sumber daya ini memiliki ARN unik yang terkait dengannya, seperti yang ditunjukkan pada tabel berikut.

| Jenis Sumber Daya | ARN |
| --- | --- |
| ledger | `arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}` |
| table | `arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}/table/${TableId}` |

| Jenis Sumber Daya | ARN |
| --- | --- |
| catalog | `arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}/information_schema/user_tables` |
| stream | `arn:${Partition}:qldb:${Region}:${Account}:stream/${LedgerName}/${StreamId}` |

Misalnya, untuk menentukan `myExampleLedger` sumber daya dalam pernyataan Anda, gunakan ARN berikut.

```
"Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
```

Untuk menentukan beberapa sumber daya dalam satu pernyataan, pisahkan ARN dengan koma.

```
"Resource": [
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger1",
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger2"
]
```

Untuk melihat contoh kebijakan berbasis identitas QLDB, lihat. Contoh kebijakan berbasis identitas untuk Amazon QLDB

## Kunci kondisi kebijakan untuk QLDB

| | |
| --- | --- |
| Mendukung kunci kondisi kebijakan khusus layanan | Ya |

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen `Condition` (atau blok `Condition`) akan memungkinkan Anda menentukan kondisi yang menjadi dasar suatu pernyataan berlaku. Elemen `Condition` bersifat opsional. Anda dapat

membuat ekspresi bersyarat yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam sebuah pernyataan, atau beberapa kunci dalam elemen `Condition` tunggal, maka AWS akan mengevaluasinya menggunakan operasi `AND` logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan `OR` operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tag yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, lihat Elemen kebijakan IAM: variabel dan tag dalam Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi khusus layanan. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

Untuk melihat daftar kunci kondisi QLDB, lihat Kunci kondisi untuk Amazon QLDB di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh Amazon QLDB.

`PartiQLDropTable`Tindakan `PartiQLDropIndex` dan mendukung kunci `qldb:Purge` kondisi. Kunci kondisi ini menyaring akses dengan nilai `purge` yang ditentukan dalam pernyataan PartiQL`DROP`. Namun, QLDB saat ini `purge = true` hanya mendukung `DROP INDEX` untuk pernyataan, `purge = false` dan untuk pernyataan. `DROP TABLE` Tindakan QLDB lainnya mendukung beberapa kunci kondisi global.

Untuk melihat contoh kebijakan berbasis identitas QLDB, lihat. Contoh kebijakan berbasis identitas untuk Amazon QLDB

## Daftar kontrol akses (ACL) di QLDB

| | |
|---|---|
| Mendukung ACL | Tidak |

Daftar kontrol akses (ACL) mengendalikan pengguna utama mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

## Kontrol akses berbasis atribut (ABAC) dengan QLDB

| | |
|---|---|
| Mendukung ABAC (tanda dalam kebijakan) | Ya |

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut. Dalam AWS, atribut ini disebut tag. Anda dapat melampirkan tag ke entitas IAM (pengguna atau peran) dan ke banyak AWS sumber daya. Penandaan ke entitas dan sumber daya adalah langkah pertama dari ABAC. Kemudian rancanglah kebijakan ABAC untuk mengizinkan operasi ketika tag milik prinsipal cocok dengan tag yang ada di sumber daya yang ingin diakses.

ABAC sangat berguna di lingkungan yang berkembang dengan cepat dan berguna di situasi saat manajemen kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tag di elemen kondisi dari kebijakan menggunakan kunci kondisi `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi untuk hanya beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Apa itu ABAC? dalam Panduan Pengguna IAM. Untuk melihat tutorial yang menguraikan langkah-langkah pengaturan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang menandai sumber daya QLDB, lihat. Pemberian tag pada sumber daya Amazon QLDB

Untuk melihat contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat Memperbarui buku besar QLDB berdasarkan tag.

## Menggunakan kredensyal Sementara dengan QLDB

| | |
|---|---|
| Mendukung penggunaan kredensial sementara | Ya |

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensil sementara. Untuk informasi tambahan, termasuk yang Layanan AWS bekerja dengan kredensyal sementara, lihat Layanan AWS yang bekerja dengan IAM di Panduan Pengguna IAM.

Anda menggunakan kredensi sementara jika Anda masuk AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Misalnya, ketika Anda mengakses AWS menggunakan tautan masuk tunggal (SSO) perusahaan Anda, proses tersebut secara otomatis membuat kredensil sementara. Anda juga akan secara otomatis membuat kredensial sementara ketika Anda masuk ke konsol sebagai seorang pengguna lalu beralih peran. Untuk informasi selengkapnya tentang peralihan peran, lihat Peralihan peran (konsol) dalam Panduan Pengguna IAM.

Anda dapat membuat kredensyal sementara secara manual menggunakan API AWS CLI atau AWS . Anda kemudian dapat menggunakan kredensi sementara tersebut untuk mengakses. AWS AWS merekomendasikan agar Anda secara dinamis menghasilkan kredensi sementara alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM.

## Izin utama lintas layanan untuk QLDB

| | |
|---|---|
| Mendukung sesi akses maju (FAS) | Tidak |

Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses maju.

## Peran layanan untuk QLDB

| | |
|---|---|
| Mendukung peran layanan | Ya |

Peran layanan adalah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak fungsionalitas QLDB. Edit peran layanan hanya jika QLDB memberikan panduan untuk melakukannya.

QLDB mendukung peran layanan untuk operasi `StreamJournalToKinesis` dan API, seperti yang dijelaskan di `ExportJournalToS3` bagian berikut.

Memilih peran IAM di QLDB

Saat Anda mengekspor atau mengalirkan blok jurnal di QLDB, Anda harus memilih peran untuk mengizinkan QLDB menulis objek ke tujuan yang diberikan atas nama Anda. Jika sebelumnya Anda telah membuat peran layanan, QLDB memberi Anda daftar peran yang dapat dipilih. Penting untuk memilih peran yang memungkinkan akses untuk menulis ke bucket Amazon S3 yang ditentukan untuk ekspor, atau ke sumber daya Amazon Kinesis Data Streams yang ditentukan untuk aliran. Untuk informasi selengkapnya, lihat Izin ekspor jurnal di QLDB atau Izin streaming di QLDB.

> **Note**
>
> Untuk meneruskan peran ke QLDB saat meminta ekspor atau streaming jurnal, Anda harus memiliki izin untuk melakukan `iam:PassRole` tindakan pada sumber daya peran IAM. Ini adalah tambahan untuk izin untuk melakukan `qldb:ExportJournalToS3` pada sumber daya buku besar QLDB, atau `qldb:StreamJournalToKinesis` pada subsumber daya aliran QLDB.

## Peran terkait layanan untuk QLDB

| | |
|---|---|
| Mendukung peran terkait layanan | Tidak |

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang membuat atau mengelola peran terkait layanan, lihat Layanan AWS bahwa bekerja dengan IAM. Cari layanan dalam tabel yang memiliki `Yes` di kolom Peran terkait layanan. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

# Memulai dengan mode izin standar di Amazon QLDB

Gunakan bagian ini untuk memulai mode izin standar di Amazon QLDB. Bagian ini menyediakan tabel referensi untuk membantu Anda saat menulis kebijakan berbasis identitas di AWS Identity and Access Management (IAM) untuk tindakan PartiQL dan sumber daya tabel di QLDB. Ini juga mencakup step-by-step tutorial untuk membuat kebijakan izin di IAM, dan instruksi untuk menemukan tabel ARN dan membuat tag tabel di QLDB.

Topik

- STANDARDMode izin
- Referensi izin PartiQL
- Menemukan ID tabel dan ARN
- Menandai tabel
- Tutorial mulai cepat: Membuat kebijakan izin

## **`STANDARD`**Mode izin

QLDB sekarang mendukung mode izin untuk sumber `STANDARD` daya buku besar. Mode izin standar memungkinkan kontrol akses dengan perincian yang lebih halus untuk buku besar, tabel, dan perintah PartiQL. Secara default, mode ini menolak semua permintaan untuk menjalankan perintah PartiQL pada tabel apa pun dalam buku besar.

> **Note**
>
> Sebelumnya, satu-satunya mode izin yang tersedia untuk buku besar adalah. `ALLOW_ALL` `ALLOW_ALL`Mode ini memungkinkan kontrol akses dengan perincian tingkat API untuk buku besar, dan terus didukung — tetapi tidak direkomendasikan — untuk buku besar QLDB. Mode ini memungkinkan pengguna yang memiliki izin `SendCommand` API untuk menjalankan semua perintah PartiQL pada tabel apa pun di buku besar yang ditentukan oleh kebijakan izin (karenanya, “izinkan semua” perintah PartiQL).
> Anda dapat mengubah mode izin buku besar yang ada dari `ALLOW_ALL` ke. `STANDARD` Untuk informasi, lihat Migrasi ke mode izin standar.

Untuk mengizinkan perintah dalam mode standar, Anda harus membuat kebijakan izin di IAM untuk sumber daya tabel tertentu dan tindakan PartiQL. Ini merupakan tambahan dari izin `SendCommand` API untuk buku besar. Untuk memfasilitasi kebijakan dalam mode ini, QLDB memperkenalkan serangkaian tindakan IAM untuk perintah PartiQL, dan Nama Sumber Daya Amazon (ARN) untuk tabel QLDB. Untuk informasi selengkapnya tentang model objek data QLDB, lihat. Konsep inti dan terminologi di Amazon QLDB

## Referensi izin PartiQL

Tabel berikut mencantumkan setiap perintah QLDB PartiQL, tindakan IAM terkait yang harus Anda berikan izin untuk menjalankan perintah, AWS dan sumber daya yang dapat Anda berikan izin. Anda menentukan tindakan dalam bidang `Action` kebijakan, dan Anda menentukan nilai sumber daya pada bidang `Resource` kebijakan.

**Important**

- Kebijakan IAM yang memberikan izin untuk perintah PartiQL ini hanya berlaku untuk buku besar Anda jika mode `STANDARD` izin ditetapkan ke buku besar. Kebijakan tersebut tidak berlaku untuk buku besar dalam mode `ALLOW_ALL` izin.

  Untuk mempelajari cara menentukan mode izin saat Anda membuat atau memperbarui buku besar, lihatOperasi dasar untuk buku besar Amazon QLDB, atau Langkah 1: Membuat buku besar baru di Memulai konsol.
- Untuk menjalankan perintah PartiQL pada buku besar, Anda juga harus memberikan izin untuk tindakan API untuk sumber daya `SendCommand` buku besar. Ini adalah tambahan untuk tindakan PartiQL dan sumber daya tabel yang tercantum dalam tabel berikut. Untuk informasi selengkapnya, lihat Menjalankan transaksi data.

Perintah Amazon QLDB PartiQL dan izin yang diperlukan

| Perintah | Izin yang diperlukan (tindakan IAM) | Sumber daya | Tindakan bergantung |
|---|---|---|---|
| CREATE TABLE | qldb:PartiQLCreateTable | arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/* | qldb:TagResource (untuk menandai pada pembuatan) |
| MEJA DROP | qldb:PartiQLDropTable | arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id* | |
| TABEL 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* | |
| HAPUS<br>FROM-REMOVE (untuk seluruh dokumen) | qldb:PartiQLDelete | arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id* | qldb:PartiQLSelect |

| Perintah | Izin yang diperlukan (tindakan IAM) | Sumber daya | Tindakan bergantung |
|---|---|---|---|
| SISIPKAN | qldb:PartiQLInsert | arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id* | |
| PERBARUI<br>DARI (INSERT, HAPUS, atau SET) | qldb:PartiQLUpdate | arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id* | qldb:PartiQLSelect |
| REDACT_REVISI(prosedur tersimpan) | qldb:PartiQLRedact | arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id* | |
| PILIH DARI table_name | qldb:PartiQLSelect | arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id* | |
| PILIH DARI information_schema.user_tables | qldb:PartiQLSelect | arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /information_schema/user_tables | |

| Perintah | Izin yang diperlukan (tindakan IAM) | Sumber daya | Tindakan bergantung |
|---|---|---|---|
| PILIH DARI riwayat (table_name) | qldb:PartiQLHistoryFunction | arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id* | |

Untuk contoh dokumen kebijakan IAM yang memberikan izin untuk perintah PartiQL ini, lanjutkan ke atau lihat. Tutorial mulai cepat: Membuat kebijakan izin Contoh kebijakan berbasis identitas untuk Amazon QLDB

## Menemukan ID tabel dan ARN

Anda dapat menemukan ID tabel dengan menggunakan AWS Management Console atau dengan menanyakan tabel information_schema.user_tables. Untuk melihat detail tabel di konsol, atau untuk menanyakan tabel katalog sistem ini, Anda harus memiliki `SELECT` izin pada sumber daya katalog sistem. Misalnya, untuk menemukan ID tabel `Vehicle` tabel, Anda dapat menjalankan pernyataan berikut.

```
SELECT * FROM information_schema.user_tables
WHERE name = 'Vehicle'
```

Query ini mengembalikan hasil dalam format yang mirip dengan contoh berikut.

```
{
    tableId: "Au1EiThbt8s0z9wM26REZN",
    name: "Vehicle",
    indexes: [
        { indexId: "Djg2nt0yIs2GY0T29Kud1z", expr: "[VIN]", status: "ONLINE" },
        { indexId: "4tPW3fUhaVhDinRgKRLhGU", expr: "[LicensePlateNumber]", status:
 "BUILDING" }
    ],
    status: "ACTIVE"
```

```
}
```

Untuk memberikan izin untuk menjalankan pernyataan PartiQL pada tabel, Anda menentukan sumber daya tabel dalam format ARN berikut.

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

Berikut ini adalah contoh dari tabel ARN untuk ID tabel. `Au1EiThbt8s0z9wM26REZN`

```
arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/Au1EiThbt8s0z9wM26REZN
```

Menggunakan konsol

Anda juga dapat menggunakan konsol QLDB untuk menemukan tabel ARN.

Untuk menemukan ARN tabel (konsol)

1. Masuk ke AWS Management Console, dan buka konsol QLDB Amazon di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih Buku Besar.
3. Dalam daftar Buku Besar, pilih nama buku besar yang tabelnya ARN ingin Anda temukan.
4. Pada halaman detail buku besar, di bawah tab Tabel, cari nama tabel yang ARN yang ingin Anda temukan. Untuk menyalin ARN, pilih ikon salin ( ) di sebelahnya.

## Menandai tabel

Anda dapat menandai sumber daya tabel Anda. Untuk mengelola tag untuk tabel yang ada, gunakan operasi AWS Management Console atau API `TagResource``UntagResource`, dan`ListTagsForResource`. Untuk informasi selengkapnya, lihat Pemberian tag pada sumber daya Amazon QLDB.

> **Note**
>
> Sumber daya tabel tidak mewarisi tag sumber daya buku besar root mereka.

Menandai tabel pada pembuatan saat ini didukung untuk buku besar dalam mode `STANDARD` izin saja.

Anda juga dapat menentukan tag tabel saat Anda membuat tabel dengan menggunakan konsol QLDB atau dengan menentukannya dalam pernyataan PartiQL. `CREATE TABLE` Contoh berikut membuat tabel bernama `Vehicle` dengan tag`environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

Menandai tabel pada pembuatan membutuhkan akses ke `qldb:TagResource` tindakan `qldb:PartiQLCreateTable` dan tindakan.

Dengan menandai sumber daya saat sedang dibuat, Anda dapat menghilangkan kebutuhan untuk menjalankan skrip penandaan khusus setelah pembuatan sumber daya. Setelah tabel ditandai, Anda dapat mengontrol akses ke tabel berdasarkan tag tersebut. Misalnya, Anda dapat memberikan akses penuh hanya ke tabel yang memiliki tag tertentu. Untuk contoh kebijakan JSON, lihatAkses penuh ke semua tindakan berdasarkan tag tabel.

Menggunakan konsol

Anda juga dapat menggunakan konsol QLDB untuk menentukan tag tabel saat Anda membuat tabel.

Untuk menandai tabel pada pembuatan (konsol)

1. Masuk ke AWS Management Console, dan buka konsol QLDB Amazon di https://console.aws.amazon.com/qldb.
2. Di panel navigasi, pilih Buku Besar.
3. Dalam daftar Buku Besar, pilih nama buku besar yang ingin Anda buat tabel.
4. Pada halaman detail buku besar, di bawah tab Tabel, pilih Buat tabel.
5. Pada halaman Buat tabel, lakukan hal berikut:

   - Nama tabel - Masukkan nama tabel.
   - Tag - Tambahkan metadata ke tabel dengan melampirkan tag sebagai pasangan kunci-nilai. Anda dapat menambahkan tag ke tabel Anda untuk membantu mengatur dan mengidentifikasi mereka.

     Pilih Tambahkan tag, lalu masukkan pasangan nilai kunci apa pun yang sesuai.

6. Jika pengaturan sudah sesuai keinginan Anda, pilih Buat tabel.

## Tutorial mulai cepat: Membuat kebijakan izin

Tutorial ini memandu Anda melalui langkah-langkah untuk membuat kebijakan izin di IAM untuk buku besar QLDB Amazon dalam mode izin. `STANDARD` Anda kemudian dapat menetapkan izin untuk pengguna, grup, atau peran Anda.

Untuk lebih banyak contoh dokumen kebijakan IAM yang memberikan izin ke perintah PartiQL dan sumber tabel, lihat. Contoh kebijakan berbasis identitas untuk Amazon QLDB

Topik

- Prasyarat
- Membuat kebijakan hanya-baca
- Membuat kebijakan akses penuh
- Membuat kebijakan hanya-baca untuk tabel tertentu
- Menetapkan izin

### Prasyarat

Sebelum memulai, pastikan Anda melakukan hal berikut:

1. Ikuti petunjuk AWS pengaturan diMengakses Amazon QLDB, jika Anda belum melakukannya. Langkah-langkah ini termasuk mendaftar AWS dan membuat pengguna administratif.
2. Buat buku besar baru dan pilih mode `STANDARD` izin untuk buku besar. Untuk mempelajari caranya, lihat Langkah 1: Membuat buku besar baru di Memulai dengan konsol, atauOperasi dasar untuk buku besar Amazon QLDB.

### Membuat kebijakan hanya-baca

Untuk menggunakan editor kebijakan JSON untuk membuat kebijakan hanya-baca untuk semua tabel dalam buku besar dalam mode izin standar, lakukan hal berikut:

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di kolom navigasi di sebelah kiri, pilih Kebijakan.

Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul laman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.

3. Di bagian atas halaman, pilih Buat kebijakan.
4. Pilih tab JSON.
5. Salin dan tempel dokumen kebijakan JSON berikut. Kebijakan contoh ini memberikan akses hanya-baca ke semua tabel dalam buku besar.

   Untuk menggunakan kebijakan ini, ganti *us-east-1*, 123456789012, dan dalam contoh dengan informasi Anda sendiri. *myExampleLedger*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "QLDBSendCommandPermission",
      "Effect": "Allow",
      "Action": "qldb:SendCommand",
      "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
    },
    {
      "Sid": "QLDBPartiQLReadOnlyPermissions",
      "Effect": "Allow",
      "Action": [
        "qldb:PartiQLSelect",
        "qldb:PartiQLHistoryFunction"
      ],
      "Resource": [
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
      ]
    }
  ]
}
```

6. Pilih Tinjau kebijakan.

> **Note**
>
> Anda dapat berpindah antara tab Editor visual dan JSON kapan pun. Namun, apabila Anda melakukan perubahan atau memilih Tinjau kebijakan pada tab Editor visual, IAM dapat merestrukturisasi kebijakan Anda untuk menjadikannya optimal bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. Pada halaman Peninjauan Kebijakan, ketikkan Nama dan Deskripsi opsional untuk kebijakan yang sedang Anda buat. Tinjau Summary (Ringkasan) kebijakan untuk melihat izin yang diberikan oleh kebijakan Anda. Kemudian pilih Buat kebijakan untuk menyimpan pekerjaan Anda.

Membuat kebijakan akses penuh

Untuk membuat kebijakan akses penuh untuk semua tabel dalam buku besar QLDB dalam mode izin standar, lakukan hal berikut:

- Ulangi langkah sebelumnya menggunakan dokumen kebijakan berikut. Kebijakan contoh ini memberikan akses ke semua perintah PartiQL untuk semua tabel dalam buku besar, dengan menggunakan wildcard (*) untuk mencakup semua tindakan PartiQL dan semua sumber daya di bawah buku besar.

  > **Warning**
  >
  > Ini adalah contoh penggunaan karakter wildcard (*) untuk memungkinkan semua tindakan PartiQL, termasuk operasi administratif dan baca/tulis pada semua tabel dalam buku besar QLDB. Sebaliknya, ini adalah praktik terbaik untuk secara eksplisit menentukan setiap tindakan yang akan diberikan, dan hanya apa yang dibutuhkan pengguna, peran, atau grup itu.

  Untuk menggunakan kebijakan ini, ganti *us-east-1*, 123456789012, dan dalam contoh dengan informasi Anda sendiri. *myExampleLedger*

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "QLDBSendCommandPermission",
      "Effect": "Allow",
      "Action": "qldb:SendCommand",
      "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
    },
    {
      "Sid": "QLDBPartiQLFullPermissions",
      "Effect": "Allow",
      "Action": [
        "qldb:PartiQL*"
      ],
      "Resource": [
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
      ]
    }
  ]
}
```

Membuat kebijakan hanya-baca untuk tabel tertentu

Untuk membuat kebijakan akses hanya-baca untuk tabel tertentu dalam buku besar QLDB dalam mode izin standar, lakukan hal berikut:

1. Temukan ARN untuk tabel dengan menggunakan AWS Management Console atau dengan menanyakan tabel katalog sistem. `information_schema.user_tables` Untuk petunjuk, lihat Menemukan ID tabel dan ARN.
2. Gunakan tabel ARN untuk membuat kebijakan yang memungkinkan akses hanya-baca ke tabel. Untuk melakukan ini, ulangi langkah-langkah sebelumnya menggunakan dokumen kebijakan berikut.

   Kebijakan contoh ini memberikan akses hanya-baca ke tabel yang ditentukan saja. Dalam contoh ini, ID tabel adalah`Au1EiThbt8s0z9wM26REZN`. *Untuk menggunakan kebijakan ini, ganti us-east-1, 123456789012,, dan Au1 8S0Z9WM26Rezn dalam contoh dengan informasi Anda myExampleLedgersendiri. EiThbt*

   ```
   {
   ```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "QLDBSendCommandPermission",
      "Effect": "Allow",
      "Action": "qldb:SendCommand",
      "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
    },
    {
      "Sid": "QLDBPartiQLReadOnlyPermissions",
      "Effect": "Allow",
      "Action": [
        "qldb:PartiQLSelect",
        "qldb:PartiQLHistoryFunction"
      ],
      "Resource": [
        "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
table/Au1EiThbt8s0z9wM26REZN"
      ]
    }
  ]
}
```

Menetapkan izin

Setelah membuat kebijakan izin QLDB, Anda kemudian menetapkan izin sebagai berikut.

Untuk memberikan akses, menambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

  Buat rangkaian izin. Ikuti instruksi di Buat rangkaian izin di Panduan Pengguna AWS IAM Identity Center .

- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti instruksi dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) di Panduan Pengguna IAM.

- Pengguna IAM:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam Membuat peran untuk pengguna IAM dalam Panduan Pengguna IAM.

- (Tidak disarankan) Pasang kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti instruksi dalam Menambahkan izin ke pengguna (konsol) dalam Panduan Pengguna IAM.

## Contoh kebijakan berbasis identitas untuk Amazon QLDB

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya QLDB. Mereka juga tidak dapat melakukan tugas dengan menggunakan AWS Management Console, AWS Command Line Interface (AWS CLI), atau AWS API. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian akan dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh QLDB, termasuk format ARN untuk setiap jenis sumber daya, lihat Kunci tindakan, sumber daya, dan kondisi untuk Amazon QLDB di Referensi Otorisasi Layanan.

Daftar Isi

    - Izinkan akses untuk membuat tabel berdasarkan tag permintaan
- Mengekspor jurnal ke ember Amazon S3
- Streaming jurnal ke Kinesis Data Streams
- Memperbarui buku besar QLDB berdasarkan tag

## Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya QLDB di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan yang dikelola AWS atau Kebijakan yang dikelola AWS untuk fungsi tugas dalam Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat Kebijakan dan izin dalam IAM dalam Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Kondisi dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang

dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat Validasi kebijakan IAM Access Analyzer dalam Panduan Pengguna IAM.

- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi akses API yang dilindungi MFA dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan dalam IAM dalam Panduan Pengguna IAM.

## Menggunakan konsol QLDB

Untuk mengakses konsol QLDB Amazon, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang sumber daya QLDB di Anda. Akun AWS Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna atau peran) dengan kebijakan tersebut.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau AWS API. Sebagai gantinya, izinkan akses hanya ke tindakan yang sesuai dengan operasi API yang coba mereka lakukan.

Untuk memastikan bahwa pengguna dan peran memiliki akses penuh ke konsol QLDB dan semua fitur-fiturnya, lampirkan kebijakan terkelola AWS berikut ke entitas. Untuk informasi selengkapnyaAWS kebijakan terkelola untuk Amazon QLDB, lihat, dan Menambahkan izin ke pengguna di Panduan Pengguna IAM.

```
AmazonQLDBConsoleFullAccess
```

Izin riwayat kueri

Selain izin QLDB, beberapa fitur konsol memerlukan izin untuk Layanan Metadata Kueri Database (awalan layanan:). `dbqms` Ini adalah layanan internal saja yang mengelola kueri terbaru dan tersimpan di editor kueri konsol untuk QLDB dan lainnya. Layanan AWSUntuk daftar lengkap tindakan API DBQMS, lihat Layanan Metadata Kueri Database di Referensi Otorisasi Layanan.

Untuk mengizinkan izin riwayat kueri, Anda dapat menggunakan kebijakan AWS terkelola ConsoleFullAccessAmazonQlDB. Kebijakan ini menggunakan wildcard (`dbqms:*`) untuk mengizinkan semua tindakan DBQMS untuk semua sumber daya.

Atau, Anda dapat membuat kebijakan IAM kustom dan menyertakan tindakan DBQMS berikut. Editor kueri PartiQL di konsol QLDB memerlukan izin untuk menggunakan tindakan ini untuk fitur riwayat kueri.

```
dbqms:CreateFavoriteQuery
dbqms:CreateQueryHistory
dbqms:DeleteFavoriteQueries
dbqms:DeleteQueryHistory
dbqms:DescribeFavoriteQueries
dbqms:DescribeQueryHistory
dbqms:UpdateFavoriteQuery
```

Izin konsol akses penuh tanpa riwayat kueri

Untuk mengizinkan akses penuh ke konsol QLDB tanpa izin riwayat kueri, Anda dapat membuat kebijakan IAM khusus yang mengecualikan semua tindakan DBQMS. Misalnya, dokumen kebijakan berikut memungkinkan izin yang sama yang diberikan oleh kebijakan AWS terkelola AmazonQLDB ConsoleFullAccess, kecuali tindakan yang dimulai dengan awalan layanan. `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"
        }
      }
    }
  ]
```

```
}
```

## Mengizinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. 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": "*"
        }
    ]
}
```

## Menjalankan transaksi data

Untuk berinteraksi dengan API data transaksional QLDB (Sesi QLDB) dengan menjalankan pernyataan PartiQL pada buku besar, Anda harus memberikan izin untuk tindakan API. `SendCommand` Dokumen JSON berikut adalah contoh kebijakan yang memberikan izin hanya untuk tindakan `SendCommand` API pada buku besar. `myExampleLedger`

Untuk menggunakan kebijakan ini, ganti *`us-east-1`*, 123456789012*`,`* dan dalam contoh dengan informasi Anda sendiri. *`myExampleLedger`*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        }
    ]
}
```

Jika `myExampleLedger` menggunakan mode `ALLOW_ALL` izin, kebijakan ini memberikan izin untuk menjalankan semua perintah PartiQL pada tabel mana pun di buku besar.

Anda juga dapat menggunakan kebijakan AWS terkelola untuk memberikan akses penuh ke semua sumber daya QLDB. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk Amazon QLDB.

Izin standar untuk tindakan PartiQL dan sumber daya tabel

Untuk buku besar dalam mode `STANDARD` izin, Anda dapat merujuk ke dokumen kebijakan IAM berikut sebagai contoh pemberian izin PartiQL yang sesuai. Untuk daftar izin yang diperlukan untuk setiap perintah PartiQL, lihat. Referensi izin PartiQL

Topik

- Akses penuh ke semua tindakan
- Akses penuh ke semua tindakan berdasarkan tag tabel
- Akses baca/tulis
- Akses hanya-baca

- Akses hanya-baca ke tabel tertentu
- Izinkan akses untuk membuat tabel
- Izinkan akses untuk membuat tabel berdasarkan tag permintaan

Akses penuh ke semua tindakan

Dokumen kebijakan JSON berikut memberikan akses penuh untuk menggunakan semua perintah PartiQL pada semua tabel di. `myExampleLedger` Kebijakan ini menghasilkan efek yang sama seperti menggunakan mode `ALLOW_ALL` izin untuk buku besar.

Untuk menggunakan kebijakan ini, ganti *us-east-1*, 123456789012, dan dalam contoh dengan informasi Anda sendiri. *myExampleLedger*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLFullPermissions",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLCreateIndex",
                "qldb:PartiQLDropIndex",
                "qldb:PartiQLCreateTable",
                "qldb:PartiQLDropTable",
                "qldb:PartiQLUndropTable",
                "qldb:PartiQLDelete",
                "qldb:PartiQLInsert",
                "qldb:PartiQLUpdate",
                "qldb:PartiQLRedact",
                "qldb:PartiQLSelect",
                "qldb:PartiQLHistoryFunction"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
```

```
            ]
        }
    ]
}
```

Akses penuh ke semua tindakan berdasarkan tag tabel

Dokumen kebijakan JSON berikut menggunakan kondisi yang didasarkan pada tag sumber daya tabel untuk memberikan akses penuh untuk menggunakan semua perintah PartiQL pada semua tabel di. `myExampleLedger` Izin diberikan hanya jika tag tabel `environment` memiliki nilai`development`.

> **Warning**
>
> Ini adalah contoh penggunaan karakter wildcard (*) untuk memungkinkan semua tindakan PartiQL, termasuk operasi administratif dan baca/tulis pada semua tabel dalam buku besar QLDB. Sebaliknya, ini adalah praktik terbaik untuk secara eksplisit menentukan setiap tindakan yang akan diberikan, dan hanya apa yang dibutuhkan pengguna, peran, atau grup itu.

Untuk menggunakan kebijakan ini, ganti *us-east-1*, 123456789012, dan dalam contoh dengan informasi Anda sendiri. *myExampleLedger*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLFullPermissionsBasedOnTags",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQL*"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
```

```
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
            ],
            "Condition": {
                "StringEquals": { "aws:ResourceTag/environment": "development" }
            }
        }
    ]
}
```

Akses baca/tulis

Dokumen kebijakan JSON berikut memberikan izin untuk memilih, menyisipkan, memperbarui, dan menghapus data pada semua tabel. `myExampleLedger` Kebijakan ini tidak memberikan izin untuk menyunting data atau mengubah skema, misalnya, membuat dan menghapus tabel dan indeks.

> **Note**
>
> UPDATEPernyataan memerlukan izin untuk kedua `qldb:PartiQLSelect` tindakan `qldb:PartiQLUpdate` dan pada tabel yang sedang dimodifikasi. Saat Anda menjalankan UPDATE pernyataan, ia melakukan operasi baca selain operasi pembaruan. Memerlukan kedua tindakan memastikan bahwa hanya pengguna yang diizinkan membaca isi tabel yang diberikan UPDATE izin.
>
> Demikian pula, DELETE pernyataan memerlukan izin untuk `qldb:PartiQLSelect` tindakan `qldb:PartiQLDelete` dan tindakan.

Untuk menggunakan kebijakan ini, ganti *us-east-1*, 123456789012*,* dan dalam contoh dengan informasi Anda sendiri. *myExampleLedger*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLReadWritePermissions",
```

```
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLDelete",
                "qldb:PartiQLInsert",
                "qldb:PartiQLUpdate",
                "qldb:PartiQLSelect",
                "qldb:PartiQLHistoryFunction"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
            ]
        }
    ]
}
```

Akses hanya-baca

Dokumen kebijakan JSON berikut memberikan izin hanya-baca pada semua tabel di. `myExampleLedger` Untuk menggunakan kebijakan ini, ganti *us-east-1*, 123456789012, dan dalam contoh dengan informasi Anda sendiri. *myExampleLedger*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLReadOnlyPermissions",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLSelect",
                "qldb:PartiQLHistoryFunction"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
```

```
            ]
        }
    ]
}
```

Akses hanya-baca ke tabel tertentu

Dokumen kebijakan JSON berikut memberikan izin hanya-baca pada tabel tertentu di. `myExampleLedger` Dalam contoh ini, ID tabel adalah`Au1EiThbt8s0z9wM26REZN`.

*Untuk menggunakan kebijakan ini, ganti us-east-1, 123456789012,, dan Au1 8S0Z9WM26Rezn dalam contoh dengan informasi Anda myExampleLedgersendiri. EiThbt*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLReadOnlyPermissionsOnTable",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLSelect",
                "qldb:PartiQLHistoryFunction"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
table/Au1EiThbt8s0z9wM26REZN"
            ]
        }
    ]
}
```

Izinkan akses untuk membuat tabel

Dokumen kebijakan JSON berikut memberikan izin untuk membuat tabel di. `myExampleLedger` `qldb:PartiQLCreateTable`Tindakan ini memerlukan izin untuk jenis sumber daya tabel. Namun,

ID tabel tabel baru tidak diketahui pada saat Anda menjalankan `CREATE TABLE` pernyataan. Jadi, kebijakan yang memberikan `qldb:PartiQLCreateTable` izin harus menggunakan wildcard (*) dalam tabel ARN untuk menentukan sumber daya.

Untuk menggunakan kebijakan ini, ganti *us-east-1*, 123456789012, dan dalam contoh dengan informasi Anda sendiri. *myExampleLedger*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLCreateTablePermission",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLCreateTable"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*"
            ]
        }
    ]
}
```

Izinkan akses untuk membuat tabel berdasarkan tag permintaan

Dokumen kebijakan JSON berikut menggunakan kondisi berdasarkan kunci `aws:RequestTag` konteks untuk memberikan izin untuk membuat tabel dimyExampleLedger. Izin diberikan hanya jika tag permintaan `environment` memiliki nilaidevelopment. Menandai tabel pada pembuatan membutuhkan akses ke `qldb:TagResource` tindakan `qldb:PartiQLCreateTable` dan tindakan. Untuk mempelajari cara menandai tabel pada pembuatan, lihatMenandai tabel.

Untuk menggunakan kebijakan ini, ganti *us-east-1*, 123456789012, dan dalam contoh dengan informasi Anda sendiri. *myExampleLedger*

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLCreateTablePermission",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLCreateTable",
                "qldb:TagResource"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*"
            ],
            "Condition": {
                "StringEquals": { "aws:RequestTag/environment": "development" }
            }
        }
    ]
}
```

## Mengekspor jurnal ke ember Amazon S3

### Langkah 1: Izin ekspor jurnal QLDB

Dalam contoh berikut, Anda memberi pengguna Akun AWS izin untuk melakukan `qldb:ExportJournalToS3` tindakan pada sumber daya buku besar QLDB. Anda juga memberikan izin untuk melakukan `iam:PassRole` tindakan pada sumber daya peran IAM yang ingin diteruskan ke layanan QLDB. Ini diperlukan untuk semua permintaan ekspor jurnal.

*Untuk menggunakan kebijakan ini, ganti us-east-1, 123456789012 myExampleLedger, dan qldb-s3-export dalam contoh dengan informasi Anda sendiri.*

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

Langkah 2: Izin bucket Amazon S3

Dalam contoh berikut, Anda menggunakan peran IAM untuk memberikan akses QLDB untuk menulis ke salah satu bucket Amazon S3 Anda. `DOC-EXAMPLE-BUCKET` Ini juga diperlukan untuk semua ekspor jurnal QLDB.

Selain memberikan izin, kebijakan juga memberikan `s3:PutObject` izin untuk kemampuan menyetel `s3:PutObjectAcl` izin daftar kontrol akses (ACL) untuk objek.

Untuk menggunakan kebijakan ini, ganti DOC-EXAMPLE-BUCKET dalam contoh dengan nama bucket Amazon S3 Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
```

```
    ]
}
```

Kemudian, Anda melampirkan kebijakan izin ini ke peran IAM yang dapat diasumsikan QLDB untuk mengakses bucket Amazon S3 Anda. Dokumen JSON berikut adalah contoh kebijakan kepercayaan yang memungkinkan QLDB untuk mengambil peran IAM untuk sumber daya QLDB apa pun di akun saja. `123456789012`

Untuk menggunakan kebijakan ini, ganti *`us-east-1 dan`* 123456789012 dalam contoh dengan informasi Anda sendiri.

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

## Streaming jurnal ke Kinesis Data Streams

### Langkah 1: Izin aliran jurnal QLDB

Dalam contoh berikut, Anda memberi pengguna Akun AWS izin untuk melakukan `qldb:StreamJournalToKinesis` tindakan pada semua subresource aliran QLDB dalam buku besar. Anda juga memberikan izin untuk melakukan `iam:PassRole` tindakan pada sumber daya peran IAM yang ingin diteruskan ke layanan QLDB. Ini diperlukan untuk semua permintaan aliran jurnal.

Untuk menggunakan kebijakan ini, ganti *us-east-1*, 123456789012 *myExampleLedger,* dan dalam contoh dengan informasi Anda sendiri. *qldb-kinesis-stream*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalStreamPermission",
            "Effect": "Allow",
            "Action": "qldb:StreamJournalToKinesis",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:stream/myExampleLedger/*"
        },
        {
            "Sid": "IAMPassRolePermission",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/qldb-kinesis-stream",
            "Condition": {
               "StringEquals": {
                  "iam:PassedToService": "qldb.amazonaws.com"
               }
            }
        }
    ]
}
```

Langkah 2: Izin Kinesis Data Streams

Dalam contoh berikut, Anda menggunakan peran IAM untuk memberikan akses QLDB untuk menulis catatan data ke aliran data Amazon Kinesis,. *stream-for-qldb* Ini juga diperlukan untuk semua aliran jurnal QLDB.

Untuk menggunakan kebijakan ini, ganti *us-east-1*, 123456789012*,* dan dalam contoh dengan informasi Anda sendiri. *stream-for-qldb*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBStreamKinesisPermissions",
            "Action": [ "kinesis:PutRecord*", "kinesis:DescribeStream",
 "kinesis:ListShards" ],
```

```
            "Effect": "Allow",
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-for-qldb"
        }
    ]
}
```

Kemudian, Anda melampirkan kebijakan izin ini ke peran IAM yang dapat diasumsikan QLDB untuk mengakses aliran data Kinesis Anda. Dokumen JSON berikut adalah contoh kebijakan kepercayaan yang memungkinkan QLDB untuk mengambil peran IAM untuk aliran QLDB apa pun di akun untuk buku besar saja. `123456789012` `myExampleLedger`

Untuk menggunakan kebijakan ini, ganti *`us-east-1`*, 123456789012*,* dan dalam contoh dengan informasi Anda sendiri. *`myExampleLedger`*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "qldb.amazonaws.com"
            },
            "Action": [ "sts:AssumeRole" ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:qldb:us-
east-1:123456789012:stream/myExampleLedger/*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

## Memperbarui buku besar QLDB berdasarkan tag

Anda dapat menggunakan kondisi dalam kebijakan berbasis identitas untuk mengontrol akses ke sumber daya QLDB berdasarkan tag. Contoh ini menunjukkan bagaimana Anda dapat membuat kebijakan yang memungkinkan memperbarui buku besar. Namun, izin diberikan hanya jika tag buku

besar `Owner` memiliki nilai nama pengguna pengguna tersebut. Kebijakan ini juga memberi izin yang diperlukan untuk menyelesaikan tindakan ini pada konsol tersebut.

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

Anda dapat melampirkan kebijakan ini ke pengguna di akun Anda. Jika pengguna bernama `richard-roe` mencoba memperbarui buku besar QLDB, buku besar harus diberi tag atau. `Owner=richard-roe` `owner=richard-roe` Jika tidak, aksesnya akan ditolak. Kunci tag kondisi `Owner` cocok dengan keduanya `Owner` dan `owner` karena nama kunci kondisi tidak peka huruf besar/kecil. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Kondisi dalam Panduan Pengguna IAM.

## Pencegahan confused deputy lintas layanan

Masalah confused deputy adalah masalah keamanan saat entitas yang tidak memiliki izin untuk melakukan suatu tindakan dapat memaksa entitas yang lebih berhak untuk melakukan tindakan tersebut. Pada tahun AWS, peniruan lintas layanan dapat mengakibatkan masalah wakil yang membingungkan.

Peniruan identitas lintas layanan dapat terjadi ketika satu layanan (layanan yang dipanggil) memanggil layanan lain (layanan yang dipanggil). Layanan pemanggilan dapat dimanipulasi menggunakan izinnya untuk bertindak pada sumber daya pelanggan lain dengan cara yang seharusnya tidak dilakukannya kecuali bila memiliki izin untuk mengakses. Untuk mencegah masalah wakil yang membingungkan, AWS sediakan alat yang membantu Anda melindungi data Anda untuk semua layanan dengan prinsip layanan yang telah diberikan akses ke sumber daya di akun Anda.

Sebaiknya gunakan kunci konteks kondisi `aws:SourceAccount`global `aws:SourceArn`dan global dalam kebijakan sumber daya untuk membatasi izin yang diberikan Amazon QLDB layanan lain ke sumber daya. Jika Anda menggunakan kedua kunci konteks kondisi global, `aws:SourceAccount` nilai dan akun dalam `aws:SourceArn` nilai harus menggunakan ID akun yang sama saat digunakan dalam pernyataan kebijakan yang sama.

Tabel berikut mencantumkan kemungkinan nilai `aws:SourceArn` untuk operasi API QLDB `ExportJournalToS3`dan `StreamsJournalToKinesis`QLDB. Operasi ini berada dalam lingkup untuk masalah keamanan ini karena mereka memanggil AWS Security Token Service (AWS STS) untuk mengambil peran IAM yang Anda tentukan.

| Operasi API | Layanan yang dipanggil | aws: SourceArn |
|---|---|---|
| `ExportJournalToS3` | AWS STS (`AssumeRole`) | Memungkinkan QLDB untuk mengambil peran untuk sumber daya QLDB apa pun di akun:<br>`arn:aws:qldb: us-east-1 :123456789012 :*`<br>Saat ini, QLDB hanya mendukung ARN wildcard ini untuk ekspor jurnal. |
| `StreamsJournalToKinesis` | AWS STS (`AssumeRole`) | Memungkinkan QLDB untuk mengambil peran untuk aliran QLDB tertentu:<br>`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /IiPT4brpZCqCq3f4MTHbYy` |

| Operasi API | Layanan yang dipanggil | aws: SourceArn |
| --- | --- | --- |
| | | Catatan: Anda hanya dapat menentukan ID aliran di ARN setelah sumber daya aliran dibuat. Dengan menggunakan ARN ini, Anda dapat mengizinkan peran hanya digunakan untuk satu aliran QLDB.<br><br>Memungkinkan QLDB untuk mengambil peran untuk setiap aliran QLDB dari buku besar:<br><br>`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /*`<br><br>Memungkinkan QLDB untuk mengambil peran untuk setiap aliran QLDB di akun:<br><br>`arn:aws:qldb: us-east-1 :123456789012 :stream/*`<br><br>Memungkinkan QLDB untuk mengambil peran untuk sumber daya QLDB apa pun di akun:<br><br>`arn:aws:qldb: us-east-1 :123456789012 :*` |

Cara paling efektif untuk melindungi dari masalah confused deputy adalah dengan menggunakan kunci konteks kondisi global `aws:SourceArn` dengan ARN lengkap sumber daya. Jika Anda tidak mengetahui ARN lengkap sumber daya atau jika Anda menentukan beberapa sumber daya, gunakan kunci kondisi konteks `aws:SourceArn` global dengan karakter wildcard (*) untuk bagian ARN yang tidak diketahui—misalnya,. `arn:aws:qldb:us-east-1:123456789012:*`

Contoh kebijakan kepercayaan berikut untuk peran IAM menunjukkan bagaimana Anda dapat menggunakan kunci konteks kondisi `aws:SourceAccount` global `aws:SourceArn` dan untuk mencegah masalah wakil yang membingungkan. Dengan kebijakan kepercayaan ini, QLDB dapat mengambil peran untuk aliran QLDB apa pun di akun untuk buku besar saja. `123456789012` `myExampleLedger`

Untuk menggunakan kebijakan ini, ganti *us-east-1*, 123456789012, dan dalam contoh dengan informasi Anda sendiri. *myExampleLedger*

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "qldb.amazonaws.com"
    },
    "Action": [ "sts:AssumeRole" ],
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:qldb:us-east-1:123456789012:stream/myExampleLedger/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

## AWS kebijakan terkelola untuk Amazon QLDB

Kebijakan AWS terkelola adalah kebijakan mandiri yang dibuat dan dikelola oleh AWS. AWS Kebijakan terkelola dirancang untuk memberikan izin bagi banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwa kebijakan AWS terkelola mungkin tidak memberikan izin hak istimewa paling sedikit untuk kasus penggunaan spesifik Anda karena tersedia untuk digunakan semua pelanggan. AWS Kami menyarankan Anda untuk mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalam kebijakan AWS terkelola. Jika AWS memperbarui izin yang ditentukan dalam kebijakan AWS terkelola, pembaruan akan memengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan kebijakan tersebut. AWS kemungkinan besar akan memperbarui kebijakan AWS terkelola saat baru Layanan AWS diluncurkan atau operasi API baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat AWS kebijakan yang dikelola dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang operasi QLDB API dalam kebijakan terkelola AWS ini, lihat. Referensi API Amazon QLDB

Topik

- AWS kebijakan terkelola: AmazonQLDB ReadOnly
- AWS kebijakan terkelola: AmazonQLDB FullAccess
- AWS kebijakan terkelola: AmazonQLDB ConsoleFullAccess
- Pembaruan QLDB pada kebijakan terkelola AWS

## AWS kebijakan terkelola: AmazonQLDB ReadOnly

Gunakan ReadOnly kebijakan AmazonQLDB untuk memberikan izin hanya-baca ke semua sumber daya QLDB. Anda dapat melampirkan kebijakan ini ke identitas IAM Anda.

Detail izin

Kebijakan ini mencakup izin berikut untuk `qldb` layanan.

- Memungkinkan prinsipal untuk mendeskripsikan dan mencantumkan semua sumber daya QLDB dan tag mereka. Sumber daya ini mencakup buku besar, pekerjaan ekspor Amazon S3, dan aliran ke Kinesis Data Streams.
- Memungkinkan kepala sekolah untuk mendapatkan blok, intisari, atau revisi dari jurnal di buku besar apa pun untuk memverifikasi data secara kriptografis.
- Tidak mengizinkan prinsipal untuk menjalankan perintah PartiQL apa pun pada tabel apa pun di buku besar apa pun.

## AWS kebijakan terkelola: AmazonQLDB FullAccess

Gunakan FullAccess kebijakan AmazonQLDB untuk memberikan izin administratif penuh ke semua sumber daya QLDB melalui QLDB API atau file. AWS CLI Anda dapat melampirkan kebijakan ini ke identitas IAM Anda.

Detail izin

Kebijakan ini mencakup izin berikut.

- `qldb`
  - Memungkinkan prinsipal untuk membuat, mendeskripsikan, membuat daftar, dan mengelola semua sumber daya QLDB dan tag mereka. Sumber daya ini mencakup buku besar, pekerjaan ekspor Amazon S3, dan aliran ke Kinesis Data Streams.
  - Memungkinkan prinsipal untuk menjalankan semua perintah PartiQL pada semua tabel di buku besar apa pun dengan menggunakan driver QLDB atau shell QLDB.
  - Memungkinkan kepala sekolah untuk mendapatkan blok, intisari, atau revisi dari jurnal di buku besar apa pun untuk memverifikasi data secara kriptografis.
- `iam`— Memungkinkan kepala sekolah untuk meneruskan sumber daya peran IAM apa pun di akun Anda ke layanan QLDB. Ini diperlukan untuk semua permintaan ekspor dan aliran jurnal.

## AWS kebijakan terkelola: AmazonQLDB ConsoleFullAccess

Gunakan ConsoleFullAccess kebijakan AmazonQLDB untuk memberikan izin administratif penuh ke semua sumber daya QLDB melalui AWS Management Console, QLDB API, atau file. AWS CLI Anda dapat melampirkan kebijakan ini ke identitas IAM Anda.

Detail izin

Kebijakan ini mencakup izin berikut.

- `qldb`
  - Memungkinkan prinsipal untuk membuat, mendeskripsikan, membuat daftar, dan mengelola semua sumber daya QLDB dan tag mereka. Sumber daya ini mencakup buku besar, pekerjaan ekspor Amazon S3, dan aliran ke Kinesis Data Streams.
  - Memungkinkan prinsipal menjalankan semua perintah PartiQL pada semua tabel di buku besar apa pun dengan menggunakan konsol QLDB, driver QLDB, atau shell QLDB.
  - Memungkinkan prinsipal untuk memasukkan data aplikasi sampel dalam buku besar apa pun dengan menggunakan konsol QLDB.
  - Memungkinkan kepala sekolah untuk mendapatkan blok, intisari, atau revisi dari jurnal di buku besar apa pun untuk memverifikasi data secara kriptografis.
- `dbqms`— Memungkinkan prinsipal untuk menggunakan semua tindakan dalam Layanan Metadata Kueri Database. Ini adalah layanan internal saja yang diperlukan konsol QLDB untuk membuat, mendeskripsikan, dan mengelola kueri terbaru dan tersimpan untuk editor kueri PartiQL.

- `kinesis`— Memungkinkan kepala sekolah untuk mendeskripsikan dan mencantumkan sumber daya Amazon Kinesis Data Streams. Sumber daya ini adalah tujuan target yang sumber daya aliran QLDB dapat menulis data.
- `iam`— Memungkinkan kepala sekolah untuk meneruskan sumber daya peran IAM apa pun di akun Anda ke layanan QLDB. Ini diperlukan untuk semua permintaan ekspor dan aliran jurnal.

## Pembaruan QLDB pada kebijakan terkelola AWS

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk QLDB sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman riwayat Rilis QLDB.

| Perubahan | Deskripsi | Tanggal |
| --- | --- | --- |
| AmazonQLDB, AmazonQLDB - FullAccess Perbarui ke kebijakan ConsoleFullAccess yang ada | QLDB menambahkan izin baru untuk mengizinkan prinsipal menyunting revisi dokumen di semua buku besar dalam mode izin. `STANDARD` | 4 November 2022 |
| AmazonQLDB, AmazonQLDB - FullAccess Perbarui ke kebijakan ConsoleFullAccess yang ada | QLDB menambahkan izin baru untuk mengizinkan prinsipal meneruskan sumber daya peran IAM apa pun di akun Anda ke layanan QLDB. Ini diperlukan untuk semua permintaan ekspor dan aliran jurnal. | 2 September 2021 |
| AmazonQLDB ReadOnly - Perbarui ke kebijakan yang ada | QLDB menghapus tindakan `qldb:GetBlock` duplikat yang sebelumnya terdaftar dua kali, dan menyusun ulang bidang sehingga `"Effect"` | 1 Juli 2021 |

| Perubahan | Deskripsi | Tanggal |
| --- | --- | --- |
| | muncul sebelum bidang. `"Action"` | |
| AmazonQLDB, AmazonQLDB - FullAccess Perbarui ke kebijakan ConsoleFullAccess yang ada | QLDB menambahkan izin baru untuk memungkinkan prinsipal memperbarui mode izin di semua buku besar, dan menjalankan semua perintah PartiQL di semua buku besar dalam mode izin baru. `STANDARD`<br><br>Mode `STANDARD` izin mendukung kontrol akses tingkat tabel dan perincian untuk perintah PartiQL. Untuk memfasilitasi mode izin baru, QLDB memperkenalkan serangkaian tindakan IAM untuk tipe perintah PartiQL, dan Nama Sumber Daya Amazon (ARN) untuk sumber daya tabel QLDB. Kedua kebijakan ini diperbarui untuk menyertakan tindakan PartiQL baru untuk memberikan akses penuh ke buku besar. `STANDARD` | 27 Mei 2021 |
| QLDB mulai melacak perubahan | QLDB mulai melacak perubahan untuk AWS kebijakan yang dikelola. | 1 Maret 2021 |

# Memecahkan masalah identitas dan akses QLDB Amazon

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan QLDB dan IAM.

Topik



## Saya tidak berwenang untuk melakukan tindakan di QLDB

Jika AWS Management Console memberitahu Anda bahwa Anda tidak berwenang untuk melakukan tindakan, maka Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Contoh kesalahan berikut terjadi ketika pengguna `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`myExampleLedger`* fiktif, tetapi tidak memiliki izin `qldb:`*`DescribeLedger`* fiktif.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 qldb:DescribeLedger on resource: myExampleLedger
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya untuk mengizinkan dia mengakses sumber daya *`myExampleLedger`* menggunakan tindakan `qldb:`*`DescribeLedger`*.

## Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan bahwa Anda tidak berwenang untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke QLDB.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di QLDB. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Untuk panduan pemecahan masalah tentang kesalahan ini khusus untuk operasi ekspor atau aliran jurnal, lihat. Pemecahan masalah Amazon QLDB

## Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya QLDB saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mengetahui apakah QLDB mendukung fitur-fitur ini, lihat. Bagaimana Amazon QLDB bekerja dengan IAM
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara penggunaan kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.

# Pencatatan dan pemantauan di Amazon QLDB

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja Amazon QLDB dan solusi Anda. AWS Anda harus mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. Namun, sebelum Anda mulai memantau QLDB, Anda harus membuat rencana pemantauan yang mencakup jawaban atas pertanyaan-pertanyaan berikut:

- Apa tujuan pemantauan Anda?
- Sumber daya apa yang akan Anda pantau?
- Seberapa sering Anda akan memantau sumber daya ini?
- Alat pemantauan apa yang akan Anda gunakan?
- Siapa yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

Langkah selanjutnya adalah menetapkan garis dasar untuk kinerja QLDB normal di lingkungan Anda, dengan mengukur kinerja pada berbagai waktu dan dalam kondisi beban yang berbeda. Saat Anda memantau QLDB, simpan data pemantauan historis sehingga Anda dapat membandingkannya dengan data kinerja saat ini, mengidentifikasi pola kinerja normal dan anomali kinerja, dan merancang metode untuk mengatasi masalah.

Untuk menetapkan batas dasar, setidaknya Anda harus memantau item berikut:

- Baca dan tulis I/Os dan penyimpanan, sehingga Anda dapat melacak pola konsumsi buku besar Anda untuk tujuan penagihan.
- Latensi perintah, sehingga Anda dapat melacak kinerja buku besar Anda saat menjalankan operasi data.
- Pengecualian, sehingga Anda dapat menentukan apakah ada permintaan yang mengakibatkan kesalahan.

Topik

- Mencatat panggilan API QLDB Amazon dengan AWS CloudTrail

# Alat pemantauan

AWS menyediakan berbagai alat yang dapat Anda gunakan untuk memantau Amazon QLDB. Anda dapat mengonfigurasi beberapa alat ini untuk melakukan pemantauan untuk Anda, sementara beberapa alat memerlukan intervensi manual. Kami menyarankan agar Anda mengautomasi tugas pemantauan sebanyak mungkin.

Topik

- Alat pemantauan otomatis
- Alat pemantauan manual

## Alat pemantauan otomatis

Anda dapat menggunakan alat pemantauan otomatis berikut untuk menonton QLDB dan melaporkan ketika ada sesuatu yang salah:

- CloudWatch Alarm Amazon — Tonton satu metrik selama periode waktu yang Anda tentukan, dan lakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas tertentu selama beberapa periode waktu. Tindakannya adalah pemberitahuan yang dikirim ke topik Amazon Simple Notification Service (Amazon SNS) atau kebijakan Amazon EC2 Auto Scaling. CloudWatch alarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu; negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu. Untuk informasi selengkapnya, lihat Pemantauan CloudWatch dengan Amazon.
- Amazon CloudWatch Logs — Pantau, simpan, dan akses file log Anda dari AWS CloudTrail atau sumber lain. Untuk informasi selengkapnya, lihat Memantau file log di Panduan CloudWatch Pengguna Amazon.
- CloudWatch Acara Amazon — Cocokkan acara dan arahkan ke satu atau beberapa fungsi atau aliran target untuk membuat perubahan, menangkap informasi status, dan mengambil tindakan korektif. Untuk informasi selengkapnya, lihat Apa itu CloudWatch Acara Amazon di Panduan CloudWatch Pengguna Amazon.
- AWS CloudTrail Pemantauan Log - Bagikan file log antar akun, pantau file CloudTrail log secara real time dengan mengirimkannya ke CloudWatch Log, menulis aplikasi pemrosesan log di Java, dan validasi bahwa file log Anda tidak berubah setelah pengiriman oleh CloudTrail. Untuk informasi selengkapnya, lihat Bekerja dengan file CloudTrail log di Panduan AWS CloudTrail Pengguna.

### Alat pemantauan manual

Bagian penting lainnya dari pemantauan QLDB melibatkan pemantauan secara manual item-item yang tidak tercakup oleh CloudWatch alarm. QLDB CloudWatch,, Trusted Advisor, dan dasbor AWS Management Console lainnya memberikan pandangan tentang at-a-glance keadaan lingkungan Anda. AWS Kami menyarankan Anda juga memeriksa file log di Amazon QLDB.

- Dasbor QLDB menunjukkan hal berikut:
  - Baca dan tulis I/Os
  - Jurnal dan penyimpanan yang diindeks
  - Latensi perintah
  - Pengecualian
- CloudWatch Halaman beranda menunjukkan yang berikut:
  - Alarm dan status saat ini
  - Grafik alarm dan sumber daya
  - Status kesehatan layanan

  Selain itu, Anda dapat menggunakan CloudWatch untuk melakukan hal berikut:
  - Membuat dasbor yang disesuaikan untuk memantau layanan yang penting bagi Anda
  - Data metrik grafik untuk memecahkan masalah dan mengungkap tren
  - Cari dan telusuri semua metrik AWS sumber daya Anda
  - Membuat dan mengedit alarm untuk menerima notifikasi terkait masalah

## Pemantauan CloudWatch dengan Amazon

Anda dapat memantau Amazon QLDB CloudWatch menggunakan, yang mengumpulkan dan memproses data mentah dari Amazon QLDB menjadi metrik yang dapat dibaca. near-real-time Ini mencatat statistik ini selama dua minggu sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif yang lebih baik tentang kinerja aplikasi atau layanan web Anda. Secara default, data metrik QLDB secara otomatis dikirim CloudWatch ke dalam periode 1 atau 15 menit. Untuk informasi selengkapnya, lihat Apa itu Amazon CloudWatch, CloudWatch Acara Amazon, dan CloudWatch Log Amazon? di Panduan CloudWatch Pengguna Amazon.

Topik

- Bagaimana cara menggunakan metrik QLDB?

- Metrik dan dimensi Amazon QLDB
- Membuat CloudWatch alarm untuk memantau Amazon QLDB

## Bagaimana cara menggunakan metrik QLDB?

Metrik yang dilaporkan oleh QLDB memberikan informasi yang dapat Anda analisis dengan berbagai cara. Daftar berikut menunjukkan beberapa penggunaan umum untuk metrik. Daftar ini berisi saran agar Anda dapat memulai, bukan daftar komprehensif.

- Anda dapat memantau `JournalStorage` dan `IndexedStorage` selama periode waktu tertentu, untuk melacak berapa banyak ruang disk yang dikonsumsi buku besar Anda.
- Anda dapat memantau `ReadIOs` dan `WriteIOs` selama periode waktu tertentu, untuk melacak berapa banyak permintaan yang diproses oleh buku besar Anda.
- Anda dapat memantau `CommandLatency` untuk melacak kinerja buku besar Anda untuk operasi data dan menganalisis jenis perintah yang menghasilkan latensi paling banyak.

## Metrik dan dimensi Amazon QLDB

Saat Anda berinteraksi dengan Amazon QLDB, itu mengirimkan metrik dan dimensi berikut ke. CloudWatch Metrik penyimpanan dilaporkan setiap 15 menit, dan semua metrik lainnya dikumpulkan dan dilaporkan setiap menit. Anda dapat menggunakan prosedur berikut untuk melihat metrik QLDB.

Untuk melihat metrik menggunakan konsol CloudWatch

Metrik dikelompokkan terlebih dahulu berdasarkan namespace layanan, lalu berdasarkan berbagai kombinasi dimensi dalam setiap namespace.

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Jika perlu, ubah Wilayah. Pada bilah navigasi, pilih Wilayah tempat AWS sumber daya Anda berada. Untuk informasi selengkapnya, lihat Wilayah dan titik akhir.
3. Di panel navigasi, pilih Metrik.
4. Di bawah tab Semua metrik, pilih QLDB.

Untuk melihat metrik menggunakan AWS CLI

- Pada jendela perintah, gunakan perintah berikut.

```
aws cloudwatch list-metrics --namespace "AWS/QLDB"
```

CloudWatch menampilkan metrik berikut untuk QLDB.

Dimensi dan metrik Amazon QLDB

Metrik dan dimensi yang dikirimkan Amazon QLDB ke CloudWatch Amazon tercantum di sini.

Metrik QLDB

| Metrik | Deskripsi |
| --- | --- |
| `JournalStorage` | Jumlah total ruang disk yang digunakan oleh jurnal buku besar, dilaporkan dalam interval 15 menit. Jurnal ini berisi riwayat lengkap, tidak dapat diubah, dan dapat diverifikasi dari semua perubahan pada data Anda.<br><br>Unit: `Bytes`<br><br>Dimensi: `LedgerName` |
| `IndexedStorage` | Jumlah total ruang disk yang digunakan oleh tabel buku besar, indeks, dan riwayat yang diindeks, dilaporkan dalam interval 15 menit. Penyimpanan terindeks terdiri dari data buku besar yang dioptimalkan untuk kueri berkinerja tinggi.<br><br>Unit: `Bytes`<br><br>Dimensi: `LedgerName` |
| `ReadIOs` | Jumlah permintaan I/O baca, dilaporkan dalam interval satu menit. Ini menangkap semua jenis operasi baca, termasuk transaksi data, permintaan verifikasi, ekspor jurnal, dan aliran jurnal.<br><br>Unit: `Count`<br><br>Dimensi: `LedgerName` |

| Metrik | Deskripsi |
| --- | --- |
| `WriteIOs` | Jumlah permintaan I/O tulis, dilaporkan dalam interval satu menit.<br><br>Unit: `Count`<br><br>Dimensi: `LedgerName` |
| `CommandLatency` | Jumlah waktu yang dibutuhkan untuk operasi data, dilaporkan dalam interval satu menit.<br><br>Unit: `Milliseconds`<br><br>Dimensi: `CommandType, LedgerName` |
| `IsImpaired` | Bendera yang menunjukkan jika aliran jurnal ke Kinesis Data Streams terganggu, dilaporkan dalam interval satu menit. Nilai `1` menunjukkan bahwa aliran dalam keadaan terganggu, dan `0` menunjukkan sebaliknya.<br><br>Unit: `Boolean` (0 atau 1)<br><br>Dimensi: `LedgerName, StreamId` |
| `OccConflictExceptions` | Jumlah permintaan ke QLDB yang menghasilkan file. `OccConflictException` Untuk informasi tentang kontrol konkurensi optimis (OCC), lihat. Model Konkurensi Amazon QLDB<br><br>Unit: `Count` |
| `Session4xxExceptions` | Jumlah permintaan ke QLDB yang menghasilkan kesalahan HTTP 4xx.<br><br>Unit: `Count` |
| `Session5xxExceptions` | Jumlah permintaan ke QLDB yang menghasilkan kesalahan HTTP 5xx.<br><br>Unit: `Count` |

| Metrik | Deskripsi |
| --- | --- |
| SessionRateExceededExceptions | Jumlah permintaan ke QLDB yang menghasilkan a. SessionRateExceededException<br><br>Unit: Count |

Dimensi untuk metrik QLDB

Metrik untuk QLDB dikualifikasikan berdasarkan nilai untuk akun, nama buku besar, ID aliran, atau jenis perintah. Anda dapat menggunakan CloudWatch konsol untuk mengambil data QLDB sepanjang salah satu dimensi dalam tabel berikut.

| Dimensi | Deskripsi |
| --- | --- |
| LedgerName | Dimensi ini membatasi data ke buku besar tertentu. Nilai ini dapat berupa nama buku besar apa pun di saat ini Wilayah AWS dan saat ini Akun AWS. |
| StreamId | Dimensi ini membatasi data ke aliran jurnal tertentu. Nilai ini dapat berupa ID aliran apa pun untuk buku besar saat ini Wilayah AWS dan saat ini Akun AWS. |
| CommandType | Dimensi ini membatasi data ke salah satu perintah API data QLDB berikut:<br><br>• AbortTransaction<br>• CommitTransaction<br>• EndSession<br>• ExecuteStatement<br>• FetchPage<br>• StartSession<br>• StartTransaction |

| Dimensi | Deskripsi |
| --- | --- |
| | Untuk mempelajari cara QLDB menggunakan perintah ini untuk mengelola operasi data, lihat. Manajemen sesi dengan pengemudi |

### Membuat CloudWatch alarm untuk memantau Amazon QLDB

Anda dapat membuat CloudWatch alarm Amazon yang mengirimkan pesan Amazon Simple Notification Service (Amazon SNS) saat alarm berubah status. Alarm mengawasi metrik tunggal selama periode waktu yang Anda tentukan. Alarm tersebut melakukan satu atau beberapa tindakan berdasarkan nilai metrik yang relatif terhadap ambang batas tertentu selama beberapa periode waktu. Tindakan ini adalah notifikasi yang dikirim ke topik Amazon SNS atau kebijakan Penskalaan Otomatis.

Alarm memanggil tindakan untuk perubahan status berkelanjutan saja. CloudWatch alarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu. Status harus diubah dan dipelihara selama jangka waktu tertentu.

Untuk informasi selengkapnya tentang membuat CloudWatch alarm, lihat Menggunakan CloudWatch alarm Amazon di  CloudWatch Panduan Pengguna Amazon.

## Mengotomatiskan Amazon QLDB dengan Acara CloudWatch

Amazon CloudWatch Events memungkinkan Anda mengotomatiskan Layanan AWS dan merespons secara otomatis peristiwa sistem seperti masalah ketersediaan aplikasi atau perubahan sumber daya. Acara dari Layanan AWS dikirim ke CloudWatch Acara dalam waktu nyaris nyata. Anda dapat menulis aturan sederhana untuk menunjukkan kejadian mana yang sesuai kepentingan Anda, dan tindakan otomatis apa yang diambil ketika suatu kejadian sesuai dengan suatu aturan. Tindakan yang dapat dipicu secara otomatis meliputi hal-hal berikut:

- Memanggil fungsi AWS Lambda
- Meminta Perintah Amazon EC2 Run
- Mengirim peristiwa ke Amazon Kinesis Data Streams
- Mengaktifkan mesin AWS Step Functions negara
- Memberi tahu topik Amazon SNS atau antrean Amazon SQS

Amazon QLDB melaporkan peristiwa CloudWatch ke Acara setiap kali status sumber daya buku besar dalam perubahan Anda. Akun AWS Acara saat ini dipancarkan at-least-once berdasarkan jaminan, hanya untuk sumber daya buku besar QLDB.

Berikut ini adalah contoh peristiwa yang dilaporkan QLDB, di mana status buku besar berubah menjadi. `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"
    }
}
```

Beberapa contoh penggunaan CloudWatch Acara dengan QLDB dapat mencakup tetapi tidak terbatas pada hal-hal berikut:

- Mengaktifkan fungsi Lambda setiap kali buku besar baru awalnya dibuat `CREATING` dalam keadaan dan akhirnya menjadi. `ACTIVE`
- Memberi tahu topik Amazon SNS saat status buku besar Anda berubah menjadi dan kemudian `DELETING` menjadi. `DELETED`

Untuk informasi selengkapnya, lihat Panduan Pengguna CloudWatch Acara Amazon.

## Mencatat panggilan API QLDB Amazon dengan AWS CloudTrail

Amazon QLDB terintegrasi AWS CloudTrail dengan, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau dalam QLDB. Layanan AWS CloudTrail menangkap semua panggilan API manajemen sumber daya untuk QLDB sebagai peristiwa. Panggilan yang ditangkap termasuk panggilan dari konsol QLDB dan panggilan kode ke operasi QLDB API. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara terus menerus ke bucket Amazon Simple Storage Service (Amazon S3), termasuk peristiwa untuk QLDB. Jika Anda tidak

mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat untuk QLDB, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, termasuk cara mengonfigurasi dan mengaktifkannya, lihat Panduan AWS CloudTrail Pengguna.

## Informasi QLDB di CloudTrail

CloudTrail diaktifkan pada Akun AWS saat Anda membuat akun. Ketika aktivitas yang didukung terjadi di QLDB, aktivitas tersebut dicatat dalam CloudTrail suatu peristiwa bersama dengan peristiwa Layanan AWS lain dalam riwayat Peristiwa. Anda dapat melihat, mencari, dan mengunduh acara terbaru di situs Anda Akun AWS. Untuk informasi selengkapnya, lihat Melihat peristiwa dengan riwayat CloudTrail acara.

Untuk catatan acara yang sedang berlangsung di Anda Akun AWS, termasuk acara untuk QLDB, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk semua Wilayah AWS. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi lainnya Layanan AWS untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log.

Untuk informasi selengkapnya, lihat topik berikut di Panduan Pengguna AWS CloudTrail :

- Ikhtisar untuk membuat jejak
- CloudTrail layanan dan integrasi yang didukung
- Mengonfigurasi notifikasi Amazon SNS untuk CloudTrail
- Menerima file CloudTrail log dari beberapa Wilayah
- Menerima file CloudTrail log dari beberapa akun

Semua manajemen sumber daya QLDB dan tindakan API data non-transaksional dicatat oleh CloudTrail dan didokumentasikan dalam referensi API QLDB Amazon. Misalnya, panggilan ke`CreateLedger`,`DescribeLedger`, dan `DeleteLedger` tindakan menghasilkan entri dalam file CloudTrail log.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut ini:

- Apakah permintaan dibuat dengan root atau kredensyal pengguna
- Baik permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan
- Apakah permintaan itu dibuat oleh orang lain Layanan AWS

Untuk informasi selengkapnya, lihat elemen CloudTrail UserIdentity.

## Memahami entri file log QLDB

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, jadi file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan tindakan ini:

- `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"
    }
  ]
}
```

# Validasi kepatuhan untuk Amazon QLDB

Auditor pihak ketiga menilai keamanan dan kepatuhan Amazon QLDB sebagai bagian dari AWS beberapa program kepatuhan, termasuk namun tidak terbatas pada hal-hal berikut:

- Kontrol Sistem dan Organisasi (System and Organization Controls/SOC)
- Industri Kartu Pembayaran (Payment Card Industry/PCI)
- Organisasi Internasional untuk Standardisasi (ISO)
- Program Pengelolaan dan Penilaian Keamanan Sistem Informasi (ISMAP)
- Undang-Undang Akuntabilitas dan Portabilitas Asuransi Kesehatan (HIPAA)

> **Note**
>
> Ini bukan daftar lengkap sertifikasi QLDB Amazon.

Untuk mengetahui apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Memulai Cepat Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar AWS yang berfokus pada keamanan dan kepatuhan.
- Arsitektur untuk Keamanan dan Kepatuhan HIPAA di Amazon Web Services — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat aplikasi yang memenuhi syarat HIPAA.

> **Note**
>
> Tidak semua memenuhi Layanan AWS syarat HIPAA. Untuk informasi selengkapnya, lihat Referensi Layanan yang Memenuhi Syarat HIPAA.

- AWS Sumber Daya AWS — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- AWS Panduan Kepatuhan Pelanggan - Memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi (ISO)).
- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub— Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk sumber daya AWS Anda serta untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat Referensi kontrol Security Hub.
- Amazon GuardDuty — Ini Layanan AWS mendeteksi potensi ancaman terhadap beban kerja Akun AWS, kontainer, dan data Anda dengan memantau lingkungan Anda untuk aktivitas mencurigakan dan berbahaya. GuardDuty dapat membantu Anda mengatasi berbagai persyaratan kepatuhan, seperti PCI DSS, dengan memenuhi persyaratan deteksi intrusi yang diamanatkan oleh kerangka kerja kepatuhan tertentu.
- AWS Audit ManagerIni Layanan AWS membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

## Ketahanan di Amazon QLDB

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones. Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara

otomatis melakukan failover di antara Zona Ketersediaan tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur biasa yang terdiri dari satu atau beberapa pusat data.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat Infrastruktur AWS Global.

## Daya tahan penyimpanan

Penyimpanan jurnal QLDB menampilkan replikasi sinkron ke beberapa Availability Zone pada komitmen transaksi. Ini memastikan bahwa bahkan kegagalan Zona Ketersediaan penuh dari penyimpanan jurnal tidak akan mengganggu integritas data atau kemampuan untuk mempertahankan layanan aktif. Selain itu, jurnal QLDB menampilkan arsip asinkron ke penyimpanan yang toleran terhadap kesalahan. Fitur ini mendukung pemulihan bencana jika terjadi kegagalan penyimpanan simultan yang sangat tidak mungkin terjadi untuk beberapa Availability Zone.

Penyimpanan terindeks QLDB didukung oleh replikasi ke beberapa Availability Zone. Ini memastikan bahwa bahkan kegagalan Zona Ketersediaan penuh dari penyimpanan yang diindeks tidak akan mengganggu integritas data atau kemampuan untuk mempertahankan layanan aktif.

## Fitur daya tahan data

Selain infrastruktur AWS global, QLDB menawarkan fitur-fitur berikut untuk membantu mendukung ketahanan data dan kebutuhan cadangan Anda.

### Fitur layanan QLDB

Ekspor jurnal sesuai permintaan

QLDB menyediakan fitur ekspor jurnal sesuai permintaan. Akses konten jurnal Anda dengan mengekspor blok jurnal dari buku besar Anda ke bucket Amazon S3. Anda dapat menggunakan data ini untuk berbagai tujuan seperti retensi data, analitik, dan audit. Untuk informasi selengkapnya, lihat Mengekspor data jurnal dari Amazon QLDB.

Pencadangan dan pemulihan

Pemulihan otomatis untuk ekspor tidak didukung saat ini. Ekspor menyediakan kemampuan dasar untuk membuat redundansi data tambahan pada frekuensi yang Anda tentukan. Namun, skenario pemulihan bergantung pada aplikasi, di mana catatan yang diekspor mungkin ditulis kembali ke buku besar baru dengan memanfaatkan metode konsumsi yang sama atau serupa.

QLDB tidak menyediakan cadangan khusus dan fitur pemulihan terkait saat ini.

Aliran jurnal

QLDB juga menyediakan kemampuan aliran jurnal berkelanjutan. Anda dapat mengintegrasikan aliran jurnal QLDB dengan platform streaming Amazon Kinesis untuk memproses data jurnal waktu nyata. Untuk informasi selengkapnya, lihat Streaming data jurnal dari Amazon QLDB.

### Fitur desain QLDB

QLDB dirancang untuk menjadi tangguh terhadap korupsi logis. Jurnal QLDB tidak dapat diubah, memastikan bahwa semua transaksi yang dilakukan tetap ada di jurnal. Selain itu, setiap perubahan dokumen yang berkomitmen dicatat, karena ini memungkinkan point-in-timevisibilitas untuk setiap perubahan yang tidak diinginkan pada data buku besar.

QLDB tidak menyediakan fitur pemulihan otomatis untuk skenario korupsi logis saat ini.

## Keamanan infrastruktur di Amazon QLDB

Sebagai layanan terkelola, Amazon QLDB dilindungi oleh prosedur keamanan jaringan global AWS yang dijelaskan dalam whitepaper Amazon Web Services: Ikhtisar Proses Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses QLDB melalui jaringan. Klien harus mendukung Keamanan Lapisan Pengangkutan (TLS) 1.0 atau versi yang lebih baru. Kami merekomendasikan TLS 1.2 atau versi yang lebih baru. Klien juga harus mendukung suite cipher dengan perfect forward secrecy (PFS) seperti Ephemeral Diffie-Hellman (DHE) atau Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Sebagian besar sistem-sistem modern seperti Java 7 dan versi yang lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani dengan menggunakan kredensi terprogram yang terkait dengan prinsipal IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Anda juga dapat menggunakan titik akhir virtual private cloud (VPC) untuk QLDB. Titik akhir VPC antarmuka memungkinkan sumber daya VPC Amazon Anda menggunakan alamat IP pribadinya untuk mengakses QLDB tanpa eksposur ke internet publik. Untuk informasi selengkapnya, lihat Akses Amazon QLDB menggunakan titik akhir antarmuka ()AWS PrivateLink.

# Akses Amazon QLDB menggunakan titik akhir antarmuka ()AWS PrivateLink

Anda dapat menggunakan AWS PrivateLink untuk membuat koneksi pribadi antara VPC Anda dan Amazon QLDB. Anda dapat mengakses QLDB seolah-olah berada di VPC Anda, tanpa menggunakan gateway internet, perangkat NAT, koneksi VPN, atau koneksi. AWS Direct Connect Instans di VPC Anda tidak memerlukan alamat IP publik untuk mengakses QLDB.

Anda membuat koneksi pribadi ini dengan membuat titik akhir antarmuka, yang didukung oleh AWS PrivateLink. Kami membuat antarmuka jaringan endpoint di setiap subnet yang Anda aktifkan untuk titik akhir antarmuka. Ini adalah antarmuka jaringan yang dikelola pemohon yang berfungsi sebagai titik masuk untuk lalu lintas yang ditujukan untuk QLDB.

Untuk informasi selengkapnya, lihat Mengakses Layanan AWS melalui AWS PrivateLink di Panduan AWS PrivateLink .

Topik



## Pertimbangan untuk QLDB

Sebelum Anda menyiapkan titik akhir antarmuka untuk QLDB, tinjau Pertimbangan dalam Panduan.AWS PrivateLink

> **Note**
>
> QLDB hanya mendukung panggilan ke API data transaksional Sesi QLDB melalui titik akhir antarmuka. API ini hanya mencakup SendCommandoperasi. Dalam mode `STANDARD` izin buku besar, Anda dapat mengontrol izin untuk tindakan PartiQL tertentu di API ini.

## Buat titik akhir antarmuka untuk QLDB

Anda dapat membuat titik akhir antarmuka untuk QLDB menggunakan konsol VPC Amazon atau (). AWS Command Line Interface AWS CLI Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka di AWS PrivateLink Panduan.

Buat endpoint antarmuka untuk QLDB menggunakan nama layanan berikut:

```
com.amazonaws.region.qldb.session
```

Jika Anda mengaktifkan DNS pribadi untuk titik akhir antarmuka, Anda dapat membuat permintaan API ke QLDB menggunakan nama DNS Regional default. Misalnya, `session.qldb.us-east-1.amazonaws.com`.

## Buat kebijakan titik akhir untuk titik akhir antarmuka Anda

Kebijakan endpoint adalah sumber daya IAM yang dapat Anda lampirkan ke titik akhir antarmuka. Kebijakan endpoint default memungkinkan akses penuh ke QLDB melalui titik akhir antarmuka. Untuk mengontrol akses yang diizinkan ke QLDB dari VPC Anda, lampirkan kebijakan titik akhir kustom ke titik akhir antarmuka.

kebijakan titik akhir mencantumkan informasi berikut:

- Prinsipal yang dapat melakukan tindakan (Akun AWS, pengguna, dan peran).
- Tindakan yang dapat dilakukan.
- Sumber daya untuk melakukan tindakan.

Untuk informasi selengkapnya, lihat Mengontrol akses ke layanan menggunakan kebijakan titik akhir di Panduan AWS PrivateLink .

Anda juga dapat menggunakan `Condition` bidang dalam kebijakan yang dilampirkan ke pengguna, grup, atau peran untuk mengizinkan akses hanya dari titik akhir antarmuka tertentu. Ketika digunakan bersama-sama, kebijakan endpoint dan kebijakan IAM dapat membatasi akses ke tindakan QLDB tertentu pada buku besar tertentu ke titik akhir antarmuka tertentu.

Contoh kebijakan titik akhir: Batasi akses ke buku besar QLDB tertentu

Berikut ini adalah contoh kebijakan endpoint kustom untuk QLDB. Saat Anda melampirkan kebijakan ini ke titik akhir antarmuka Anda, kebijakan ini memberikan akses ke `SendCommand` tindakan dan

tindakan hanya-baca PartiQL untuk semua prinsipal pada sumber daya buku besar yang ditentukan. Dalam contoh ini, buku besar harus dalam mode STANDARD izin.

Untuk menggunakan kebijakan ini, ganti *us-east-1*, 123456789012, dan dalam contoh dengan informasi Anda sendiri. *myExampleLedger*

```
{
   "Statement": [
      {
         "Sid": "QLDBSendCommandPermission",
         "Principal": "*",
         "Effect": "Allow",
         "Action": "qldb:SendCommand",
         "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
      },
      {
         "Sid": "QLDBPartiQLReadOnlyPermissions",
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "qldb:PartiQLSelect",
            "qldb:PartiQLHistoryFunction"
         ],
         "Resource": [
            "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*",
            "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
information_schema/user_tables"
         ]
      }
   ]
}
```

Contoh kebijakan IAM: Batasi akses ke buku besar QLDB dari titik akhir antarmuka tertentu saja

Berikut ini adalah contoh kebijakan berbasis identitas IAM untuk QLDB. Jika Anda melampirkan kebijakan ini ke pengguna, peran, atau grup, kebijakan ini memungkinkan SendCommand akses ke sumber daya buku besar hanya dari titik akhir antarmuka yang ditentukan.

*Untuk menggunakan kebijakan ini, ganti us-east-1, 123456789012 myExampleLedger,, dan vpce-1a2b3c4d dalam contoh dengan informasi Anda sendiri.*

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

## Ketersediaan titik akhir antarmuka untuk QLDB

Amazon QLDB mendukung titik akhir antarmuka dengan kebijakan di semua Wilayah AWS tempat QLDB tersedia. Untuk daftar lengkap Wilayah yang tersedia, lihat titik akhir dan kuota Amazon QLDB di bagian. Referensi Umum AWS

# Pemecahan masalah Amazon QLDB

Bagian berikut ini menyediakan daftar agregat kesalahan umum yang mungkin Anda alami ketika menggunakan Amazon QLDB, dan panduan cara untuk memecahkan masalah tersebut.

Untuk panduan pemecahan masalah khusus untuk akses IAM, lihatMemecahkan masalah identitas dan akses QLDB Amazon.

Untuk praktik terbaik untuk menyetel pernyataan PartiQL Anda, lihatMengoptimalkan kinerja kueri.

Topik



## Menjalankan transaksi menggunakan driver QLDB

Bagian ini mencantumkan pengecualian umum yang dapat dikembalikan oleh driver Amazon QLDB saat Anda menggunakannya untuk menjalankan transaksi PartiQL pada buku besar. Untuk informasi selengkapnya tentang fitur ini, lihat Memulai dengan driver. Untuk praktik terbaik dalam mengonfigurasi dan menggunakan driver, lihatRekomendasi pengemudi.

Setiap pengecualian mencakup pesan kesalahan tertentu, diikuti dengan deskripsi singkat dan saran untuk solusi yang mungkin.

CapacityExceededException

> Pesan: Kapasitas terlampaui
>
> Amazon QLDB menolak permintaan karena melebihi kapasitas pemrosesan buku besar. QLDB memberlakukan batas skala internal per buku besar untuk menjaga kesehatan dan kinerja layanan. Batas ini bervariasi tergantung pada ukuran beban kerja setiap permintaan individu. Misalnya, permintaan dapat memiliki beban kerja yang meningkat jika melakukan transaksi data yang tidak efisien, seperti pemindaian tabel yang dihasilkan dari kueri non-indeks yang memenuhi syarat.

Kami menyarankan Anda menunggu sebelum mencoba ulang permintaan. Jika aplikasi Anda secara konsisten menemukan pengecualian ini, optimalkan pernyataan Anda dan kurangi tingkat dan volume permintaan yang Anda kirim ke buku besar. Contoh optimasi pernyataan termasuk menjalankan lebih sedikit pernyataan per transaksi dan menyetel indeks tabel Anda. Untuk mempelajari cara mengoptimalkan pernyataan dan menghindari pemindaian tabel, lihatMengoptimalkan kinerja kueri.

Kami juga merekomendasikan menggunakan versi terbaru dari driver QLDB. Driver memiliki kebijakan coba ulang default yang menggunakan Exponential Backoff dan Jitter untuk secara otomatis mencoba lagi pengecualian seperti ini. Konsep backoff eksponensial adalah menggunakan waktu tunggu yang semakin lama antara percobaan ulang untuk respons kesalahan berturut-turut.

InvalidSessionException

Pesan: *`Transaksi TransactionID`* telah kedaluwarsa

Transaksi melebihi masa maksimumnya. Transaksi dapat berjalan hingga 30 detik sebelum dilakukan. Setelah batas batas waktu ini, setiap pekerjaan yang dilakukan pada transaksi ditolak, dan QLDB membuang sesi. Batas ini melindungi klien dari sesi bocor dengan memulai transaksi dan tidak melakukan atau membatalkannya.

Jika ini adalah pengecualian umum dalam aplikasi Anda, kemungkinan transaksi hanya membutuhkan waktu terlalu lama untuk dijalankan. Jika runtime transaksi memakan waktu lebih dari 30 detik, optimalkan laporan Anda untuk mempercepat transaksi. Contoh optimasi pernyataan termasuk menjalankan lebih sedikit pernyataan per transaksi dan menyetel indeks tabel Anda. Untuk informasi selengkapnya, lihat Mengoptimalkan kinerja kueri.

InvalidSessionException

Pesan: Session *`sessionId`* telah kedaluwarsa

QLDB dibuang sesi karena melebihi total seumur hidup maksimum. QLDB membuang sesi setelah 13-17 menit, terlepas dari transaksi aktif. Sesi dapat hilang atau terganggu karena sejumlah alasan, seperti kegagalan perangkat keras, kegagalan jaringan, atau restart aplikasi. Jadi, QLDB memberlakukan seumur hidup maksimum pada sesi untuk memastikan bahwa perangkat lunak klien tangguh terhadap kegagalan sesi.

Jika Anda menemukan pengecualian ini, kami sarankan Anda memperoleh sesi baru dan mencoba kembali transaksi. Kami juga merekomendasikan penggunaan driver QLDB versi terbaru, yang mengelola kumpulan sesi dan kesehatannya atas nama aplikasi.

InvalidSessionException

Pesan: Tidak ada sesi seperti itu

Klien mencoba untuk bertransaksi dengan QLDB menggunakan sesi yang tidak ada. Dengan asumsi bahwa klien menggunakan sesi yang sebelumnya ada, sesi mungkin tidak lagi ada karena salah satu dari berikut ini:

- Jika sesi terlibat dalam kegagalan server internal (yaitu, kesalahan dengan kode respons HTTP 500), QLDB mungkin memilih untuk membuang sesi sepenuhnya, daripada memungkinkan pelanggan untuk bertransaksi dengan sesi keadaan tidak pasti. Kemudian, setiap percobaan ulang pada sesi itu gagal dengan kesalahan ini.
- Sesi kedaluwarsa akhirnya dilupakan oleh QLDB. Kemudian, setiap upaya untuk terus menggunakan sesi mengakibatkan kesalahan ini, bukan awal`InvalidSessionException`.

Jika Anda menemukan pengecualian ini, kami sarankan Anda memperoleh sesi baru dan mencoba kembali transaksi. Kami juga merekomendasikan penggunaan driver QLDB versi terbaru, yang mengelola kumpulan sesi dan kesehatannya atas nama aplikasi.

RateExceededException

Pesan: Tarif terlampaui

QLDB mencekik klien berdasarkan identitas penelepon. QLDB memberlakukan throttling pada basis per wilayah, per akun menggunakan algoritma throttling token bucket. QLDB melakukan ini untuk membantu kinerja layanan, dan untuk memastikan penggunaan yang adil bagi semua pelanggan QLDB. Misalnya, mencoba memperoleh sejumlah besar sesi bersamaan menggunakan`StartSessionRequest` operasi dapat menyebabkan pelambatan.

Untuk menjaga kesehatan aplikasi Anda dan mengurangi pelambatan lebih lanjut, Anda dapat mencoba lagi pengecualian ini menggunakan Exponential Backoff dan Jitter. Konsep backoff eksponensial adalah menggunakan waktu tunggu yang semakin lama antara percobaan ulang untuk respons kesalahan berturut-turut. Sebaiknya gunakan versi terbaru dari driver QLDB. Driver memiliki kebijakan coba ulang default yang menggunakan backoff eksponensial dan jitter untuk secara otomatis mencoba lagi pengecualian seperti ini.

Versi terbaru dari driver QLDB juga dapat membantu jika aplikasi Anda secara konsisten mendapatkan throttled oleh QLDB untuk`StartSessionRequest` panggilan. Pengemudi mempertahankan kumpulan sesi yang digunakan kembali di seluruh transaksi, yang dapat membantu mengurangi jumlah`StartSessionRequest` panggilan yang dibuat aplikasi Anda. Untuk meminta peningkatan batas pembatasan API, hubungi AWS SupportPusat.

LimitExceededException

Pesan: Melebihi batas sesi

Buku besar melebihi kuota (juga dikenal sebagai batas) pada jumlah sesi aktif. Kuota ini didefinisikan dalamKuota dan batasan di Amazon QLDB. Jumlah sesi aktif buku besar pada akhirnya konsisten, dan buku besar secara konsisten berjalan di dekat kuota mungkin secara berkala melihat pengecualian ini.

Untuk menjaga kesehatan aplikasi Anda, kami sarankan untuk mencoba kembali pengecualian ini. Untuk menghindari pengecualian ini, pastikan bahwa Anda belum mengkonfigurasi lebih dari 1.500 sesi bersamaan untuk digunakan untuk buku besar tunggal di semua klien. Misalnya, Anda dapat menggunakan maxConcurrentTransactionsmetode driver Amazon QLDB untuk Java untuk mengonfigurasi jumlah maksimum sesi yang tersedia dalam instance driver.

QldbClientException

Pesan: Hasil streaming hanya berlaku saat transaksi induk terbuka

Transaksi ditutup, dan tidak dapat digunakan untuk mengambil hasil dari QLDB. Transaksi ditutup saat transaksi dilakukan atau dibatalkan.

Pengecualian ini terjadi ketika klien bekerja secara langsung dengan`Transaction` objek, dan itu mencoba untuk mengambil hasil dari QLDB setelah melakukan atau membatalkan transaksi. Untuk mengurangi masalah ini, klien harus membaca data sebelum menutup transaksi.

# Mengekspor data jurnal

Bagian ini mencantumkan pengecualian umum yang dapat dikembalikan QLDB saat Anda mengekspor data jurnal dari buku besar ke bucket Amazon S3. Untuk informasi selengkapnya tentang fitur ini, lihat Mengekspor data jurnal dari Amazon QLDB.

Setiap pengecualian mencakup pesan kesalahan tertentu, diikuti dengan deskripsi singkat dan saran untuk solusi yang mungkin.

AccessDeniedException

Pesan: Pengguna: *`UserArn`* tidak diizinkan untuk melakukan: iam:PassRole pada sumber daya: *`roleARN`*

Anda tidak memiliki izin untuk meneruskan sebuah peran IAM ke layanan QLDB. QLDB memerlukan peran untuk semua permintaan ekspor jurnal, dan Anda harus memiliki izin untuk meneruskan peran ini ke QLDB. Peran ini memberikan izin tulis kepada QLDB di bucket Amazon S3 yang Anda tentukan.

Pastikan Anda menentukan kebijakan IAM yang memberikan izin untuk melakukan operasi`PassRole` API pada sumber daya peran IAM yang Anda tentukan untuk layanan QLDB (`qldb.amazonaws.com`). Untuk contoh kebijakan, lihatContoh kebijakan berbasis identitas untuk Amazon QLDB.

IllegalArgumentException

Pesan: QLDB mengalami kesalahan memvalidasi konfigurasi S3: *`errorCode errorMessage`*

Kemungkinan penyebab kesalahan ini adalah bucket Amazon S3 yang disediakan tidak ada di Amazon S3. Atau, QLDB tidak memiliki izin yang cukup untuk menulis objek ke dalam bucket Amazon S3 yang Anda tentukan.

Pastikan nama bucket S3 yang Anda berikan dalam permintaan pekerjaan ekspor sudah benar. Untuk informasi selengkapnya tentang penamaan bucket, lihat Pembatasan dan batasan Bucket dalam Panduan Pengguna Amazon Simple Storage Service.

Juga, verifikasi bahwa Anda menentukan kebijakan untuk bucket yang ditentukan yang memberikan`PutObject` dan`PutObjectAcl` izin ke layanan QLDB (`qldb.amazonaws.com`). Untuk mempelajari selengkapnya, lihat Izin ekspor.

IllegalArgumentException

Pesan: Respons tak terduga dari Amazon S3 saat memvalidasi konfigurasi S3. Tanggapan dari S3: *`errorCode errorMessage`*

Upaya untuk menulis data ekspor jurnal ke bucket S3 yang disediakan gagal dengan respons kesalahan Amazon S3 yang disediakan. Untuk informasi selengkapnya tentang kemungkinan penyebabnya, lihat Pemecahan Masalah Amazon S3 di Panduan Pengguna Amazon Simple Storage Service.

IllegalArgumentException

Pesan: Awalan bucket Amazon S3 tidak boleh melebihi 128 karakter

Awalan yang disediakan dalam permintaan ekspor jurnal berisi lebih dari 128 karakter.

IllegalArgumentException

Pesan: Tanggal mulai tidak boleh lebih besar dari tanggal akhir

Baik`InclusiveStartTime` dan`ExclusiveEndTime` harus berada dalam format tanggal dan waktu ISO 8601 dan dalam Waktu Universal Terkoordinasi (UTC).

IllegalArgumentException

Pesan: Tanggal akhir tidak bisa di masa future

Keduanya`InclusiveStartTime` dan`ExclusiveEndTime` harus dalam format`ISO 8601` tanggal dan waktu dan di UTC.

IllegalArgumentException

Pesan: Pengaturan enkripsi objek yang disediakan (S3EncryptionConfiguration) tidak kompatibel dengan kunciAWS Key Management Service (AWS KMS)

Anda menyediakan`KMSKeyArn` dengan`ObjectEncryptionType` salah satu`NO_ENCRYPTION` atau`SSE_S3`. Anda hanya dapat menyediakan pelanggan yang dikelolaAWS KMS key untuk jenis enkripsi objek`SSE_KMS`. Untuk mempelajari lebih lanjut tentang opsi enkripsi sisi server di Amazon S3, lihat Melindungi data menggunakan enkripsi sisi server dalam Panduan Pengembang Amazon S3.

LimitExceededException

Pesan: Melebihi batas 2 secara bersamaan menjalankan pekerjaan ekspor Jurnal

QLDB memberlakukan batas default dari dua pekerjaan ekspor jurnal bersamaan.

## Streaming data jurnal

Bagian ini mencantumkan pengecualian umum yang dapat dikembalikan QLDB saat Anda melakukan streaming data jurnal dari buku besar ke Amazon Kinesis Data Streams. Untuk informasi selengkapnya tentang fitur ini, lihat Streaming data jurnal dari Amazon QLDB.

Setiap pengecualian mencakup pesan kesalahan tertentu, diikuti dengan deskripsi singkat dan saran untuk solusi yang mungkin.

AccessDeniedException

Pesan: Pengguna: *`UserArn`* tidak diizinkan untuk melakukan: iam:PassRole pada sumber daya: *`roleARN`*

Anda tidak memiliki izin untuk meneruskan sebuah peran IAM ke layanan QLDB. QLDB memerlukan peran untuk semua permintaan aliran jurnal, dan Anda harus memiliki izin untuk meneruskan peran ini ke QLDB. Peran ini memberi QLDB izin tulis di sumber daya Amazon Kinesis Data Streams yang Anda tentukan.

Pastikan Anda menentukan kebijakan IAM yang memberikan izin untuk melakukan operasi`PassRole` API pada sumber daya peran IAM yang Anda tentukan untuk layanan QLDB (`qldb.amazonaws.com`). Untuk contoh kebijakan, lihatContoh kebijakan berbasis identitas untuk Amazon QLDB.

IllegalArgumentException

Pesan: QLDB mengalami kesalahan saat memvalidasi Kinesis Data Streams: Respons dari Kinesis: *`errorCode errorMessage`*

Kemungkinan penyebab kesalahan ini adalah sumber daya Kinesis Data Streams yang disediakan tidak ada. Atau, QLDB tidak memiliki cukup izin untuk menulis catatan data ke aliran data Kinesis yang Anda tentukan.

Pastikan bahwa aliran data Kinesis yang Anda berikan dalam permintaan stream sudah benar. Untuk informasi selengkapnya, lihat Membuat dan memperbarui aliran data dalam Panduan Developer Amazon Kinesis Data Streams.

Juga, verifikasi bahwa Anda menentukan kebijakan untuk aliran data Kinesis yang ditentukan yang memberikan izin layanan (`qldb.amazonaws.com`) QLDB ke tindakan berikut. Untuk informasi selengkapnya, lihat Izin streaming.

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

Pesan: Respons tak terduga dari Kinesis Data Streams saat memvalidasi konfigurasi Kinesis. Tanggapan dari Kinesis: *`errorCode errorMessage`*

Upaya untuk menulis catatan data ke aliran data Kinesis yang disediakan gagal dengan respons kesalahan Kinesis yang disediakan. Untuk informasi selengkapnya tentang kemungkinan penyebabnya, lihat Pemecahan masalah produsen Amazon Kinesis Data Streams dalam Panduan Developer Amazon Kinesis Data Streams.

IllegalArgumentException

Pesan: Tanggal mulai tidak boleh lebih besar dari tanggal akhir.

Baik`InclusiveStartTime` dan`ExclusiveEndTime` harus berada dalam format tanggal dan waktu ISO 8601 dan dalam Waktu Universal Terkoordinasi (UTC).

IllegalArgumentException

Pesan: Tanggal mulai tidak dapat di masa future.

Keduanya`InclusiveStartTime` dan`ExclusiveEndTime` harus dalam format`ISO 8601` tanggal dan waktu dan di UTC.

LimitExceededException

Pesan: Melampaui batas 5 secara bersamaan menjalankan aliran Jurnal ke Kinesis Data Streams

QLDB memberlakukan batas default lima aliran jurnal bersamaan.

## Memverifikasi data jurnal

Bagian ini mencantumkan pengecualian umum yang dapat dikembalikan QLDB saat Anda memverifikasi data jurnal dalam buku besar. Untuk informasi selengkapnya tentang fitur ini, lihat Verifikasi data di Amazon QLDB.

Setiap pengecualian menyertakan pesan kesalahan tertentu, diikuti oleh operasi API yang dapat membuangnya, deskripsi singkat, dan saran untuk kemungkinan solusi.

IllegalArgumentException

Pesan: Nilai Ion yang disediakan tidak valid dan tidak dapat diurai.

Operasi API:`GetDigest, GetBlock, GetRevision`

Pastikan Anda memberikan nilai Amazon Ion yang valid sebelum mencoba ulang permintaan Anda.

IllegalArgumentException

Pesan: Alamat blok yang disediakan tidak valid.

Operasi API:`GetDigest, GetBlock, GetRevision`

Pastikan Anda memberikan alamat pemblokiran yang valid sebelum mencoba ulang permintaan Anda. Alamat blok adalah struktur Amazon Ion yang memiliki dua bidang:`strandId` dan`sequenceNo`.

Misalnya: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`

IllegalArgumentException

Pesan: Nomor urut alamat tip intisari yang disediakan berada di luar catatan komitmen terbaru untai.

Operasi API:`GetDigest, GetBlock, GetRevision`

Alamat tip intisari yang Anda berikan harus memiliki nomor urut kurang dari atau sama dengan nomor urut catatan berkomitmen terbaru untai jurnal. Sebelum mencoba ulang permintaan Anda, pastikan Anda memberikan alamat tip intisari dengan nomor urut yang valid.

IllegalArgumentException

Pesan: ID Strand dari alamat blok yang disediakan tidak valid.

Operasi API:`GetDigest, GetBlock, GetRevision`

Alamat blok yang Anda berikan harus memiliki ID untai yang cocok dengan ID untai jurnal. Sebelum mencoba ulang permintaan Anda, pastikan Anda memberikan alamat pemblokiran dengan ID untai yang valid.

IllegalArgumentException

Pesan: Nomor urut alamat blok yang disediakan berada di luar catatan komitmen terbaru untai.

Operasi API:`GetBlock, GetRevision`

Alamat blok yang Anda berikan harus memiliki nomor urut kurang dari atau sama dengan nomor urut catatan berkomitmen terbaru untai. Sebelum mencoba ulang permintaan Anda, pastikan Anda memberikan alamat pemblokiran dengan nomor urut yang valid.

IllegalArgumentException

Pesan: Strand ID dari alamat blok yang disediakan harus sesuai dengan ID Strand dari alamat tip intisari yang disediakan.

Operasi API:`GetBlock, GetRevision`

Anda hanya dapat memverifikasi revisi atau memblokir dokumen jika ada di untaian jurnal yang sama dengan intisari yang Anda berikan.

IllegalArgumentException

Pesan: Nomor urut alamat blok yang disediakan tidak boleh lebih besar dari nomor urut alamat tip intisari yang disediakan.

Operasi API:`GetBlock, GetRevision`

Anda hanya dapat memverifikasi revisi atau memblokir dokumen jika dicakup oleh intisari yang Anda berikan. Ini berarti bahwa itu berkomitmen untuk jurnal sebelum alamat tip mencerna.

IllegalArgumentException

Pesan: ID Dokumen yang disediakan tidak ditemukan di blok di alamat blok yang ditentukan.

Operasi API:`GetRevision`

ID dokumen yang Anda berikan harus ada di alamat blok yang Anda berikan. Sebelum mencoba ulang permintaan Anda, pastikan kedua parameter ini konsisten.

# Referensi PartiQLDB QLDB

Amazon QLDB mendukung subset bahasa kueri PartiQL. Topik berikut menjelaskan implementasi QLDB dari PartiQL.

> **Note**
>
> - QLDB tidak mendukung semua operasi PartiQL.
> - Semua pernyataan PartiQL di QLDB tunduk pada batas transaksi, sebagaimana didefinisikan dalamKuota dan batasan di Amazon QLDB.
> - Referensi ini menyediakan sintaks dasar dan penggunaan contoh pernyataan PartiQL yang Anda jalankan secara manual di konsol QLDB atau QLDB. Untuk contoh kode yang menunjukkan cara pemrograman menjalankan pernyataan serupa menggunakan driver QLDB, lihat tutorial diMemulai dengan driver.

Topik

# Apa yang dimaksud dengan PartiQL?

PartiQL menyediakan akses kueri yang kompatibel dengan SQL di beberapa penyimpanan data yang berisi data terstruktur, data semi terstruktur, dan data bersarang. Layanan ini banyak digunakan dalam Amazon dan sekarang tersedia sebagai bagian dari banyak halLayanan AWS, termasuk QLDB.

Untuk spesifikasi PartiQL dan tutorial tentang bahasa kueri inti, lihat Dokumentasi PartiQL.

PartiQL memperluas SQL-92 untuk mendukung dokumen dalam format data Amazon Ion. Untuk informasi tentang Amazon Ion, lihatReferensi format data Amazon Ion di Amazon QLDB.

# PartiQL di QLDB

Untuk menjalankan kueri PartiQL di QLDB, Anda dapat menggunakan salah satu hal berikut:

- Editor PartiQL padaAWS Management Console untuk QLDB
- Baris perintah QLDB shell
- Driver QLDB yangAWS disediakan untuk menjalankan kueri secara terprogram

Untuk informasi tentang cara menggunakan metode ini guna mengakses QLDB, lihatMengakses Amazon QLDB.

Untuk mempelajari cara mengontrol akses untuk menjalankan setiap perintah PartiQL pada tabel tertentu, lihatMemulai dengan mode izin standar di Amazon QLDB.

# kiat cepat PartiQL di QLDB

Berikut ini adalah ringkasan singkat tips dan praktik terbaik untuk bekerja dengan PartiQL di QLDB:

- Memahami batas konkurensi dan transaksi — Semua pernyataan, termasuk`SELECT` kueri, tunduk pada konflik kontrol konkurensi (OCC) yang optimis dan batas transaksi, termasuk batas waktu transaksi 30 detik.
- Gunakan indeks - Gunakan indeks kardinalitas tinggi dan jalankan kueri yang ditargetkan untuk mengoptimalkan pernyataan Anda dan menghindari pemindaian tabel penuh. Untuk mempelajari selengkapnya, lihat Mengoptimalkan kinerja kueri.

- Gunakan predikat kesetaraan - Pencarian terindeks memerlukan operator kesetaraan (`=`atau`IN`). Operator ketidaksetaraan (`<>`,`LIKE`,,`BETWEEN`) tidak memenuhi syarat untuk pencarian yang diindeks dan menghasilkan pemindaian tabel penuh.
- Gunakan batin bergabung saja - QLDB mendukung batin bergabung saja. Sebagai praktik terbaik, bergabunglah di bidang yang diindeks untuk setiap tabel yang Anda ikuti. Pilih indeks kardinalitas tinggi untuk kriteria gabungan dan predikat kesetaraan.

# Konvensi referensi Amazon QLDB PartiQL

Bagian ini menjelaskan konvensi yang digunakan untuk menulis sintaks untuk perintah, fungsi, dan ekspresi PartiQL yang dijelaskan dalam referensi PartiQL Amazon QLDB. Konvensi ini tidak harus bingung dengan sintaks dan semantik bahasa query PartiQL itu sendiri.

| Karakter | Deskripsi |
|---|---|
| PENUTUP | Kata dalam huruf besar adalah kata kunci. |
| [ ] | Kurung menunjukkan argumen opsional atau klausa. Beberapa argumen dalam tanda kurung menunjukkan bahwa Anda dapat memilih sejumlah argumen. Selain itu, argumen dalam tanda kurung pada baris terpisah menunjukkan bahwa parser QLDB mengharapkan argumen untuk berada dalam urutan bahwa mereka terdaftar dalam sintaks. |
| \| | Pipa menunjukkan bahwa Anda dapat memilih di antara argumen. |
| *miring merah* | Kata yang dicetak miring merah menunjukkan placeholder. Masukkan nilai yang sesuai di tempat kata yang dicetak miring merah. |
| . . . | Elipsis menunjukkan bahwa Anda dapat mengulangi elemen sebelumnya. |
| ' | Nilai dalam tanda kutip tunggal menunjukkan bahwa Anda harus memasukkan tanda kutip tunggal. Di PartiQL, tanda kutip tunggal menunjukkan nilai string, atau nama bidang dalam struktur Amazon Ion. |
| " | Nilai dalam tanda kutip ganda menunjukkan bahwa Anda harus memasukkan tanda kutip ganda. Dalam PartiQL, tanda kutip ganda menunjukkan pengidentifikasi dikutip. |

| Karakter | Deskripsi |
| --- | --- |
| ` | Nilai di backticks menunjukkan bahwa Anda harus memasukkan backticks. Di PartiQL, backticks menunjukkan nilai literal Ion. |

# Tipe data di Amazon QLDB

Amazon QLDB menyimpan dokumen dalam format Amazon Ion. Amazon Ion adalah format serialisasi data (baik dalam bentuk teks maupun dalam bentuk yang dikodekan biner) yang merupakan superset JSON. Tabel berikut mencantumkan tipe data Ion yang dapat Anda gunakan di dokumen QLDB.

| Tipe data | Deskripsi |
| --- | --- |
| `null` | Nilai null generik |
| `bool` | Nilai boolean |
| `int` | Bilangan bulat yang ditandatangani dengan ukuran sewenang-wenang |
| `decimal` | Bilangan real yang dikodekan desimal dengan presisi sewenang-wenang |
| `float` | Nomor floating point yang dikodekan biner (IEEE 64-bit) |
| `timestamp` | Tanggal/waktu/zona waktu momen presisi sewenang-wenang |
| `string` | Literal teks Unicode |
| `symbol` | Atom simbolik Unicode (pengidentifikasi) |
| `blob` | Data biner dari pengkodean yang ditentukan pengguna |

| Tipe data | Deskripsi |
| --- | --- |
| `clob` | Data teks pengkodean yang ditentukan pengguna |
| `struct` | Koleksi pasangan nama-nilai |
| `list` | Memerintahkan koleksi nilai heterogen |

Lihat dokumen spesifikasi Ion di GitHub situs Amazon untuk daftar lengkap tipe data inti Ion dengan deskripsi lengkap dan detail pemformatan nilai.

# Dokumen QLDB B B B B B B B B B

Amazon QLDB menyimpan catatan data sebagai dokumen, yang hanya`struct` objek Amazon Ion yang dimasukkan ke dalam tabel. Untuk spesifikasi Ion, lihat GitHub situs Amazon Ion.

Topik

- Struktur dokumen ion
- Pemetaan tipe partiQL-ion
- ID Dokumen

## Struktur dokumen ion

Seperti JSON, dokumen QLDB terdiri dari pasangan nama-nilai dalam struktur berikut.

```
{
  name1: value1,
  name2: value2,
  name3: value3,
  ...
  nameN: valueN
}
```

Nama-nama adalah token simbol, dan nilainya tidak dibatasi. Setiap pasangan nama-nilai disebut bidang. Nilai bidang dapat berupa salah satu IonJenis Data, termasuk tipe kontainer: struktur bersarang, daftar, dan daftar struktur.

Juga seperti JSON, `struct` dilambangkan dengan kurung kurawal (`{...}`), dan`list` dilambangkan dengan tanda kurung persegi (`[...]`). Contoh berikut adalah dokumen dari data sampel di[Memulai dengan konsol Amazon QLDB](#) yang berisi nilai-nilai dari berbagai jenis.

```
{
    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**
>
> Di Ion, tanda kutip ganda menunjukkan nilai string, dan simbol yang tidak dikutip mewakili nama bidang. Tapi di PartiQL, tanda kutip tunggal menunjukkan string dan nama bidang. Perbedaan sintaks ini memungkinkan bahasa kueri PartiQL mempertahankan kompatibilitas SQL, dan format data Amazon Ion untuk mempertahankan kompatibilitas JSON. Untuk rincian tentang sintaks dan semantik PartiQL di QLDB, lihat[Kueri Ion dengan PartiQL](#).

## Pemetaan tipe partiQL-ion

Di QLDB, PartiQL memperluas sistem tipe SQL untuk menutupi model data Ion. Pemetaan ini dijelaskan sebagai berikut:

- Jenis skalar SQL ditutupi oleh rekan-rekan Ion mereka. Misalnya:
  - `CHAR`dan`VARCHAR` merupakan urutan Unicode yang memetakan ke`string` tipe Ion.
  - `NUMBER`peta ke`decimal` tipe Ion.
- `struct`Tipe ion setara dengan tupel SQL, yang secara tradisional mewakili baris tabel.
  - Namun, dengan konten terbuka dan tanpa skema, kueri yang bergantung pada sifat yang diurutkan dari tupel SQL tidak didukung (seperti urutan keluaran`SELECT *`).

- Selain ituNULL, PartiQL memilikiMISSING tipe. Ini adalah spesialisasiNULL dan menunjukkan kurangnya lapangan. Jenis ini diperlukan karenastruct bidang Ion mungkin jarang.

### ID Dokumen

QLDB memberikan ID dokumen untuk setiap dokumen yang Anda masukkan ke dalam tabel. Semua ID yang ditetapkan sistem adalah pengidentifikasi unik universal (UUID) yang masing-masing diwakili dalam string yang dikodekan Base62 (misalnya,3Qv67yjXEwB9SjmvkuG6Cp). Untuk informasi selengkapnya, lihat ID unik di Amazon QLDB.

Setiap revisi dokumen diidentifikasi secara unik dengan kombinasi ID dokumen dan nomor versi berbasis nol.

Bidang ID dokumen dan versi disertakan dalam metadata dokumen, yang dapat Anda kueri dalam tampilan berkomitmen (tampilan tabel yang ditentukan sistem). Untuk informasi lebih lanjut tentang tampilan di QLDB, lihatKonsep inti. Untuk mempelajari tentang metadata, lihatMelakukan Kueri Metadata Dokumen.

## Menanyakan Ion dengan PartiQL di Amazon QLDB

Saat Anda melakukan kueri data di Amazon QLDB, Anda menulis pernyataan dalam format PartiQL, tetapi QLDB mengembalikan hasil dalam format Amazon Ion. PartiQL dimaksudkan untuk menjadi SQL-kompatibel, sedangkan Ion adalah perpanjangan dari JSON. Hal ini menyebabkan perbedaan sintaksis antara bagaimana Anda mencatat data dalam pernyataan kueri Anda dibandingkan dengan bagaimana hasil kueri Anda ditampilkan.

Bagian ini menjelaskan sintaks dasar dan semantik untuk menjalankan pernyataan PartiQL secara manual dengan menggunakan konsol QLDB atau shell QLDB.

> **Tip**
>
> Ketika Anda menjalankan query PartiQL pemrograman, praktek terbaik adalah dengan menggunakan pernyataan parameter. Anda dapat menggunakan tanda tanya (?) sebagai placeholder variabel mengikat dalam pernyataan Anda untuk menghindari aturan sintaks ini. Ini juga lebih aman dan efisien.
> Untuk mempelajari selengkapnya, lihat tutorial berikut di Memulai dengan driver:
>
> - Jawa:Tutorial Quick Start |Referensi buku masak

- .NET:Quick start tutorial |Referensi Referensi
- Pergi:Tutorial Quick start |Referensi buku masak
- Node.js:Tutorial Quick Start |Referensi buku masak
- Python:Tutorial Quick Start |Referensi buku masak

Topik

- Sintaksis dan semantik
- Notasi backtick
- Navigasi jalur
- Aliasing
- Spesifikasi PartiQL

## Sintaksis dan semantik

Bila menggunakan konsol QLDB atau shell QLDB untuk query data Ion, berikut ini adalah sintaks dasar dan semantik PartiQL:

Sensitivitas kasus

Semua nama objek sistem QLDB — termasuk nama field, nama tabel, dan nama ledger — adalah case sensitive.

Nilai string

Di Ion, tanda kutip ganda (`"..."`) menunjukkan string.

Dalam PartiQL, tanda kutip tunggal (`'...'`) menunjukkan string.

Simbol dan pengidentifikasi

Di Ion, tanda kutip tunggal (`'...'`) menunjukkan simbol. Sebuah subset dari simbol di Ion disebut pengidentifikasi diwakili oleh teks dikutip.

Dalam PartiQL, tanda kutip ganda (`"..."`) menunjukkan identifier PartiQL dikutip, seperti kata reserved yang digunakan sebagai nama tabel. Teks yang tidak dikutip mewakili pengenal PartiQL biasa, seperti nama tabel yang bukan kata yang dipesan.

Literal ion

Setiap literal Ion dapat dilambangkan dengan backticks (`` `...` ``) dalam pernyataan PartiQL.

Nama Bidang

Nama bidang ion adalah simbol peka huruf besar dan kecil. PartiQL memungkinkan Anda menunjukkan nama field dengan tanda kutip tunggal dalam sebuah pernyataan DML. Ini adalah alternatif singkatan untuk menggunakan`cast` fungsi PartiQL untuk menentukan simbol. Ini juga lebih intuitif daripada menggunakan backtick untuk menunjukkan simbol Ion literal.

## Literal

Literal dari bahasa query PartiQL sesuai dengan tipe data Ion, sebagai berikut:

Skalar

Ikuti sintaks SQL bila berlaku, seperti yang dijelaskan diPemetaan tipe partiQL-ion bagian. Misalnya:

- `5`
- `'foo'`
- `null`

Struct

Juga dikenal sebagai tupel atau objek dalam banyak format dan model data lainnya.

Dilambangkan dengan kurung kurawal (`{...}`) dengan`struct` elemen dipisahkan dengan koma.

- `{ 'id' : 3, 'arr': [1, 2] }`

Daftar

Juga dikenal sebagai array.

Dilambangkan dengan tanda kurung persegi (`[...]`) dengan elemen daftar dipisahkan dengan koma.

- `[ 1, 'foo' ]`

Tas

Koleksi unordered di PartiQL.

Dilambangkan dengan kurung sudut ganda (`<<...>>`) dengan elemen tas dipisahkan dengan koma. Di QLDB, meja dapat dianggap sebagai tas. Namun, tas tidak dapat bersarang di dalam dokumen di meja.

- `<< 1, 'foo' >>`

### Contoh

Berikut ini adalah contoh dari sintaks untuk`INSERT` pernyataan dengan berbagai jenis Ion.

```
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`
}
```

## Notasi backtick

PartiQL sepenuhnya mencakup semua tipe data Ion, sehingga Anda dapat menulis pernyataan apa pun tanpa menggunakan backtick. Tapi ada kasus di mana sintaks literal Ion ini dapat membuat pernyataan Anda lebih jelas dan lebih ringkas.

Misalnya, untuk menyisipkan dokumen dengan Ion timestamp dan simbol nilai-nilai, Anda dapat menulis pernyataan berikut menggunakan murni sintaks PartiQL saja.

```
INSERT INTO myTable VALUE
{
    'myTimestamp': to_timestamp('2019-09-04T'),
    'mySymbol': cast('foo' as symbol)
```

```
}
```

Ini cukup verbose, jadi sebagai gantinya, Anda dapat menggunakan backticks untuk menyederhanakan pernyataan Anda.

```
INSERT INTO myTable VALUE
{
    'myTimestamp': `2019-09-04T`,
    'mySymbol': `foo`
}
```

Anda juga dapat melampirkan seluruh struktur di backticks untuk menyimpan beberapa penekanan tombol lagi.

```
INSERT INTO myTable VALUE
`{
    myTimestamp: 2019-09-04T,
    mySymbol: foo
}`
```

> **Important**
>
> String dan simbol adalah kelas yang berbeda di PartiQL. Ini berarti bahwa bahkan jika mereka memiliki teks yang sama, mereka tidak sama. Misalnya, ekspresi PartiQL berikut mengevaluasi nilai Ion yang berbeda.
>
> ```
> 'foo'
> ```
>
> ```
> `foo`
> ```

## Navigasi jalur

Ketika menulis bahasa manipulasi data (DML/DL) atau pernyataan query, Anda dapat mengakses bidang dalam struktur bersarang menggunakan langkah-langkah jalan. PartiQL mendukung notasi dot untuk mengakses nama field dari struktur induk. Contoh berikut mengakses`Model` bidang induk`Vehicle`.

```
Vehicle.Model
```

Untuk mengakses elemen tertentu dari daftar, Anda dapat menggunakan operator kurung persegi untuk menunjukkan nomor urut berbasis nol. Contoh berikut mengakses elemen`SecondaryOwners` dengan nomor urut2. Dengan kata lain, ini adalah elemen ketiga dari daftar.

```
SecondaryOwners[2]
```

## Aliasing

QLDB mendukung konten terbuka dan skema. Jadi, ketika Anda mengakses bidang tertentu dalam sebuah pernyataan, cara terbaik untuk memastikan bahwa Anda mendapatkan hasil yang Anda harapkan adalah dengan menggunakan alias. Misalnya, jika Anda tidak menentukan alias eksplisit, sistem akan menghasilkan alias implisit untuk`FROM` sumber Anda.

```
SELECT VIN FROM Vehicle
--is rewritten to
SELECT Vehicle.VIN FROM Vehicle AS Vehicle
```

Tetapi hasilnya tidak dapat diprediksi untuk konflik nama bidang. Jika bidang lain bernama`VIN` ada dalam struktur bersarang di dalam dokumen,`VIN` nilai yang dikembalikan oleh kueri ini mungkin akan mengejutkan Anda. Sebagai praktik terbaik, tulis pernyataan berikut. Query ini menyatakan`v` sebagai alias yang berkisar di atas`Vehicle` meja. `AS`Kata kunci adalah opsional.

```
SELECT v.VIN FROM Vehicle [ AS ] v
```

Aliasing sangat berguna ketika pathing ke koleksi bersarang dalam dokumen. Misalnya, pernyataan berikut menyatakan`o` sebagai alias yang berkisar atas koleksi`VehicleRegistration.Owners`.

```
SELECT o.SecondaryOwners
FROM VehicleRegistration AS r, @r.Owners AS o
```

`@`Karakter secara teknis opsional di sini. Tetapi secara eksplisit menunjukkan bahwa Anda ingin`Owners` struktur dalam`VehicleRegistration`, bukan koleksi yang berbeda bernama`Owners` (jika ada).

## Spesifikasi PartiQL

Untuk informasi lebih lanjut tentang bahasa query PartiQL, lihat Spesifikasi PartiQL.

# Perintah PartiQL di Amazon QLDB

PartiQL memperluas SQL-92 untuk mendukung dokumen dalam format data Amazon Ion. Amazon QLDB B B B B B B B B B B B B B B B B B B B B B B B B.

Untuk mempelajari cara mengontrol akses untuk menjalankan setiap perintah PartiQL pada tabel tertentu, lihatMemulai dengan mode izin standar di Amazon QLDB.

> **Note**
>
> - QLDB tidak mendukung semua perintah PartiQL.
> - Semua pernyataan PartiQL di QLDB tunduk pada batas transaksi, sebagaimana didefinisikan dalamKuota dan batasan di Amazon QLDB.
> - Referensi ini menyediakan sintaks dasar dan penggunaan contoh pernyataan PartiQL yang Anda jalankan secara manual di konsol QLDB atau QLDB. Untuk contoh kode yang menunjukkan cara menjalankan pernyataan serupa menggunakan bahasa pemrograman yang didukung, lihat tutorial diMemulai dengan driver.

## pernyataan DDL (bahasa definisi data)

Bahasa definisi data (DL) adalah kumpulan pernyataan PartiQL yang Anda gunakan untuk mengelola objek basis data, seperti tabel dan indeks. Anda menggunakan DDL untuk membuat dan menjatuhkan benda-benda ini.

- CREATE INDEX
- CREATE TABLE
- DROP INDEX
- MEJA DROP
- TABEL UNDROP

## pernyataan DML-nya (bahasa manipulasi data)

Bahasa manipulasi data (DL) adalah kumpulan pernyataan PartiQL yang Anda gunakan untuk mengelola data dalam tabel QLDB. Anda menggunakan pernyataan DML untuk menambah, mengubah, atau menghapus data dalam sebuah tabel.

Berikut DML dan kueri bahasa pernyataan yang didukung:

- HAPUS
- DARI (INSERT, HAPUS, atau SET)
- SISIPKAN
- Pilih
- PERBARUI

## CREATE INDEX perintah di Amazon QLDB

Di Amazon QLDB, gunakan`CREATE INDEX` perintah untuk membuat indeks untuk bidang dokumen di atas meja.

Untuk mempelajari cara mengontrol akses untuk menjalankan perintah PartiQL ini pada tabel tertentu, lihatMemulai dengan mode izin standar di Amazon QLDB.

> **Important**
>
> QLDB membutuhkan indeks untuk secara efisien mencari dokumen. Tanpa indeks, QLDB perlu melakukan pemindaian meja penuh saat membaca dokumen. Hal ini dapat menyebabkan masalah kinerja pada tabel besar, termasuk konflik konkurensi dan batas waktu transaksi.
>
> Untuk menghindari pemindaian tabel, Anda harus menjalankan pernyataan dengan klausa`WHERE` predikat menggunakan operator kesetaraan (`=`or`IN`) pada bidang yang diindeks atau ID dokumen. Untuk informasi selengkapnya, lihat Mengoptimalkan kinerja kueri.

Perhatikan kendala berikut saat membuat indeks:

- Indeks hanya dapat dibuat pada satu bidang tingkat atas. Indeks komposit, bersarang, unik, dan berbasis fungsi tidak didukung.

- Anda dapat membuat indeks pada jenis data Ion apa pun, termasuk`list` dan`struct`. Namun, Anda hanya dapat melakukan pencarian terindeks dengan kesetaraan seluruh nilai Ion terlepas dari jenis Ion. Misalnya, saat menggunakan`list` tipe sebagai indeks, Anda tidak dapat melakukan pencarian terindeks oleh satu item di dalam daftar.
- Kinerja kueri ditingkatkan hanya jika Anda menggunakan predikat kesetaraan; misalnya,`WHERE indexedField = 123` atau`WHERE indexedField IN (456, 789)`.

  QLDB tidak menghormati ketidaksetaraan dalam predikat query. Akibatnya, pemindaian rentang yang difilter tidak diimplementasikan.
- Nama kolom yang diindeks bersifat huruf kapital dan maksimum 128 karakter.
- Pembuatan indeks di QLDB bersifat asinkron. Jumlah waktu yang dibutuhkan untuk menyelesaikan membangun indeks pada tabel yang tidak kosong bervariasi tergantung pada ukuran tabel. Untuk informasi selengkapnya, lihat Mengelola indeks.

Topik



## Sintaks

```
CREATE INDEX ON table_name (field)
```

## Parameter

***nama meja***

Nama tabel tempat Anda ingin membuat indeks. Tabel harus sudah ada.

Nama tabel peka huruf besar/kecil.

***bidang***

Nama field dokumen yang untuk membuat indeks. Bidang harus berupa atribut tingkat atas.

Nama kolom yang diindeks bersifat huruf kapital dan maksimum 128 karakter.

Anda dapat membuat indeks pada jenis data Amazon Ion apa pun, termasuk`list` dan`struct`. Namun, Anda hanya dapat melakukan pencarian terindeks dengan kesetaraan seluruh nilai Ion terlepas dari jenis Ion. Misalnya, saat menggunakan`list` tipe sebagai indeks, Anda tidak dapat melakukan pencarian terindeks oleh satu item di dalam daftar.

### Nilai kembali

`tableId`- ID unik dari tabel yang Anda buat indeks.

### Contoh

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

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

### Menjalankan pemrograman menggunakan driver

Untuk mempelajari cara menjalankan pernyataan ini secara terprogram menggunakan driver QLDB, lihat tutorial berikut dalam Memulai driver:

- Jawa:Tutorial Quick Start |Referensi buku masak
- .NET:Quick start tutorial |Referensi Referensi
- Pergi:Tutorial Quick start |Referensi buku masak
- Node.js:Tutorial Quick Start |Referensi buku masak
- Python:Tutorial Quick Start |Referensi buku masak

## CREATE TABLE perintah di Amazon QLDB

Di Amazon QLDB, gunakan`CREATE TABLE` perintah untuk membuat tabel baru.

Tabel memiliki nama sederhana tanpa ruang nama. QLDB mendukung konten terbuka dan tidak menegakkan skema, sehingga Anda tidak menentukan atribut atau tipe data saat membuat tabel.

> **Note**
>
> Untuk mempelajari cara mengontrol akses untuk menjalankan perintah PartiQL ini dalam buku besar, lihatMemulai dengan mode izin standar di Amazon QLDB.

Topik



## Sintaks

```
CREATE TABLE table_name [ WITH (aws_tags = `{'key': 'value'}`) ]
```

## Parameter

***nama meja***

Nama unik dari tabel untuk dibuat. Tabel aktif dengan nama yang sama tidak boleh ada. Berikut ini adalah kendala penamaan:

- Harus hanya berisi 1-128 karakter alfanumerik atau garis bawah.
- Harus memiliki sebuah huruf atau garis bawah untuk karakter pertama.
- Dapat memiliki kombinasi karakter alfanumerik dan garis bawah untuk karakter yang tersisa.
- Peka terhadap huruf besar dan kecil.
- Tidak harus berupa kata QLDB PartiQL reserved.

'***key***': '***nilai***'

(Opsional) Tag untuk dilampirkan ke sumber daya tabel selama pembuatan. Setiap tag didefinisikan sebagai pasangan kunci-nilai, di mana kunci dan nilai masing-masing dilambangkan dengan tanda kutip tunggal. Setiap pasangan key-value didefinisikan di dalam struktur Amazon Ion yang dilambangkan dengan backtick.

Tabel penandaan pada pembuatan saat ini didukung untuk buku*STANDARD* besar dalam mode izin saja.

## Nilai kembali

`tableId`- ID unik dari tabel yang Anda buat.

## Beri tag tabel saat penciptaan

> **Note**
>
> Tabel penandaan pada pembuatan saat ini didukung untuk bukuSTANDARD besar dalam mode izin saja.

Opsional, Anda dapat menandai sumber daya tabel Anda dengan menentukan tag dalam`CREATE TABLE` pernyataan. Untuk informasi selengkapnya tentang tanda, lihat Pemberian tag pada sumber daya Amazon QLDB. Contoh berikut membuat tabel bernama`Vehicle` dengan tag`environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

Tagging tabel pada penciptaan membutuhkan akses ke kedua`qldb:PartiQLCreateTable` dan`qldb:TagResource` tindakan. Untuk mempelajari lebih lanjut tentang izin untuk sumber daya QLDB, lihatBagaimana Amazon QLDB bekerja dengan IAM.

Dengan menandai sumber daya saat sedang dibuat, Anda dapat menghilangkan kebutuhan untuk menjalankan skrip penandaan khusus setelah pembuatan sumber daya. Setelah tabel ditandai, Anda dapat mengontrol akses ke tabel berdasarkan tag tersebut. Misalnya, Anda dapat memberikan akses penuh hanya ke tabel yang memiliki tanda tertentu. Untuk contoh kebijakan JSON, lihatAkses penuh ke semua tindakan berdasarkan tag tabel.

## Contoh

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'development'}`)
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'key1': 'value1', 'key2': 'value2'}`)
```

### Menjalankan pemrograman menggunakan driver

Untuk mempelajari cara menjalankan pernyataan ini secara terprogram menggunakan driver QLDB, lihat tutorial berikut dalam Memulai driver:

- Jawa:Tutorial Quick Start |Referensi buku masak
- .NET:Quick start tutorial |Referensi Referensi
- Pergi:Tutorial Quick start |Referensi buku masak
- Node.js:Tutorial Quick Start |Referensi buku masak
- Python:Tutorial Quick Start |Referensi buku masak

## DELETE perintah di Amazon QLDB

Di Amazon QLDB, gunakan`DELETE` perintah untuk menandai dokumen aktif sebagai dihapus dalam tabel dengan membuat revisi dokumen yang baru namun akhir. Revisi akhir ini menunjukkan bahwa dokumen dihapus. Operasi ini mengakhiri siklus hidup dokumen, yang berarti bahwa tidak ada revisi dokumen lebih lanjut dengan ID dokumen yang sama dapat dibuat.

Operasi ini tidak dapat diubah. Anda masih dapat query sejarah revisi dokumen dihapus dengan menggunakanFungsi Riwayat.

> **Note**
>
> Untuk mempelajari cara mengontrol akses untuk menjalankan perintah PartiQL ini pada tabel tertentu, lihatMemulai dengan mode izin standar di Amazon QLDB.

Topik

- Sintaks
- Parameter
- Nilai kembali
- Contoh
- Menjalankan pemrograman menggunakan driver

## Sintaks

```
DELETE FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
```

## Parameter

***nama meja***

Nama tabel pengguna yang berisi data yang akan dihapus. Pernyataan DMLhanya didukung dalam tampilan pengguna default. Setiap pernyataan hanya dapat berjalan pada satu meja.

SEBAGAI ***table_alias***

(Opsional) Sebuah alias yang ditentukan pengguna yang berkisar di atas tabel yang akan dihapus dari. ASKata kunci adalah opsional.

OLEH ***id_alias***

(Opsional) Alias yang ditentukan pengguna yang mengikat ke bidang`id` metadata setiap dokumen dalam kumpulan hasil. Alias harus dinyatakan dalam`FROM` klausa menggunakan`BY` kata kunci. Hal ini berguna ketika Anda ingin memfilter pada ID dokumen saat query tampilan pengguna default. Untuk informasi selengkapnya, lihat Menggunakan klausa BY untuk query ID dokumen.

***Kondisi*** WHERE

Kriteria seleksi untuk dokumen yang akan dihapus.

> **Note**
>
> Jika Anda menghapus`WHERE` klausul, semua dokumen dalam tabel akan dihapus.

## Nilai kembali

`documentId`- ID unik dari setiap dokumen yang Anda hapus.

## Contoh

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

### Menjalankan pemrograman menggunakan driver

Untuk mempelajari cara menjalankan pernyataan ini secara terprogram menggunakan driver QLDB, lihat tutorial berikut dalam Memulai driver:

- Jawa:Tutorial Quick Start |Referensi buku masak
- .NET:Quick start tutorial |Referensi Referensi
- Pergi:Tutorial Quick start |Referensi buku masak
- Node.js:Tutorial Quick Start |Referensi buku masak
- Python:Tutorial Quick Start |Referensi buku masak

## DROP INDEX perintah di Amazon QLDB

Di Amazon QLDB, gunakan`DROP INDEX` perintah untuk menghapus indeks di atas meja.

> **Note**
>
> Untuk mempelajari cara mengontrol akses untuk menjalankan perintah PartiQL ini pada tabel tertentu, lihatMemulai dengan mode izin standar di Amazon QLDB.

Topik

- Sintaksis
- Parameter
- Nilai kembali
- Contoh

### Sintaksis

```
DROP INDEX "indexId" ON table_name WITH (purge = true)
```

> **Note**
>
> Klausa`WITH (purge = true)` ini diperlukan untuk semua`DROP INDEX` pernyataan, dan saat`true` ini satu-satunya nilai yang didukung.

Kata kuncipurge harus mengandung huruf kapital dan harus berupa huruf kecil.

### Parameter

"***IndexID***"

ID unik dari indeks untuk menjatuhkan, dilambangkan dengan tanda kutip ganda.

PADA ***table_name***

Nama tabel yang ingin Anda turunkan indeks.

### Nilai kembali

`tableId`- ID unik dari tabel yang indeksnya Anda jatuhkan.

### Contoh

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

## DROP perintah TABLE di Amazon QLDB

Di Amazon QLDB, gunakan`DROP TABLE` perintah untuk menonaktifkan tabel yang ada. Anda dapat menggunakanTABEL UNDROP pernyataan untuk mengaktifkannya kembali. Menonaktifkan atau mengaktifkan kembali tabel tidak berpengaruh pada dokumen atau indeksnya.

> **Note**
>
> Untuk mempelajari cara mengontrol akses untuk menjalankan perintah PartiQL ini pada tabel tertentu, lihatMemulai dengan mode izin standar di Amazon QLDB.

Topik

### Sintaks

```
DROP TABLE table_name
```

### Parameter

***nama meja***

Nama tabel untuk dinonaktifkan. Tabel harus sudah ada dan memiliki status`ACTIVE`.

### Nilai kembali

`tableId`- ID unik dari tabel yang Anda nonaktifkan.

### Contoh

```
DROP TABLE VehicleRegistration
```

## DARI (INSERT, REMOVE, atau SET) perintah di Amazon QLDB

Di Amazon QLDB, pernyataan yang dimulai dengan`FROM` adalah ekstensi PartiQL yang memungkinkan Anda untuk menyisipkan dan menghapus elemen tertentu dalam dokumen. Anda juga dapat menggunakan pernyataan ini untuk memperbarui elemen yang ada dalam dokumen, mirip dengan[PERBARUI](#) perintah.

> **Note**
>
> Untuk mempelajari cara mengontrol akses untuk menjalankan perintah PartiQL ini pada tabel tertentu, lihat[Memulai dengan mode izin standar di Amazon QLDB](#).

Topik

- Menjalankan pemrograman menggunakan driver

## Sintaksis

### DARI-INSERT

Menyisipkan elemen baru dalam dokumen yang ada. Untuk memasukkan dokumen tingkat atas baru ke dalam tabel, Anda harus menggunakanSISIPKAN.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
INSERT INTO element VALUE data [ AT key_name ]
```

### DARI-HAPUS

Menghapus elemen yang ada dalam dokumen, atau menghapus seluruh dokumen tingkat atas. Yang terakhir ini semantik sama denganHAPUS sintaks tradisional.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
REMOVE element
```

### DARI-SET

Memperbarui satu atau lebih elemen dalam dokumen. Jika elemen tidak ada, ini dimasukkan. Ini semantik sama denganPERBARUI sintaks tradisional.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
SET element = data [, element = data, ... ]
```

## Parameter

***nama meja***

Nama tabel pengguna yang berisi data yang akan diubah. Pernyataan DMLhanya didukung dalam tampilan pengguna default. Setiap pernyataan hanya dapat berjalan pada satu meja.

Dalam klausa ini, Anda juga dapat menyertakan satu atau lebih koleksi yang bersarang dalam tabel tertentu. Untuk rincian lebih lanjut, lihat Koleksi Nest.

SEBAGAI *table_alias*

(Opsional) Alias yang ditentukan pengguna yang berkisar di atas tabel yang akan dimodifikasi. Semua alias tabel yang digunakan dalam`SET`,,`REMOVE``INSERT INTO`, atau`WHERE` klausa harus dinyatakan dalam`FROM` klausa. `AS`Kata kunci adalah opsional.

OLEH *id_alias*

(Opsional) Alias yang ditentukan pengguna yang mengikat ke bidang`id` metadata setiap dokumen dalam kumpulan hasil. Alias harus dinyatakan dalam`FROM` klausa menggunakan`BY` kata kunci. Hal ini berguna ketika Anda ingin memfilter pada ID dokumen saat query tampilan pengguna default. Untuk informasi selengkapnya, lihat Menggunakan klausa BY untuk query ID dokumen.

***Kondisi*** WHERE

Kriteria seleksi dokumen yang akan diubah.

> **Note**
>
> Jika Anda menghapus`WHERE` klausul, semua dokumen dalam tabel akan diubah.

***elemen***

Elemen dokumen yang akan dibuat atau diubah.

***data***

Nilai baru untuk elemen.

DI *key_name*

Sebuah nama kunci yang akan ditambahkan dalam dokumen yang akan dimodifikasi. Anda harus menentukan yang sesuai`VALUE` bersama dengan nama kunci. Hal ini diperlukan untuk memasukkan nilai baru posisi`AT` tertentu dalam dokumen.

## Koleksi Nest

Meskipun Anda dapat menjalankan pernyataan DMLS pada satu meja saja, Anda dapat menentukan koleksi bersarang dalam dokumen dalam tabel itu sebagai sumber tambahan. Setiap alias yang Anda mendeklarasikan untuk koleksi bersarang dapat digunakan dalam`WHERE` klausa dan`SET`,`INSERT INTO`, atau`REMOVE` klausa.

Misalnya,`FROM` sumber pernyataan berikut mencakup`VehicleRegistration` tabel dan`Owners.SecondaryOwners` struktur bersarang.

```
FROM VehicleRegistration r, @r.Owners.SecondaryOwners o
WHERE r.VIN = '1N4AL11D75C109151' AND o.PersonId = 'abc123'
SET o.PersonId = 'def456'
```

Contoh ini memperbarui elemen tertentu dari`SecondaryOwners` daftar yang memiliki`PersonId` dari`'abc123'` dalam`VehicleRegistration` dokumen yang memiliki`VIN` dari`'1N4AL11D75C109151'`. Ekspresi ini memungkinkan Anda menentukan elemen daftar dengan nilainya daripada indeksnya.

## Nilai kembali

`documentId`- ID unik dari setiap dokumen yang Anda perbarui atau hapus.

## Contoh

Memodifikasi elemen dalam dokumen. Jika elemen tidak ada, ini dimasukkan.

```
FROM Vehicle AS v
WHERE v.VIN = '1N4AL11D75C109151' AND v.Color = 'Silver'
SET v.Color = 'Shiny Gray'
```

Memodifikasi atau menyisipkan elemen dan filter pada bidang`id` metadata dokumen sistem-ditugaskan.

```
FROM Vehicle AS v BY v_id
WHERE v_id = 'documentId'
SET v.Color = 'Shiny Gray'
```

Memodifikasi`PersonId` bidang elemen pertama dalam`Owners.SecondaryOwners` daftar dalam dokumen.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
```

Hapus elemen yang ada dalam dokumen.

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

Hapus seluruh dokumen dari tabel.

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

Hapus elemen pertama dari`Owners.SecondaryOwners` daftar dalam dokumen dalam`VehicleRegistration` tabel.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
REMOVE r.Owners.SecondaryOwners[0]
```

Masukkan`{'Mileage':26500}` sebagai pasangan nama-nilai tingkat atas dalam dokumen dalam`Vehicle` tabel.

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

Tambahkan`{'PersonId':'abc123'}` sebagai pasangan nama-nilai di`Owners.SecondaryOwners` bidang dokumen dalam`VehicleRegistration` tabel. Perhatikan bahwa`Owners.SecondaryOwners` harus sudah ada dan harus menjadi tipe data daftar untuk pernyataan ini untuk menjadi valid. Jika tidak, kata kunci`AT` diperlukan dalam`INSERT INTO` klausa.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
```

Masukkan`{'PersonId':'abc123'}` sebagai elemen pertama dalam`Owners.SecondaryOwners` daftar yang ada dalam dokumen.

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

```
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
```

Tambahkan beberapa pasangan nama-nilai ke`Owners.SecondaryOwners` daftar yang ada dalam dokumen.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
```

### Menjalankan pemrograman menggunakan driver

Untuk mempelajari cara menjalankan pernyataan ini secara terprogram menggunakan driver QLDB, lihat tutorial berikut dalam Memulai driver:

- Jawa:Tutorial Quick Start |Referensi buku masak
- .NET:Quick start tutorial |Referensi Referensi
- Pergi:Tutorial Quick start |Referensi buku masak
- Node.js:Tutorial Quick Start |Referensi buku masak
- Python:Tutorial Quick Start |Referensi buku masak

## INSERT perintah di Amazon QLDB

Di Amazon QLDB, gunakan`INSERT` perintah untuk menambahkan satu atau beberapa dokumen Amazon Ion ke tabel.

> **Note**
>
> Untuk mempelajari cara mengontrol akses untuk menjalankan perintah PartiQL ini pada tabel tertentu, lihatMemulai dengan mode izin standar di Amazon QLDB.

Topik

- Sintaksis
- Parameter
- Nilai kembali

- Contoh
- Menjalankan pemrograman menggunakan driver

## Sintaksis

Masukkan dokumen tunggal.

```
INSERT INTO table_name VALUE document
```

Masukkan beberapa dokumen.

```
INSERT INTO table_name << document, document, ... >>
```

## Parameter

***nama meja***

Nama tabel pengguna di mana Anda ingin menyisipkan data. Tabel harus sudah ada. Pernyataan DMLhanya didukung dalam tampilan pengguna default.

***dokumen***

Dokumen QLDB yang valid. Anda harus menentukan setidaknya satu dokumen. Beberapa dokumen harus dipisahkan dengan koma.

Dokumen harus dilambangkan dengan kurung kurawal (`{...}`).

Setiap nama bidang dalam dokumen adalah simbol Ion peka huruf besar yang dapat dilambangkan dengan tanda kutip tunggal (`'...'`) di PartiQL.

Nilai string juga dilambangkan dengan tanda petik tunggal (`'...'`) di PartiQL.

Setiap literal Ion dapat dilambangkan dengan backticks (`` `...` ``).

> **Note**
>
> Kurung sudut ganda (`<<...>>`) menunjukkan koleksi unordered (dikenal sebagai tas di PartiQL) dan diperlukan hanya jika Anda ingin memasukkan beberapa dokumen.

## Nilai kembali

`documentId`- ID unik dari setiap dokumen yang Anda masukkan.

## Contoh

Masukkan dokumen tunggal.

```
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`
}
```

Pernyataan ini mengembalikan ID unik dari dokumen yang Anda masukkan, sebagai berikut.

```
{
    documentId: "2kKuOPNB07D2iTPBrUTWGl"
}
```

Masukkan beberapa dokumen.

```
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'
}
>>
```

Pernyataan ini mengembalikan ID unik dari setiap dokumen yang Anda masukkan, sebagai berikut.

```
{
    documentId: "6WXzLscsJ3bDWW97Dy8nyp"
},
{
    documentId: "35e0ToZyTGJ7LGvcwrkX65"
},
{
    documentId: "BVHPcH612o7JROQ4yP8jiH"
}
```

## Menjalankan pemrograman menggunakan driver

Untuk mempelajari cara menjalankan pernyataan ini secara terprogram menggunakan driver QLDB, lihat tutorial berikut dalam Memulai driver:

- Jawa:Tutorial Quick Start |Referensi buku masak
- .NET:Quick start tutorial |Referensi Referensi
- Pergi:Tutorial Quick start |Referensi buku masak
- Node.js:Tutorial Quick Start |Referensi buku masak
- Python:Tutorial Quick Start |Referensi buku masak

# perintah SELECT di Amazon QLDB

Di Amazon QLDB, gunakan`SELECT` perintah untuk mengambil data dari satu atau beberapa tabel. Setiap`SELECT` kueri di QLDB diproses dalam transaksi dan tunduk pada batas batas waktu transaksi.

Urutan hasilnya tidak spesifik dan dapat bervariasi untuk setiap`SELECT` kueri. Anda tidak boleh bergantung pada urutan hasil untuk kueri apa pun di QLDB.

Untuk mempelajari cara mengontrol akses untuk menjalankan perintah PartiQL ini pada tabel tertentu, lihatMemulai dengan mode izin standar di Amazon QLDB.

> **Warning**
>
> Ketika Anda menjalankan query di QLDB tanpa pencarian diindeks, memanggil scan tabel penuh. PartiQL mendukung query tersebut karena SQL kompatibel. Namun, jangan jalankan pemindaian tabel untuk kasus penggunaan produksi di QLDB. Pemindaian tabel dapat menyebabkan masalah kinerja pada tabel besar, termasuk konflik konkurensi dan batas waktu transaksi.
>
> Untuk menghindari pemindaian tabel, Anda harus menjalankan pernyataan dengan klausa`WHERE` predikat menggunakan operator kesetaraan pada bidang yang diindeks atau ID dokumen; misalnya,`WHERE indexedField = 123` atau`WHERE indexedField IN (456, 789)`. Untuk informasi selengkapnya, lihat Mengoptimalkan kinerja kueri.

Topik

- Sintaks
- Parameter
- Bergabung
- Keterbatasan kueri Nest
- Contoh
- Menjalankan pemrograman menggunakan driver

## Sintaks

```
SELECT [ VALUE ] expression [ AS field_alias ] [, expression, ... ]
FROM source [ AS source_alias ] [ AT idx_alias ] [ BY id_alias ] [, source, ... ]
```

```
[ WHERE condition ]
```

## Parameter

NILAI

Kualifikasi untuk ekspresi Anda yang membuat kueri mengembalikan nilai tipe data mentah, daripada nilai yang dibungkus dalam struktur tupel.

***ekspresi***

Proyeksi yang terbentuk dari* wildcard atau daftar proyeksi dari satu atau beberapa dokumen atau lebih dari satu dokumen atau lebih dari satu dokumen atau lebih dari satu dokumen atau lebih. Ekspresi dapat terdiri dari panggilan ke fungsi PartiQL atau bidang yang diubah oleh operator PartiQL.

SEBAGAI ***field_alias***

(Opsional) Alias sementara yang ditentukan pengguna untuk bidang yang digunakan dalam kumpulan hasil akhir. ASKata kunci adalah opsional.

Jika Anda tidak menentukan alias untuk ekspresi yang bukan nama bidang sederhana, kumpulan hasil akan menerapkan nama default ke bidang tersebut.

DARI ***sumber***

Sumber yang akan ditanyakan. Satu-satunya sumber yang saat ini didukung adalah nama tabel, inner bergabung antara tabel,SELECT query bersarang (tundukKeterbatasan kueri Nest), dan fungsi sejarah panggilan untuk tabel.

Anda harus menentukan setidaknya satu sumber. Beberapa sumber harus dipisahkan dengan koma.

SEBAGAI ***source_alias***

(Opsional) Alias yang ditentukan pengguna yang berkisar di atas sumber yang akan ditanyakan. Semua sumber alias yang digunakan dalamWHERE klausaSELECT OR harus dinyatakan dalamFROM klausa. ASKata kunci adalah opsional.

DI ***idx_alias***

(Opsional) Sebuah alias yang ditetapkan pengguna yang mengikat ke indeks (ordinal) jumlah setiap elemen dalam daftar dari sumber. Alias harus dinyatakan dalamFROM klausa menggunakanAT kata kunci.

OLEH *id_alias*

(Opsional) Alias yang ditentukan pengguna yang mengikat ke bidang`id` metadata setiap dokumen dalam kumpulan hasil. Alias harus dinyatakan dalam`FROM` klausa menggunakan`BY` kata kunci. Hal ini berguna ketika Anda ingin proyek atau filter pada ID dokumen sementara query tampilan pengguna default. Untuk informasi selengkapnya, lihat Menggunakan klausa BY untuk query ID dokumen.

***Kondisi*** WHERE

Kriteria seleksi dan bergabung kriteria (jika berlaku) untuk query.

> **Note**
>
> Jika Anda menghapus`WHERE` klausul, semua dokumen dalam tabel akan diambil.

## Bergabung

Hanya inner join yang saat ini didukung. Anda dapat menulis kueri gabungan batin menggunakan`INNER JOIN` klausa eksplisit, sebagai berikut. Dalam sintaks ini,`JOIN` harus dipasangkan dengan`ON`, dan`INNER` kata kunci adalah opsional.

```
SELECT expression
FROM table1 AS t1 [ INNER ] JOIN table2 AS t2
ON t1.element = t2.element
```

Atau, Anda dapat menulis inner bergabung menggunakan sintaks implisit, sebagai berikut.

```
SELECT expression
FROM table1 AS t1, table2 AS t2
WHERE t1.element = t2.element
```

## Keterbatasan kueri Nest

Anda dapat menulis query bersarang (subqueries) dalam`SELECT` ekspresi dan dalam`FROM` sumber. Pembatasan utama adalah bahwa hanya kueri terluar yang dapat mengakses lingkungan database global. Misalnya, anggaplah bahwa Anda memiliki buku besar dengan

tabel`VehicleRegistration` dan`Person`. Berikut query bersarang tidak valid karena batin`SELECT` mencoba untuk mengakses`Person`.

```
SELECT r.VIN,
    (SELECT p.PersonId FROM Person AS p WHERE p.PersonId =
 r.Owners.PrimaryOwner.PersonId) AS PrimaryOwner
FROM VehicleRegistration AS r
```

Sedangkan query bersarang berikut ini valid.

```
SELECT r.VIN, (SELECT o.PrimaryOwner.PersonId FROM @r.Owners AS o) AS PrimaryOwner
FROM VehicleRegistration AS r
```

## Contoh

Query berikut menunjukkan dasar`SELECT` semua wildcard dengan klausa`WHERE` predikat standar yang menggunakan`IN` operator.

```
SELECT * FROM Vehicle
WHERE VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Berikut ini menunjukkan`SELECT` proyeksi dengan filter string.

```
SELECT FirstName, LastName, Address
FROM Person
WHERE Address LIKE '%Seattle%'
AND GovId = 'LEWISR261LL'
```

Berikut ini menunjukkan subquery berkorelasi yang meratakan data bersarang. Perhatikan bahwa`@` karakter secara teknis opsional di sini. Tetapi secara eksplisit menunjukkan bahwa Anda ingin`Owners` struktur yang bersarang di dalam`VehicleRegistration`, bukan koleksi yang berbeda bernama`Owners` (jika ada). Untuk konteks lebih lanjut, lihatData yang tertumpuk di bagian Bekerja dengan data dan riwayat.

```
SELECT
    r.VIN,
    o.SecondaryOwners
FROM
```

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

Berikut ini menunjukkan subquery dalam`SELECT` daftar yang memproyeksikan data bersarang, dan gabungan batin implisit.

```
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')
```

Berikut ini menunjukkan gabungan batin eksplisit.

```
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')
```

Berikut ini menunjukkan proyeksi bidang`id` metadata dokumen, menggunakan`BY` klausa.

```
SELECT
    r_id,
    r.VIN
FROM
    VehicleRegistration AS r BY r_id
WHERE
    r_id = 'documentId'
```

Berikut ini menggunakan`BY` klausa untuk bergabung`DriversLicense` dan`Person` tabel pada`id` bidang`PersonId` dan dokumen mereka masing-masing.

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
ON d.PersonId = pid
WHERE pid = 'documentId'
```

Berikut ini menggunakanTampilan berkomitmen untuk bergabung`DriversLicense` dan`Person` tabel pada`id` bidang`PersonId` dan dokumen mereka masing-masing.

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS cp
ON d.PersonId = cp.metadata.id
WHERE cp.metadata.id = 'documentId'
```

Berikut ini mengembalikan`PersonId` dan indeks (ordinal) jumlah setiap orang dalam`Owners.SecondaryOwners` daftar untuk dokumen dalam tabel`VehicleRegistration`.

```
SELECT s.PersonId, owner_idx
FROM VehicleRegistration AS r, @r.Owners.SecondaryOwners AS s AT owner_idx
WHERE r.VIN = 'KM8SRDHF6EU074761'
```

### Menjalankan pemrograman menggunakan driver

Untuk mempelajari cara menjalankan pernyataan ini secara terprogram menggunakan driver QLDB, lihat tutorial berikut dalam Memulai driver:

- Jawa:Tutorial Quick Start |Referensi buku masak
- .NET:Quick start tutorial |Referensi Referensi
- Pergi:Tutorial Quick start |Referensi buku masak
- Node.js:Tutorial Quick Start |Referensi buku masak
- Python:Tutorial Quick Start |Referensi buku masak

## PERINTAH UPDATE di Amazon QLDB

Di Amazon QLDB, gunakan`UPDATE` perintah untuk mengubah nilai dari satu atau beberapa elemen dalam dokumen. Jika elemen tidak ada, ini dimasukkan.

Anda juga dapat menggunakan perintah ini untuk secara eksplisit menyisipkan dan menghapus elemen tertentu dalam dokumen, mirip denganDARI (INSERT, HAPUS, atau SET) pernyataan.

> **Note**
>
> Untuk mempelajari cara mengontrol akses untuk menjalankan perintah PartiQL ini pada tabel tertentu, lihatMemulai dengan mode izin standar di Amazon QLDB.

Topik



## Sintaksis

### UPDATE-SET

Memperbarui satu atau lebih elemen dalam dokumen. Jika elemen tidak ada, ini dimasukkan. Ini semantik sama dengan pernyataan FROM-SET.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
SET element = data [, element = data, ... ]
[ WHERE condition ]
```

### UPDATE-INSERT

Menyisipkan elemen baru dalam dokumen yang ada. Untuk memasukkan dokumen tingkat atas baru ke dalam tabel, Anda harus menggunakanSISIPKAN.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
INSERT INTO element VALUE data [ AT key_name ]
[ WHERE condition ]
```

### UPDATE-HAPUS

Menghapus elemen yang ada dalam dokumen, atau menghapus seluruh dokumen tingkat atas. Yang terakhir ini semantik sama denganHAPUS sintaks tradisional.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
REMOVE element
[ WHERE condition ]
```

## Parameter

***nama meja***

Nama tabel pengguna yang berisi data yang akan diubah. Pernyataan DMLhanya didukung dalam tampilan pengguna default. Setiap pernyataan hanya dapat berjalan pada satu meja.

SEBAGAI ***table_alias***

(Opsional) Alias yang ditentukan pengguna yang berkisar di atas tabel yang akan diperbarui. ASKata kunci adalah opsional.

OLEH ***id_alias***

(Opsional) Alias yang ditentukan pengguna yang mengikat ke bidang`id` metadata setiap dokumen dalam kumpulan hasil. Alias harus dinyatakan dalam`UPDATE` klausa menggunakan`BY` kata kunci. Hal ini berguna ketika Anda ingin memfilter pada ID dokumen saat query tampilan pengguna default. Untuk informasi selengkapnya, lihat Menggunakan klausa BY untuk query ID dokumen.

***elemen***

Elemen dokumen yang akan dibuat atau diubah.

***data***

Nilai baru untuk elemen.

DI ***key_name***

Sebuah nama kunci yang akan ditambahkan dalam dokumen yang akan dimodifikasi. Anda harus menentukan yang sesuai`VALUE` bersama dengan nama kunci. Hal ini diperlukan untuk memasukkan nilai baru posisi`AT` tertentu dalam dokumen.

***Kondisi*** WHERE

Kriteria seleksi dokumen yang akan diubah.

> **Note**
>
> Jika Anda menghapus`WHERE` klausul, semua dokumen dalam tabel akan diubah.

## Nilai kembali

`documentId`- ID unik dari setiap dokumen yang Anda perbarui.

## Contoh

Perbarui bidang dalam dokumen. Jika bidang tidak ada, ini dimasukkan.

```
UPDATE Person AS p
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE p.GovId = '111-22-3333'
```

Filter pada bidang`id` metadata dokumen yang ditugaskan sistem.

```
UPDATE Person AS p BY pid
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE pid = 'documentId'
```

Timpa seluruh dokumen.

```
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'
```

Memodifikasi`PersonId` bidang elemen pertama dalam`Owners.SecondaryOwners` daftar dalam dokumen.

```
UPDATE VehicleRegistration AS r
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
WHERE r.VIN = '1N4AL11D75C109151'
```

Masukkan`{'Mileage':26500}` sebagai pasangan nama-nilai tingkat atas dalam dokumen dalam`Vehicle` tabel.

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

Tambahkan`{'PersonId':'abc123'}` sebagai pasangan nama-nilai di`Owners.SecondaryOwners` bidang dokumen dalam`VehicleRegistration` tabel. Perhatikan bahwa`Owners.SecondaryOwners` harus sudah ada dan harus menjadi tipe data daftar untuk pernyataan ini untuk menjadi valid. Jika tidak, kata kunci`AT` diperlukan dalam`INSERT INTO` klausa.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
WHERE r.VIN = '1N4AL11D75C109151'
```

Masukkan`{'PersonId':'abc123'}` sebagai elemen pertama dalam`Owners.SecondaryOwners` daftar yang ada dalam dokumen.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
WHERE r.VIN = '1N4AL11D75C109151'
```

Tambahkan beberapa pasangan nama-nilai ke`Owners.SecondaryOwners` daftar yang ada dalam dokumen.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
WHERE r.VIN = '1N4AL11D75C109151'
```

Hapus elemen yang ada dalam dokumen.

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

Hapus seluruh dokumen dari tabel.

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

Hapus elemen pertama dari`Owners.SecondaryOwners` daftar dalam dokumen dalam`VehicleRegistration` tabel.

```
UPDATE VehicleRegistration AS r
REMOVE r.Owners.SecondaryOwners[0]
WHERE r.VIN = '1N4AL11D75C109151'
```

### Menjalankan pemrograman menggunakan driver

Untuk mempelajari cara menjalankan pernyataan ini secara terprogram menggunakan driver QLDB, lihat tutorial berikut dalam Memulai driver:

- Jawa:Tutorial Quick Start |Referensi buku masak
- .NET:Quick start tutorial |Referensi Referensi
- Pergi:Tutorial Quick start |Referensi buku masak
- Node.js:Tutorial Quick Start |Referensi buku masak
- Python:Tutorial Quick Start |Referensi buku masak

## UNDROP perintah TABLE di Amazon QLDB

Di Amazon QLDB, gunakan`UNDROP TABLE` perintah untuk mengaktifkan kembali tabel yang sebelumnya Anda jatuhkan (yaitu, dinonaktifkan). Menonaktifkan atau mengaktifkan kembali tabel tidak berpengaruh pada dokumen atau indeksnya.

> **Note**
>
> Untuk mempelajari cara mengontrol akses untuk menjalankan perintah PartiQL ini pada tabel tertentu, lihatMemulai dengan mode izin standar di Amazon QLDB.

Topik

- Sintaks
- Parameter
- Nilai kembali
- Contoh

### Sintaks

```
UNDROP TABLE "tableId"
```

### Parameter

**“*TableID*”**

ID unik dari tabel untuk mengaktifkan kembali, dilambangkan dengan tanda kutip ganda.

Tabel harus telah sebelumnya dijatuhkan, yang berarti bahwa itu ada dalam tabel katalog sistem`information_schema.user_tables` dan memiliki status`INACTIVE`. Juga tidak boleh ada tabel aktif dan ada dengan nama yang sama.

### Nilai kembali

`tableId`- ID unik dari tabel yang Anda aktifkan kembali.

### Contoh

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

## Fungsi PartiQL di Amazon QLDB

PartiQL di Amazon QLDB mendukung varian bawaan berikut dari fungsi standar SQL.

> **Note**
>
> Fungsi SQL yang tidak termasuk dalam daftar ini saat ini tidak didukung di QLDB. Referensi fungsi ini didasarkan pada dokumentasi PartiQL Built-in Fungsi.

#### Jenis tidak diketahui (null dan hilang) propagasi

Kecuali dinyatakan lain, semua fungsi ini menyebarkan nilai argumen null dan hilang. Propagasi`NULL` atau`MISSING` didefinisikan sebagai kembali`NULL` jika argumen fungsi adalah salah satu`NULL` atau`MISSING`. Berikut ini adalah contoh propagasi ini.

```
CHAR_LENGTH(null)    -- null
```

```
CHAR_LENGTH(missing) -- null (also returns null)
```

## Fungsi agregat

- AVG
- COUNT
- MAKSIMAL
- MINIMAL
- UKURAN
- JUMLAH

## Fungsi kondisional

- BERSATU
- ADA
- NULLIF

## Fungsi tanggal dan waktu

- DATE_ADD
- DATE_DIFF
- EKSTRAK
- UTCNOW

## Fungsi skalar

- TXID

## Fungsi string

- CHAR_LENGTH
- CHARACTER_LENGTH
- LEBIH RENDAH

- SUBSTRING
- MEMANGKAS
- ATAS

## Fungsi pemformatan tipe data

- PEMERAN
- TO_STRING
- TO_TIMESTAMP

## Fungsi AVG di Amazon QLDB

Di Amazon QLDB, gunakan`AVG` fungsi untuk mengembalikan rata-rata (mean aritmatika) dari nilai ekspresi masukan. Fungsi ini bekerja dengan nilai-nilai numerik dan mengabaikan nilai nol atau hilang.

### Sintaksis

```
AVG ( expression )
```

### Pendapat

*ekspresi*

Nama bidang atau ekspresi data numerik yang menjalankan fungsi.

### Jenis Data

Jenis argumen yang didukung:

- `int`
- `decimal`
- `float`

Jenis kembali:`decimal`

### Contoh

```
SELECT AVG(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 147.19
SELECT AVG(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 2.
```

### fungsi terkait

- COUNT
- MAKSIMAL
- MINIMAL
- UKURAN
- JUMLAH

## Fungsi CAST di Amazon QLDB

Di Amazon QLDB, gunakan`CAST` fungsi untuk mengevaluasi ekspresi tertentu ke nilai dan mengonversi nilai ke tipe data target yang ditentukan. Jika konversi tidak dapat dilakukan, fungsi mengembalikan kesalahan.

### Sintaksis

```
CAST ( expression AS type )
```

### Pendapat

*`ekspresi`*

Nama bidang atau ekspresi yang mengevaluasi ke nilai yang mengubah fungsi. Mengkonversi nilai null mengembalikan nulls. Parameter ini dapat berupa salah satu yang didukungJenis Data.

*`jenis`*

Nama tipe data target untuk konversi. Parameter ini dapat menjadi salah satu yang didukungJenis Data.

### Jenis pengembalian

Tipe data yang ditentukan oleh *`jenis`* argumen.

## Contoh

Contoh berikut menunjukkan propagasi jenis yang tidak diketahui (`NULL`or`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)
```

Nilai apa pun yang bukan tipe yang tidak diketahui tidak dapat dilemparkan ke`NULL` atau`MISSING`.

```
CAST(true AS null)    -- error
CAST(true AS missing) -- error
CAST(1    AS null)    -- error
CAST(1    AS missing) -- error
```

Contoh berikut menunjukkan 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'`)
```

Contoh berikut menunjukkan 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
```

Contoh berikut menunjukkan 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
```

Contoh berikut menunjukkan 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
```

Contoh berikut menunjukkan 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
```

Contoh berikut menunjukkan casting`AS 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'
```

Contoh berikut menunjukkan 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"
```

Contoh berikut menunjukkan castingAS `struct`.

```
CAST(`{ a: 1 }` AS struct) -- {a:1}
CAST(true       AS struct) -- err
```

Contoh berikut menunjukkan 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
```

Contoh berikut adalah pernyataan runnable yang mencakup beberapa contoh sebelumnya.

```
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"
```

## fungsi terkait

- TO_STRING
- TO_TIMESTAMP

# Fungsi CHAR_LENGTH di Amazon QLDB

Di Amazon QLDB, gunakanCHAR_LENGTH fungsi untuk mengembalikan jumlah karakter dalam string yang ditentukan, di mana karakter didefinisikan sebagai titik kode unicode tunggal.

## Sintaksis

```
CHAR_LENGTH ( string )
```

CHAR_LENGTHadalah sinonim dariFungsi CHARACTER_LENGTH di Amazon QLDB.

## Pendapat

*tali*

Nama bidang atau ekspresi tipe datastring yang fungsi mengevaluasi.

## Jenis pengembalian

`int`

## Contoh

```
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)
```

## fungsi terkait

- LEBIH RENDAH
- SUBSTRING
- MEMANGKAS
- ATAS

# Fungsi CHARACTER_LENGTH di Amazon QLDB

Sinonim dariCHAR_LENGTH fungsi.

Lihat Fungsi CHAR_LENGTH di Amazon QLDB.

## Fungsi COALESCE di Amazon QLDB

Di Amazon QLDB, diberi daftar satu atau lebih argumen, gunakan`COALESCE` fungsi untuk mengevaluasi argumen secara berurutan dari kiri ke kanan dan mengembalikan nilai pertama yang bukan tipe (`NULL`atau`MISSING`) yang tidak diketahui. Jika semua jenis argumen tidak diketahui, hasilnya adalah`NULL`.

`COALESCE`Fungsi tidak merambat`NULL` dan`MISSING`.

### Sintaksis

```
COALESCE ( expression [, expression, ... ] )
```

### Pendapat

*`ekspresi`*

Daftar satu atau lebih nama bidang atau ekspresi bahwa fungsi mengevaluasi. Setiap argumen dapat berupa salah satu yang didukungJenis Data.

### Jenis pengembalian

Setiap tipe data yang didukung. Jenis kembali adalah baik`NULL` atau sama dengan jenis ekspresi pertama yang mengevaluasi ke nilai non-null dan non-hilang.

### Contoh

```
SELECT COALESCE(1, null) FROM << 0 >>          -- 1
SELECT COALESCE(null, null, 1) FROM << 0 >>    -- 1
SELECT COALESCE(null, 'string') FROM << 0 >>  -- "string"
```

### fungsi terkait

- ADA
- NULLIF

# Fungsi COUNT di Amazon QLDB

Di Amazon QLDB, gunakan`COUNT` fungsi untuk mengembalikan jumlah dokumen yang ditentukan oleh ekspresi yang diberikan. Fungsi ini memiliki dua variasi:

- `COUNT(*)`- Menghitung semua dokumen dalam tabel target apakah atau tidak mereka termasuk nilai nol atau hilang.
- `COUNT(expression)`- Menghitung jumlah dokumen dengan nilai non-null dalam bidang atau ekspresi tertentu yang ada.

> **Warning**
>
> `COUNT`Fungsi ini tidak dioptimalkan, jadi kami tidak menyarankan menggunakannya tanpa pencarian terindeks. Ketika Anda menjalankan query di QLDB tanpa pencarian diindeks, memanggil scan tabel penuh. Hal ini dapat menyebabkan masalah kinerja pada tabel besar, termasuk konflik konkurensi dan batas waktu transaksi.
> Untuk menghindari pemindaian tabel, Anda harus menjalankan pernyataan dengan klausa`WHERE` predikat menggunakan operator kesetaraan (`=`or`IN`) pada bidang yang diindeks atau ID dokumen. Untuk informasi selengkapnya, lihat Mengoptimalkan kinerja kueri.

## Sintaksis

```
COUNT ( * | expression )
```

## Pendapat

*ekspresi*

Nama bidang atau ekspresi bidang yang menjalankan fungsi. Parameter ini dapat berupa salah satu yang didukungJenis Data.

## Jenis pengembalian

`int`

### Contoh

```
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
```

### fungsi terkait

- AVG
- MAKSIMAL
- MINIMAL
- UKURAN
- JUMLAH

## Fungsi DATE_ADD di Amazon QLDB

Di Amazon QLDB, gunakan`DATE_ADD` fungsi untuk menambah nilai stempel waktu tertentu dengan interval tertentu.

### Sintaksis

```
DATE_ADD( datetimepart, interval, timestamp )
```

### Pendapat

*`datetimepart`*

Tanggal atau waktu yang menjalankan fungsi. Parameter ini dapat menjadi salah satu dari yang berikut:

- `year`
- `month`
- `day`
- `hour`
- `minute`

- `second`

*`interval`*

Integer yang menentukan interval untuk menambah *`stempel waktu`* yang diberikan. Sebuah integer negatif mengurangi interval.

*`stempel waktu`*

Nama field atau ekspresi tipe data`timestamp` yang fungsi bertahap.

Nilai literal stempel waktu Ion dapat dilambangkan dengan backticks (`` `...` ``). Untuk memformat detail dan contoh nilai stempel waktu, lihat Cap Waktu dalam dokumen spesifikasi Amazon Ion.

## Jenis pengembalian

`timestamp`

## Contoh

```
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
```

## fungsi terkait

- DATE_DIFF
- EKSTRAK
- TO_STRING
- TO_TIMESTAMP

- UTCNOW

# Fungsi DATE_DIFF di Amazon QLDB

Di Amazon QLDB, gunakan`DATE_DIFF` fungsi untuk mengembalikan perbedaan antara bagian tanggal yang ditentukan dari dua stempel waktu yang diberikan.

## Sintaksis

```
DATE_DIFF( datetimepart, timestamp1, timestamp2 )
```

## Pendapat

*`datetimepart`*

Tanggal atau waktu yang menjalankan fungsi. Parameter ini dapat menjadi salah satu dari yang berikut:

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`

*`timestamp1`*, *`timestamp2`*

Dua nama field atau ekspresi dari tipe data`timestamp` yang fungsi membandingkan. Jika *`timestamp2`* lebih lambat dari *`timestamp1`*, hasilnya positif. Jika *`timestamp2`* lebih awal dari *`timestamp1`*, hasilnya negatif.

Nilai literal stempel waktu Ion dapat dilambangkan dengan backticks (`` `...` ``). Untuk memformat detail dan contoh nilai stempel waktu, lihat Cap Waktu dalam dokumen spesifikasi Amazon Ion.

## Jenis pengembalian

`int`

### Contoh

```
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
```

### fungsi terkait

- DATE_ADD
- EKSTRAK
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

## Fungsi EXISTS di Amazon QLDB

Di Amazon QLDB, dengan nilai PartiQL, gunakan`EXISTS` fungsi untuk mengembalikan`TRUE` jika nilai adalah pengumpulan non-kosong. Jika tidak, fungsi ini kembali`FALSE`. Jika input ke`EXISTS` bukan wadah, hasilnya adalah`FALSE`.

`EXISTS`Fungsi tidak merambat`NULL` dan`MISSING`.

### Sintaksis

```
EXISTS ( value )
```

### Pendapat

*nilai*

Nama bidang atau ekspresi bidang yang menjalankan fungsi. Parameter ini dapat berupa salah satu yang didukungJenis Data.

### Jenis pengembalian

`bool`

### Contoh

```
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
```

### fungsi terkait

- BERSATU
- NULLIF

## Fungsi EXTRACT di Amazon QLDB

Di Amazon QLDB, gunakanEXTRACT fungsi untuk mengembalikan nilai integer dari tanggal atau bagian waktu yang ditentukan dari stempel waktu tertentu.

## Sintaksis

```
EXTRACT ( datetimepart FROM timestamp )
```

## Pendapat

*datetimepart*

Tanggal atau waktu bagian yang fungsi ekstrak. Parameter ini dapat menjadi salah satu dari yang berikut:

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`
- `timezone_hour`
- `timezone_minute`

*stempel waktu*

Nama bidang atau ekspresi tipe data`timestamp` yang fungsi ekstrak dari. Jika parameter ini adalah tipe yang tidak diketahui (`NULL`or`MISSING`), fungsi kembali`NULL`.

Nilai literal stempel waktu Ion dapat dilambangkan dengan backticks (`` `...` ``). Untuk memformat detail dan contoh nilai stempel waktu, lihat Cap Waktu dalam dokumen spesifikasi Amazon Ion.

## Jenis pengembalian

`int`

## Contoh

```
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
```

### fungsi terkait

- DATE_ADD
- DATE_DIFF
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

## Fungsi RENDAH di Amazon QLDB

Di Amazon QLDB, gunakan`LOWER` fungsi untuk mengonversi semua karakter huruf besar menjadi karakter huruf kecil dalam string tertentu.

### Sintaksis

```
LOWER ( string )
```

### Pendapat

*`tali`*

Nama field atau ekspresi tipe data`string` yang fungsi mengkonversi.

### Jenis pengembalian

`string`

### Contoh

```
SELECT LOWER('AbCdEfG!@#$') FROM << 0 >> -- 'abcdefg!@#$'
```

### fungsi terkait

- CHAR_LENGTH
- SUBSTRING
- MEMANGKAS
- ATAS

## Fungsi MAX di Amazon QLDB

Di Amazon QLDB, gunakan`MAX` fungsi untuk mengembalikan nilai maksimum dalam sekumpulan nilai numerik.

### Sintaksis

```
MAX ( expression )
```

### Pendapat

*`ekspresi`*

Nama bidang atau ekspresi data numerik yang menjalankan fungsi.

### Jenis Data

Jenis argumen yang didukung:

- `int`
- `decimal`
- `float`

Jenis pengembalian yang didukung:

- `int`
- `decimal`
- `float`

### Contoh

```
SELECT MAX(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 442.30
SELECT MAX(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 3
```

### fungsi terkait

- AVG
- COUNT
- MINIMAL
- UKURAN
- JUMLAH

## Fungsi MIN di Amazon QLDB

Di Amazon QLDB, gunakan`MIN` fungsi untuk mengembalikan nilai minimum dalam sekumpulan nilai numerik.

### Sintaksis

```
MIN ( expression )
```

### Pendapat

*`ekspresi`*

Nama bidang atau ekspresi data numerik yang menjalankan fungsi.

### Jenis Data

Jenis argumen yang didukung:

- `int`
- `decimal`
- `float`

Jenis pengembalian yang didukung:

- `int`
- `decimal`
- `float`

### Contoh

```
SELECT MIN(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 30.45
SELECT MIN(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 1
```

### fungsi terkait

- AVG
- COUNT
- MAKSIMAL
- UKURAN
- JUMLAH

## Fungsi NULLIF di Amazon QLDB

Di Amazon QLDB, dengan dua ekspresi, gunakan`NULLIF` fungsi tersebut untuk kembalikan`NULL` jika kedua ungkapan tersebut mengevaluasi nilai yang sama. Jika tidak, fungsi ini mengembalikan hasil evaluasi ekspresi pertama.

`NULLIF`Fungsi tidak merambat`NULL` dan`MISSING`.

### Sintaksis

```
NULLIF ( expression1, expression2 )
```

### Pendapat

*`expression1`*, *`expression2`*

Dua nama field atau ekspresi bahwa fungsi membandingkan. Parameter ini dapat berupa salah satu yang didukungJenis Data.

### Jenis pengembalian

Setiap tipe data yang didukung. Jenis kembali adalah baikNULL atau sama dengan jenis ekspresi pertama.

### Contoh

```
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
```

### fungsi terkait

- BERSATU
- ADA

## Fungsi UKURAN di Amazon QLDB

Di Amazon QLDB, gunakanSIZE fungsi untuk mengembalikan jumlah elemen dalam tipe data kontainer tertentu (daftar, struktur, atau tas).

### Sintaksis

```
SIZE ( container )
```

## Pendapat

*wadah*

Nama bidang kontainer atau ekspresi bidang yang menjalankan fungsi.

## Jenis Data

Jenis argumen yang didukung:

- Daftar
- struktur
- ransel

Jenis kembali:`int`

Jika input ke`SIZE` bukan wadah, fungsi melempar kesalahan.

## Contoh

```
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
```

## fungsi terkait

- AVG
- COUNT
- MAKSIMAL
- MINIMAL

- JUMLAH

# Fungsi SUBSTRING di Amazon QLDB

Di Amazon QLDB, gunakan`SUBSTRING` fungsi untuk mengembalikan substring dari string tertentu. Substring dimulai dari indeks awal yang ditentukan dan berakhir pada karakter terakhir dari string, atau pada panjang yang ditentukan.

## Sintaksis

```
SUBSTRING ( string, start-index [, length ] )
```

## Pendapat

*tali*

Nama bidang atau ekspresi tipe data`string` dari mana untuk mengekstrak substring.

*start-indeks*

Posisi awal dalam *string* dari mana untuk memulai ekstraksi. Jumlah ini bisa negatif.

Karakter pertama dari *string* memiliki indeks 1.

*panjang*

(Opsional) Jumlah karakter (titik kode) untuk mengekstrak dari *string*, mulai dari *start-index* dan berakhir pada (*start-index+length*) - 1. Dengan kata lain, panjang substring. Jumlah ini tidak bisa negatif.

Jika parameter ini tidak disediakan, fungsi berlangsung sampai akhir *string*.

## Jenis pengembalian

`string`

## Contoh

```
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"
```

### fungsi terkait

- CHAR_LENGTH
- LEBIH RENDAH
- MEMANGKAS
- ATAS

## Fungsi SUM di Amazon QLDB

Di Amazon QLDB, gunakan`SUM` fungsi untuk mengembalikan jumlah kolom input atau nilai ekspresi. Fungsi ini bekerja dengan nilai-nilai numerik dan mengabaikan nilai nol atau hilang.

### Sintaksis

```
SUM ( expression )
```

### Pendapat

*`ekspresi`*

Nama bidang atau ekspresi data numerik yang menjalankan fungsi.

### Jenis Data

Jenis argumen yang didukung:

- `int`
- `decimal`
- `float`

Jenis pengembalian yang didukung:

- `int`- Untuk argumen integer
- `decimal`- Untuk argumen desimal atau floating point

### Contoh

```
SELECT SUM(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 735.95
SELECT SUM(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 6
```

### fungsi terkait

- AVG
- COUNT
- MAKSIMAL
- MINIMAL
- UKURAN

## Fungsi TO_STRING di Amazon QLDB

Di Amazon QLDB, gunakan`TO_STRING` fungsi untuk mengembalikan representasi string dari stempel waktu tertentu dalam pola format yang ditentukan.

### Sintaksis

```
TO_STRING ( timestamp, 'format' )
```

### Pendapat

*`stempel waktu`*

Nama field atau ekspresi tipe data`timestamp` yang fungsi mengkonversi ke string.

Nilai literal stempel waktu Ion dapat dilambangkan dengan backticks (`` `...` ``). Untuk memformat detail dan contoh nilai stempel waktu, lihat Cap Waktu dalam dokumen spesifikasi Amazon Ion.

*format*

String literal yang menentukan pola format hasil, dalam hal bagian tanggal. Untuk format yang valid, lihatString format stempel waktu.

## Jenis pengembalian

`string`

## Contoh

```
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"
```

## fungsi terkait

- PEMERAN
- DATE_ADD
- DATE_DIFF
- EKSTRAK

- TO_TIMESTAMP
- UTCNOW

# Fungsi TO_TIMESTAMP di Amazon QLDB

Di Amazon QLDB, diberi string yang mewakili stempel waktu, gunakan`TO_TIMESTAMP` fungsi untuk mengonversi string ke tipe`timestamp` data. Ini adalah operasi terbalik`TO_STRING`.

## Sintaksis

```
TO_TIMESTAMP ( string [, 'format' ] )
```

## Pendapat

*`tali`*

Nama field atau ekspresi tipe data`string` yang fungsi mengkonversi ke stempel waktu.

*`format`*

(Opsional) String literal yang mendefinisikan pola format *`string`* input, dalam hal bagian tanggalnya. Untuk format yang valid, lihatString format stempel waktu.

Jika argumen ini dihilangkan, fungsi mengasumsikan bahwa *`string`* dalam format stempel waktu Ion standar. Ini adalah cara yang disarankan untuk mengurai stempel waktu Ion menggunakan fungsi ini.

Nol padding adalah opsional ketika menggunakan simbol format karakter tunggal (sepertiy,,`M`,d,`H`,h,`m`,,`s`) tetapi diperlukan untuk varian nol-empuk mereka (sepertiyyyy,,`MM`,dd,`HH`,hh,`mm`,`ss`).

Perlakuan khusus diberikan pada tahun dua digit (simbol formatyy). 1900 ditambahkan ke nilai yang lebih besar dari atau sama dengan 70, dan 2000 ditambahkan ke nilai kurang dari 70.

Nama bulan dan indikator AM atau PM tidak peka huruf besar.

## Jenis pengembalian

`timestamp`

### Contoh

```
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
```

### fungsi terkait

- PEMERAN
- DATE_ADD
- DATE_DIFF
- EKSTRAK
- TO_STRING
- UTCNOW

## Fungsi TRIM di Amazon QLDB

Di Amazon QLDB, gunakan`TRIM` fungsi untuk memangkas string tertentu dengan menghapus spasi kosong terkemuka dan tertinggal atau sekumpulan karakter tertentu.

### Sintaksis

```
TRIM ( [ LEADING | TRAILING | BOTH [ characters ] FROM ] string)
```

### Pendapat

#### TERKEMUKA

(Opsional) Menunjukkan bahwa spasi kosong atau karakter tertentu harus dihapus dari awal *`string`*. Jika tidak ditentukan, perilaku default adalah`BOTH`.

TERTINGGAL

(Opsional) Menunjukkan bahwa spasi kosong atau karakter tertentu harus dihapus dari akhir *`string`*. Jika tidak ditentukan, perilaku default adalah`BOTH`.

BOTH

(Opsional) Menunjukkan bahwa spasi kosong terkemuka dan tertinggal atau karakter tertentu harus dihapus dari awal dan akhir *`string`*.

*`karakter`*

(Opsional) Kumpulan karakter untuk dihapus, ditentukan sebagai`string`.

Jika parameter ini tidak disediakan, spasi kosong akan dihapus.

*`tali`*

Nama field atau ekspresi tipe data`string` yang fungsi trims.

## Jenis pengembalian

`string`

## Contoh

```
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         "
```

## fungsi terkait

- CHAR_LENGTH
- LEBIH RENDAH
- SUBSTRING

- ATAS

## Fungsi TXID di Amazon QLDB

Di Amazon QLDB, gunakan`TXID` fungsi untuk mengembalikan ID transaksi unik dari pernyataan saat ini yang sedang Anda jalankan. Ini adalah nilai yang ditetapkan ke bidang`txId` metadata dokumen ketika transaksi saat ini berkomitmen untuk jurnal.

### Sintaksis

```
TXID()
```

### Pendapat

Tidak ada

### Jenis pengembalian

`string`

### Contoh

```
SELECT TXID() FROM << 0 >>  -- "L7S9iJqcn9W2M4qOEn27ay"
SELECT TXID() FROM Person WHERE GovId = 'LEWISR261LL'  -- "BKeMb48PNyvHWJGZHkaodG"
```

## Fungsi UPPER di Amazon QLDB

Di Amazon QLDB, gunakanUPPER fungsi untuk mengonversi semua karakter huruf kecil menjadi karakter huruf besar dalam string tertentu.

### Sintaksis

```
UPPER ( string )
```

### Pendapat

*`tali`*

Nama field atau ekspresi tipe data`string` yang fungsi mengkonversi.

### Jenis pengembalian

`string`

### Contoh

```
SELECT UPPER('AbCdEfG!@#$') FROM << 0 >> -- 'ABCDEFG!@#$'
```

### fungsi terkait

- CHAR_LENGTH
- LEBIH RENDAH
- SUBSTRING
- MEMANGKAS

## Fungsi UTCNOW di Amazon QLDB

Di Amazon QLDB, gunakan`UTCNOW` fungsi untuk mengembalikan waktu saat ini di Coordinated Universal Time (UTC) sebagai `atimestamp`.

### Sintaksis

```
UTCNOW()
```

Fungsi ini mengharuskan Anda menentukan`FROM` klausa dalam`SELECT` query.

### Pendapat

Tidak ada

### Jenis pengembalian

`timestamp`

### Contoh

```
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'
```

### fungsi terkait

- DATE_ADD
- DATE_DIFF
- EKSTRAK
- TO_STRING
- TO_TIMESTAMP

## String format stempel waktu

Bagian ini memberikan informasi referensi untuk string format stempel waktu.

Format stempel waktu string berlaku untuk`TO_TIMESTAMP` fungsi`TO_STRING` dan. String ini dapat berisi pemisah bagian tanggal (seperti `'-'`,`/` '', atau`:` '') dan simbol format berikut.

| Format | Contoh | Deskripsi |
|---|---|---|
| yy | 70 | Tahun dalam dua digit |
| y | 1970 | Tahun dalam empat digit |
| yyyy | 1970 | Tahun dalam empat digit yang ditambahkan nol |
| M | 1 | integer bulan |
| MM | 01 | integer bulan yang ditambahk an nol |
| MMM | Jan | Nama bulan disingkat |
| MMMM | Januari | Nama bulan penuh |
| d | 2 | Hari dalam sebulan (1—31) |

| Format | Contoh | Deskripsi |
| --- | --- | --- |
| dd | 02 | Hari yang ditambahkan nol dalam sebulan (01—31) |
| a | Pagi atau Sore | Indikator Meridian (untuk jam 12 jam) |
| h | 3 | Jam (jam 12 jam, 1—12) |
| hh | 03 | Jam tanpa empuk (jam 12 jam, 01—12) |
| H | 3 | Jam (jam 24 jam, 0—23) |
| HH | 03 | Jam tanpa empuk (jam 24 jam, 00—23) |
| m | 4 | Menit (0—59) |
| mm | 04 | Menit tanpa empuk (00—59) |
| s | 5 | Detik (0—59) |
| ss | 05 | Detik berlapis nol (00—59) |
| S | 0 | Fraksi detik (presisi: 0,1, rentang: 0,0,0,9) |
| SS | 06 | Fraksi detik (presisi: 0,01, rentang: 0,0,0,99) |
| SSS | 060 | Fraksi detik (presisi: 0,001, rentang: 0,0,0,999) |
| X | +07 atau Z | Offset dari UTC dalam jam, atau “Z” jika offset adalah 0 |

| Format | Contoh | Deskripsi |
| --- | --- | --- |
| XX | +0700 atau Z | Offset dari UTC dalam jam dan menit, atau “Z” jika offset adalah 0 |
| XXX | + 07:00 atau Z | Offset dari UTC dalam jam dan menit, atau “Z” jika offset adalah 0 |
| x | +07 | Offset dari UTC dalam jam |
| xx | +0700 | Offset dari UTC dalam jam dan menit |
| xxx | + 07:00 | Offset dari UTC dalam jam dan menit |

# Prosedur tersimpan PartiQL di Amazon QLDB

Di Amazon QLDB, Anda dapat menggunakan`EXEC` perintah untuk menjalankan prosedur tersimpan PartiQL dalam sintaks berikut.

```
EXEC stored_procedure_name argument [, ... ]
```

QLDB mendukung sistem berikut disimpan prosedur hanya:

Topik

- Prosedur tersimpan REDACT_REVISION di Amazon QLDB

## Prosedur tersimpan REDACT_REVISION di Amazon QLDB

> **Note**
>
> Buku besar apa pun yang dibuat sebelum 22 Juli 2021 saat ini tidak memenuhi syarat untuk redaksi. Anda dapat melihat waktu pembuatan buku besar Anda di konsol Amazon QLDB.

Di Amazon QLDB, gunakan prosedur yang`REDACT_REVISION` disimpan untuk menghapus revisi dokumen individual yang tidak aktif secara permanen di penyimpanan terindeks dan penyimpanan jurnal. Prosedur yang disimpan ini menghapus semua data pengguna dalam revisi yang ditentukan. Namun, ia meninggalkan urutan jurnal dan metadata dokumen, termasuk ID dokumen dan hash, tidak berubah. Operasi ini tidak dapat diubah.

Revisi dokumen yang ditentukan harus merupakan revisi tidak aktif dalam sejarah. Revisi aktif terbaru dari dokumen tidak memenuhi syarat untuk redaksi.

Setelah Anda mengirimkan permintaan redaksi dengan menjalankan prosedur tersimpan ini, QLDB memproses redaksi data secara asinkron. Setelah redaksi selesai, data pengguna dari revisi yang ditentukan (diwakili oleh`data` struktur) digantikan oleh`dataHash` bidang baru. Nilai bidang ini adalah hash Amazon Ion dari`data` struktur yang dihapus. Akibatnya, buku besar mempertahankan integritas data secara keseluruhan dan tetap dapat diverifikasi secara kriptografis melalui operasi API verifikasi yang ada.

Untuk contoh operasi redaksi dengan data sampel, lihatContoh redaksi diMenyunting revisi dokumen.

> **Note**
>
> Untuk mempelajari cara mengontrol akses untuk menjalankan perintah PartiQL ini pada tabel tertentu, lihatMemulai dengan mode izin standar di Amazon QLDB.

Topik

- Pertimbangan dan batasan redaksi
- Sintaksis
- Pendapat
- Nilai kembali
- Contoh

## Pertimbangan dan batasan redaksi

Sebelum memulai redaksi data di Amazon QLDB, pastikan Anda meninjau pertimbangan dan batasan berikut:

- Prosedur yang`REDACT_REVISION` disimpan menargetkan data pengguna Anda dalam revisi dokumen individu yang tidak aktif. Untuk menyunting beberapa revisi, Anda harus menjalankan

prosedur yang disimpan satu kali untuk setiap revisi. Anda dapat menyunting satu revisi per transaksi.

- Untuk menyunting bidang tertentu dalam revisi dokumen, Anda harus menggunakan pernyataan bahasa manipulasi data (DHTML) terpisah untuk memodifikasi revisi terlebih dahulu. Untuk informasi selengkapnya, lihat Menyunting bidang tertentu dalam revisi.
- Setelah QLDB menerima permintaan redaksi, Anda tidak dapat membatalkan atau mengubah permintaan. Untuk mengonfirmasi apakah redaksi selesai, Anda dapat memeriksa apakah`data` struktur revisi telah diganti dengan`dataHash` bidang. Untuk mempelajari selengkapnya, lihat Memeriksa apakah redaksi selesai.
- Redaksi tidak berdampak pada data QLDB yang direplikasi di luar layanan QLDB. Ini termasuk ekspor ke Amazon S3 dan streaming ke Amazon Kinesis Data Streams. Anda harus menggunakan metode retensi data lain untuk mengelola data apa pun yang disimpan di luar QLDB.
- Redaksi tidak berdampak pada nilai literal dalam pernyataan PartiQL yang dicatat dalam jurnal. Sebagai praktik terbaik, Anda harus menjalankan pernyataan secara teratur dengan menggunakan pengganti variabel dan bukan nilai literal. Sebuah placeholder ditulis dalam jurnal sebagai tanda tanya (?), bukan informasi sensitif yang mungkin memerlukan redaksi.

  Untuk mempelajari cara menjalankan pernyataan PartiQL secara terprogram menggunakan driver QLDB, lihat tutorial untuk setiap bahasa pemrograman yang didukung diMemulai dengan driver.

## Sintaksis

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

## Pendapat

*`blok-alamat`*

Lokasi blok jurnal revisi dokumen yang akan disunting. Alamat adalah struktur Amazon Ion yang memiliki dua bidang:`strandId` dan`sequenceNo`.

Ini adalah nilai literal Ion yang dilambangkan dengan backticks. Misalnya:

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

Untuk mempelajari cara menemukan alamat blok, lihatMelakukan Kueri Metadata Dokumen.

'*tabel-id*'

ID unik dari tabel yang revisi dokumennya ingin Anda edit, dilambangkan dengan tanda kutip tunggal.

Untuk mempelajari cara menemukan ID tabel, lihatMenanyakan katalog sistem.

'*dokumen-id*'

ID dokumen unik dari revisi yang akan disunting, dilambangkan dengan tanda kutip tunggal.

Untuk mempelajari cara menemukan ID dokumen, lihatMelakukan Kueri Metadata Dokumen.

## Nilai kembali

Struktur Amazon Ion yang merepresentasikan revisi dokumen yang akan diedit, dalam format berikut.

```
{
  blockAddress: {
    strandId: String,
    sequenceNo: Int
  },
  tableId: String,
  documentId: String,
  version: Int
}
```

Kembali bidang struktur

- `blockAddress`— Lokasi blok jurnal revisi yang akan disunting. Alamat terdiri dari dua bidang berikut.
  - `strandId`- ID unik untai jurnal yang berisi blok.
  - `sequenceNo`- Nomor indeks yang menentukan lokasi blok di dalam untai.
- `tableId`- ID unik dari tabel yang revisi Anda menyunting.
- `documentId`- ID dokumen unik dari revisi yang akan disunting.
- `version`- Nomor versi revisi dokumen yang akan disunting.

Berikut ini adalah contoh.

```
{
  blockAddress: {
    strandId: "CsRnx0RDoNK6ANEEePa1ov",
    sequenceNo: 134
  },
  tableId: "6GZumdHggklLdMGyQq9DNX",
  documentId: "IXlQPSbfyKMIIsygePeKrZ",
  version: 0
}
```

### Contoh

```
EXEC REDACT_REVISION `{strandId:"7z2P0AyQKWD8oFYmGNhi8D", sequenceNo:7}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

## Operator PartiQL di Amazon QLDB

PartiQL di Amazon QLDB mendukung operator standar SQL berikut.

> **Note**
>
> Setiap operator SQL yang tidak termasuk dalam daftar ini saat ini tidak didukung di QLDB.

### Operator aritmatika

| Operator | Deskripsi |
| --- | --- |
| + | Tambahkan |
| - | Kurangi |
| * | Kalikan |
| / | Membagi |
| % | Modulo |

## Operator perbandingan

| Operator | Deskripsi |
| --- | --- |
| = | Sama dengan |
| > | Lebih besar dari |
| < | Kurang dari |
| >= | Lebih besar dari atau sama dengan |
| <= | Kurang dari atau sama dengan |
| <> | Tidak sama dengan |

## Operator logika

| Operator | Deskripsi |
| --- | --- |
| `AND` | `TRUE` jika semua kondisi dipisahkan oleh `AND` adalah `TRUE` |
| `BETWEEN` | `TRUE` jika operand berada dalam rentang perbandingan |
| `IN` | `TRUE` jika operand sama dengan salah satu dari daftar ekspresi |
| `IS` | `TRUE`jika operan adalah tipe data yang diberikan, termasuk`NULL` atau`MISSING` |
| `LIKE` | `TRUE`jika operan cocok dengan pola |
| `NOT` | Membalikkan nilai ekspresi Boolean yang diberikan |

| Operator | Deskripsi |
| --- | --- |
| `OR` | `TRUE`jika salah satu kondisi yang dipisahkan oleh`OR` adalah`TRUE` |

## Operator String

| Operasi | Deskripsi |
| --- | --- |
| `\|\|` | Merangkai dua string di kedua sisi`\|\|` operator dan mengembalikan string bersambung. Jika salah satu atau kedua string`NULL`, hasil penggabungan adalah nol. |

# Kata kunci yang dipesan di Amazon QLDB

Berikut ini adalah daftar kata kunci milik PartiQL di Amazon QLDB. Anda dapat menggunakan kata kunci yang dipesan sebagai pengidentifikasi yang dikutip dengan tanda kutip ganda (misalnya,`"user"`). Untuk informasi tentang konvensi mengutip PartiQL di QLDB, lihatKueri Ion dengan PartiQL.

> **Important**
>
> Kata kunci dalam daftar ini semua dianggap dilindungi karena PartiQL kompatibel dengan SQL-92. Namun, QLDB hanya mendukung sebagian dari kata-kata yang dipesan ini. Untuk daftar kata kunci SQL yang didukung QLDB saat ini, lihat topik berikut:
>
> - fungsi PartiQL
> - operator PartiQL
> - Perintah 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
```

# Referensi format data Amazon Ion di Amazon QLDB

Amazon QLDB menggunakan model notasi data yang menyatukan Amazon Ion dengan subset jenis PartiQL. Bagian ini memberikan gambaran umum referensi tentang format data dokumen Ion, terpisah dari integrasinya dengan PartiQL.

Menanyakan Ion dengan PartiQL di Amazon QLDB

Untuk sintaks dan semantik kueri data Ion dengan PartiQL di QLDB, lihatKueri Ion dengan PartiQL di referensi Amazon QLDB PartiQL.

Untuk contoh kode yang query dan proses data Ion dalam buku besar QLDB, lihatContoh kode Amazon Ion danBekerja dengan Amazon Ion.

Topik



## Apa itu Amazon Ion?

Ion adalah open-source, kaya diketik, self-describing, format serialisasi data hirarkis yang awalnya dikembangkan secara internal di Amazon. Ini didasarkan pada model data abstrak yang memungkinkan Anda menyimpan data terstruktur dan tidak terstruktur. Ini adalah superset dari JSON, yang berarti bahwa dokumen JSON yang valid juga merupakan dokumen Ion yang valid. Panduan ini mengasumsikan pengetahuan dasar tentang JSON. Jika Anda belum terbiasa dengan JSON, lihat Memperkenalkan JSON untuk informasi selengkapnya.

Anda dapat mencatat dokumen Ion secara bergantian dalam bentuk teks yang dapat dibaca manusia atau formulir yang dikodekan biner. Seperti JSON, bentuk teks mudah dibaca dan ditulis, mendukung pembuatan prototipe cepat. Pengkodean biner lebih kompak dan efisien untuk bertahan, mengirimkan, dan mengurai. Prosesor Ion dapat mentranskode antara kedua format untuk mewakili kumpulan struktur data yang persis sama tanpa kehilangan data. Fitur ini memungkinkan aplikasi mengoptimalkan cara mereka memproses data untuk kasus penggunaan yang berbeda.

> **Note**
>
> Model data Ion secara ketat berbasis nilai dan tidak mendukung referensi. Dengan demikian, model data dapat mewakili hirarki data yang dapat bersarang ke kedalaman sewenang-wenang, tetapi tidak diarahkan grafik.

## Spesifikasi ion

Untuk daftar lengkap tipe data inti Ion dengan deskripsi lengkap dan detail pemformatan nilai, lihat dokumen spesifikasi Ion di GitHub situs Amazon.

Untuk merampingkan pengembangan aplikasi, Amazon Ion menyediakan pustaka klien yang memproses data Ion untuk Anda. Untuk contoh kode kasus penggunaan umum untuk memproses data Ion, lihat Buku Masakan Amazon Ion aktif GitHub.

## Kompatibel dengan JSON

Mirip dengan JSON, Anda menyusun dokumen Amazon Ion dengan sekumpulan tipe data primitif dan sekumpulan tipe kontainer yang ditentukan secara rekursif. Ion mencakup tipe data JSON tradisional berikut:

- `null`: Sebuah generik, untyped null (kosong) nilai. Selain itu, seperti yang dijelaskan di bagian berikut, Ion mendukung tipe null yang berbeda untuk setiap tipe primitif.
- `bool`: Nilai Boolean.
- `string`: Literal teks Unicode.
- `list`: Memerintahkan koleksi nilai heterogen.
- `struct`: Koleksi pasangan nama-nilai. Seperti JSON,`struct` memungkinkan beberapa nilai per nama, tetapi ini umumnya tidak dianjurkan.

## Ekstensi dari JSON

### Jenis Angka

Alih-alih`number` tipe JSON yang ambigu, Amazon Ion secara ketat mendefinisikan angka sebagai salah satu jenis berikut:

- `int`: Bilangan bulat yang ditandatangani dengan ukuran sewenang-wenang.
- `decimal`: Bilangan real yang dikodekan desimal dengan presisi sewenang-wenang.
- `float`: Nomor floating point yang dikodekan biner (IEEE 64-bit).

Saat mengurai dokumen, prosesor Ion menetapkan jenis nomor sebagai berikut:

- `int`: Angka tanpa titik eksponen atau desimal (misalnya,`100200`).
- `decimal`: Angka dengan titik desimal dan tidak ada eksponen (misalnya,`0.00001`,`200.0`).
- `float`: Angka dengan eksponen, seperti notasi ilmiah atau e-notasi (misalnya,`2e0`,`3.1e-4`).

### Tipe data baru

Amazon Ion menambahkan tipe data berikut:

- `timestamp`: Tanggal/waktu/zona waktu momen presisi sewenang-wenang.
- `symbol`: Atom simbolis Unicode (seperti pengidentifikasi).
- `blob`: Data biner pengkodean yang ditentukan pengguna.
- `clob`: Data teks pengkodean yang ditentukan pengguna.
- `sexp`: Koleksi nilai yang dipesan dengan semantik yang ditentukan aplikasi.

### Jenis Null

Selain tipe null generik yang ditentukan oleh JSON, Amazon Ion mendukung tipe null yang berbeda untuk setiap tipe primitif. Hal ini menunjukkan kurangnya nilai sambil mempertahankan tipe data yang ketat.

```
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
```

## Contoh teks 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
    ],
}
```

## Referensi API

- ion-pergi
- ion-java
- ion-js
- ion-python

## Contoh kode Amazon Ion di QLDB

Bagian ini memberikan contoh kode yang memproses data Amazon Ion dengan membaca dan menulis nilai dokumen dalam buku besar Amazon QLDB. Contoh kode menggunakan driver QLDB untuk menjalankan pernyataan PartiQL pada buku besar. Contoh-contoh ini adalah bagian dari aplikasi sampel diMemulai dengan Amazon QLDB menggunakan contoh tutorial aplikasi, dan merupakan open source di GitHub situsAWS Sampel.

Untuk contoh kode umum yang menunjukkan kasus penggunaan umum pemrosesan data Ion, lihat Buku Masakan Amazon Ion aktif GitHub.

### Menjalankan kode

Kode tutorial untuk setiap bahasa pemrograman melakukan langkah-langkah berikut:

1. Connect ke buku besar`vehicle-registration` sampel.
2. Buat tabel bernama`IonTypes`.
3. Masukkan dokumen ke dalam tabel dengan satu`Name` bidang.
4. Untuk setiap tipe data Ion yang didukung:

a. PerbaruiName bidang dokumen dengan nilai literal dari tipe data.

b. Kueri tabel untuk mendapatkan revisi dokumen terbaru.

c. Validasi bahwa nilaiName dipertahankan sifat tipe data aslinya dengan memeriksa bahwa itu cocok dengan jenis yang diharapkan.

5. JatuhkanIonTypes meja.

> **Note**
>
> Sebelum Anda menjalankan kode tutorial ini, Anda harus membuat buku besar bernamavehicle-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()
```

# Referensi API Amazon QLDB

Bab ini menjelaskan operasi API tingkat rendah untuk Amazon QLDB yang dapat diakses melalui HTTP,AWS Command Line Interface(AWS CLI), atauAWSSDK:

- Amazon QLDB- API manajemen sumber daya QLDB (juga dikenal sebagaiBidang kendali). API ini hanya digunakan untuk mengelola sumber daya buku besar dan untuk operasi data non-transaksional. Anda dapat menggunakan operasi ini untuk menciptakan, menghapus, menggambarkan, membuat daftar, membuat daftar, dan memperbarui buku besar. Anda juga dapat memverifikasi data jurnal secara kriptografis, dan mengekspor atau melakukan streaming blok jurnal.
- Pengsesi Amazon QLDB- API data transaksional QLDB. Anda dapat menggunakan API ini untuk menjalankan transaksi data pada buku besar dengan PartiQL.

> **Important**
>
> Alih-alih berinteraksi langsung denganSesi QLDBAPI, kami sarankan menggunakan driver QLDB atau shell QLDB untuk menjalankan transaksi data pada buku besar.
>
> - Jika Anda bekerja denganAWSSDK, gunakan driver QLDB. Pengemudi menyediakan lapisan abstraksi tingkat tinggi di atasSesi QLDBdata API dan mengelolaSendCommandoperasi untuk Anda. Untuk informasi dan daftar bahasa pemrograman yang didukung, lihatMemulai dengan driver.
> - Jika Anda bekerja denganAWS CLI, gunakan shell QLDB. Shell adalah antarmuka baris perintah yang menggunakan driver QLDB untuk berinteraksi dengan buku besar. Untuk informasi, lihat Menggunakan shell Amazon QLDB (hanya API data).

Topik

# Tindakan

Tindakan berikut didukung oleh Amazon QLDB:

- CancelJournalKinesisStream
- CreateLedger
- DeleteLedger
- DescribeJournalKinesisStream
- DescribeJournalS3Export
- DescribeLedger
- ExportJournalToS3
- GetBlock
- GetDigest
- GetRevision
- ListJournalKinesisStreamsForLedger
- ListJournalS3Exports
- ListJournalS3ExportsForLedger
- ListLedgers
- ListTagsForResource
- StreamJournalToKinesis
- TagResource
- UntagResource
- UpdateLedger
- UpdateLedgerPermissionsMode

Tindakan berikut didukung oleh Sesi Amazon QLDB:

- SendCommand

## Amazon QLDB

The following actions are supported by Amazon QLDB:

- CancelJournalKinesisStream
- CreateLedger
- DeleteLedger
- DescribeJournalKinesisStream
- DescribeJournalS3Export
- DescribeLedger
- ExportJournalToS3
- GetBlock
- GetDigest
- GetRevision
- ListJournalKinesisStreamsForLedger
- ListJournalS3Exports
- ListJournalS3ExportsForLedger
- ListLedgers
- ListTagsForResource
- StreamJournalToKinesis
- TagResource
- UntagResource
- UpdateLedger
- UpdateLedgerPermissionsMode

## CancelJournalKinesisStream

Layanan: Amazon QLDB

Mengakhiri aliran jurnal QLDB Amazon yang diberikan. Sebelum streaming dapat dibatalkan, statusnya saat ini harus`ACTIVE`.

Anda tidak dapat memulai ulang aliran setelah Anda membatalkannya. Sumber daya aliran QLDB yang dibatalkan tunduk pada periode retensi 7 hari, sehingga akan dihapus secara otomatis setelah batas ini berakhir.

Minta Sintaks

```
DELETE /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama buku besar.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

streamId

UUID (diwakili dalam teks yang disandikan Base62) dari aliran jurnal QLDB yang akan dibatalkan.

Kendala Panjang: Panjang tetap 22.

Pola: `^[A-Za-z-0-9]+$`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

### Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "StreamId": "string"
}
```

### Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

**StreamId**

UUID (teks yang disandikan Base62) dari aliran jurnal QLDB yang dibatalkan.

Jenis: String

Kendala Panjang: Panjang tetap 22.

Pola: `^[A-Za-z-0-9]+$`

### Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

**InvalidParameterException**

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

**ResourceNotFoundException**

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

**ResourcePreconditionNotMetException**

Operasi gagal karena kondisi tidak terpenuhi sebelumnya.

Kode Status HTTP: 412

### Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## CreateLedger

Layanan: Amazon QLDB

Membuat buku besar baru di Wilayah Anda Akun AWS saat ini.

### Minta Sintaks

```
POST /ledgers HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "Tags": {
      "string" : "string"
   }
}
```

### Parameter Permintaan URI

Permintaan tidak menggunakan parameter URI apa pun.

### Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

#### DeletionProtection

Menentukan apakah buku besar dilindungi dari dihapus oleh setiap pengguna. Jika tidak ditentukan selama pembuatan buku besar, fitur ini diaktifkan (`true`) secara default.

Jika perlindungan penghapusan diaktifkan, Anda harus menonaktifkannya terlebih dahulu sebelum dapat menghapus buku besar. Anda dapat menonaktifkannya dengan memanggil `UpdateLedger` operasi untuk mengatur parameter ini`false`.

Tipe: Boolean

Wajib: Tidak

### KmsKey

Kunci in AWS Key Management Service (AWS KMS) yang akan digunakan untuk enkripsi data saat istirahat di buku besar. Untuk informasi selengkapnya, lihat Enkripsi saat istirahat di Panduan Pengembang QLDB Amazon.

Gunakan salah satu opsi berikut untuk menentukan parameter ini:

- `AWS_OWNED_KMS_KEY`: Gunakan AWS KMS kunci yang dimiliki dan dikelola oleh AWS atas nama Anda.
- Undefined: Secara default, gunakan kunci KMS yang AWS dimiliki.
- Kunci KMS terkelola pelanggan simetris yang valid: Gunakan kunci KMS enkripsi simetris yang ditentukan di akun yang Anda buat, miliki, dan kelola.

  Amazon QLDB tidak mendukung kunci asimetris. Untuk informasi selengkapnya, lihat Menggunakan kunci simetris dan asimetris di Panduan AWS Key Management Service Pengembang.

Untuk menentukan kunci KMS yang dikelola pelanggan, Anda dapat menggunakan ID kunci, Nama Sumber Daya Amazon (ARN), nama alias, atau alias ARN. Saat menggunakan nama alias, awali dengan. `"alias/"` Untuk menentukan kunci yang berbeda Akun AWS, Anda harus menggunakan kunci ARN atau alias ARN.

Sebagai contoh:

- ID Kunci: `1234abcd-12ab-34cd-56ef-1234567890ab`
- ARN kunci: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Nama alias: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

Untuk informasi selengkapnya, lihat Pengidentifikasi kunci (KeyId) di Panduan AWS Key Management Service Pengembang.

Jenis: String

Kendala Panjang: Panjang maksimum 1600.

Wajib: Tidak

### Name

Nama buku besar yang ingin Anda buat. Nama harus unik di antara semua buku besar di Wilayah Anda Akun AWS saat ini.

Batasan penamaan untuk nama buku besar didefinisikan dalam Kuota di Amazon QLDB dalam Panduan Developer Amazon QLDB.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

### PermissionsMode

Mode izin untuk ditetapkan ke buku besar yang ingin Anda buat. parameter ini dapat memiliki salah satu nilai berikut:

- `ALLOW_ALL`: Mode izin warisan yang memungkinkan kontrol akses dengan rincian tingkat API untuk buku besar.

  Mode ini memungkinkan pengguna yang memiliki izin API `SendCommand` untuk buku besar ini untuk menjalankan semua perintah PartiQL (maka, `ALLOW_ALL`) pada setiap tabel dalam buku besar yang ditentukan. Mode ini mengabaikan setiap kebijakan izin IAM tingkat tabel atau tingkat perintah yang Anda buat untuk buku besar.
- `STANDARD`: (Direkomendasikan) Mode perizinan yang memungkinkan kontrol akses dengan rincian yang lebih halus untuk buku besar, tabel, dan perintah PartiQL.

  Secara default, mode ini menyangkal semua permintaan pengguna untuk menjalankan perintah PartiQL pada setiap tabel dalam buku besar ini. Untuk mengizinkan perintah PartiQL untuk berjalan, Anda harus membuat kebijakan izin IAM untuk sumber daya tabel tertentu dan tindakan PartiQL, selain izin API `SendCommand` untuk buku besar. Untuk informasi, lihat Memulai dengan mode izin standar dalam Panduan Developer Amazon QLDB.

> **Note**
>
> Kami sangat merekomendasikan untuk menggunakan mode izin `STANDARD` untuk memaksimalkan keamanan data buku besar Anda.

Jenis: String

Nilai yang Valid: `ALLOW_ALL | STANDARD`

Wajib: Ya

## Tags

Pasangan kunci-nilai untuk ditambahkan sebagai tag ke buku besar yang ingin Anda buat. Kunci tag peka huruf besar dan kecil. Nilai tag peka huruf besar/kecil dan bisa null.

Tipe: Peta string ke string

Entri Peta: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Batasan Panjang Kunci: Panjang minimum 1. Panjang maksimum 128.

Batasan Panjang Nilai: Panjang minimum 0. Panjang maksimum 256.

Wajib: Tidak

### Sintaksis Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "CreationDateTime": number,
   "DeletionProtection": boolean,
   "KmsKeyArn": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "State": "string"
}
```

### Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

### Arn

Nama Sumber Daya Amazon (ARN) untuk buku besar.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

### CreationDateTime

Tanggal dan waktu, dalam format waktu epoch, saat buku besar dibuat. (Format waktu zaman adalah jumlah detik yang berlalu sejak 12:00:00 1 Januari 1970 UTC.)

Tipe: Timestamp

### DeletionProtection

Menentukan apakah buku besar dilindungi dari dihapus oleh setiap pengguna. Jika tidak ditentukan selama pembuatan buku besar, fitur ini diaktifkan (`true`) secara default.

Jika perlindungan penghapusan diaktifkan, Anda harus menonaktifkannya terlebih dahulu sebelum dapat menghapus buku besar. Anda dapat menonaktifkannya dengan memanggil `UpdateLedger` operasi untuk mengatur parameter ini`false`.

Jenis: Boolean

### KmsKeyArn

ARN dari kunci KMS yang dikelola pelanggan yang digunakan buku besar untuk enkripsi saat istirahat. Jika parameter ini tidak terdefinisi, buku besar menggunakan kunci KMS yang AWS dimiliki untuk enkripsi.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

### Name

Nama buku besar.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

Mode izin buku besar yang Anda buat.

Jenis: String

Nilai yang Valid: `ALLOW_ALL | STANDARD`

State

Status buku besar saat ini.

Jenis: String

Nilai yang Valid: `CREATING | ACTIVE | DELETING | DELETED`

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidParameterException

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

LimitExceededException

Anda telah mencapai batas jumlah maksimum sumber daya yang diizinkan.

Kode Status HTTP: 400

ResourceAlreadyExistsException

Sumber daya yang ditentukan sudah ada.

Kode Status HTTP: 409

ResourceInUseException

Sumber daya yang ditentukan tidak dapat dimodifikasi saat ini.

Kode Status HTTP: 409

## Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## DeleteLedger

Layanan: Amazon QLDB

Menghapus buku besar dan semua isinya. Tindakan ini tidak dapat diubah.

Jika perlindungan penghapusan diaktifkan, Anda harus menonaktifkannya terlebih dahulu sebelum dapat menghapus buku besar. Anda dapat menonaktifkannya dengan memanggil `UpdateLedger` operasi untuk mengatur parameter ini`false`.

Minta Sintaks

```
DELETE /ledgers/name HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama buku besar yang ingin Anda hapus.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidParameterException

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

ResourceInUseException

Sumber daya yang ditentukan tidak dapat dimodifikasi saat ini.

Kode Status HTTP: 409

ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

ResourcePreconditionNotMetException

Operasi gagal karena kondisi tidak terpenuhi sebelumnya.

Kode Status HTTP: 412

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## DescribeJournalKinesisStream

Layanan: Amazon QLDB

Mengembalikan informasi terperinci tentang aliran jurnal QLDB Amazon yang diberikan. Outputnya mencakup Nama Sumber Daya Amazon (ARN), nama aliran, status saat ini, waktu pembuatan, dan parameter permintaan pembuatan aliran asli.

Tindakan ini tidak mengembalikan aliran jurnal yang kedaluwarsa. Untuk informasi selengkapnya, lihat Kedaluwarsa untuk aliran terminal di Panduan Pengembang QLDB Amazon.

Minta Sintaks

```
GET /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama buku besar.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

streamId

UUID (diwakili dalam teks yang disandikan Base62) dari aliran jurnal QLDB untuk dijelaskan.

Kendala Panjang: Panjang tetap 22.

Pola: `^[A-Za-z-0-9]+$`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

### Sintaks Respons

```
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"
   }
}
```

### Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

#### Stream

Informasi tentang aliran jurnal QLDB dikembalikan oleh permintaan. `DescribeJournalS3Export`

Tipe: Objek JournalKinesisStreamDescription

### Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidParameterException

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

ResourcePreconditionNotMetException

Operasi gagal karena kondisi tidak terpenuhi sebelumnya.

Kode Status HTTP: 412

## Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## DescribeJournalS3Export

Layanan: Amazon QLDB

Mengembalikan informasi tentang pekerjaan ekspor jurnal, termasuk nama buku besar, ID ekspor, waktu pembuatan, status saat ini, dan parameter permintaan pembuatan ekspor asli.

Tindakan ini tidak mengembalikan pekerjaan ekspor yang kedaluwarsa. Untuk informasi selengkapnya, lihat Ekspor kedaluwarsa lowongan kerja di Panduan Pengembang QLDB Amazon.

Jika pekerjaan ekspor dengan yang diberikan `ExportId` tidak ada, maka lemparResourceNotFoundException.

Jika buku besar dengan yang diberikan Name tidak ada, maka lemparResourceNotFoundException.

Minta Sintaks

```
GET /ledgers/name/journal-s3-exports/exportId HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

exportId

UUID (diwakili dalam teks yang disandikan Base62) dari pekerjaan ekspor jurnal untuk dijelaskan.

Kendala Panjang: Panjang tetap 22.

Pola: `^[A-Za-z-0-9]+$`

Wajib: Ya

name

Nama buku besar.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
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"
   }
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

ExportDescription

Informasi tentang pekerjaan ekspor jurnal dikembalikan oleh `DescribeJournalS3Export` permintaan.

Tipe: Objek JournalS3ExportDescription

### Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

#### ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

### Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## DescribeLedger

Layanan: Amazon QLDB

Mengembalikan informasi tentang buku besar, termasuk statusnya, mode izin, enkripsi pada pengaturan istirahat, dan kapan dibuat.

### Minta Sintaks

```
GET /ledgers/name HTTP/1.1
```

### Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

#### name

Nama buku besar yang ingin Anda gambarkan.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

### Isi Permintaan

Permintaan tidak memiliki isi permintaan.

### Sintaks Respons

```
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"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

Arn

Nama Sumber Daya Amazon (ARN) untuk buku besar.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

CreationDateTime

Tanggal dan waktu, dalam format waktu epoch, saat buku besar dibuat. (Format waktu zaman adalah jumlah detik yang berlalu sejak 12:00:00 1 Januari 1970 UTC.)

Tipe: Timestamp

DeletionProtection

Menentukan apakah buku besar dilindungi dari dihapus oleh setiap pengguna. Jika tidak ditentukan selama pembuatan buku besar, fitur ini diaktifkan (`true`) secara default.

Jika perlindungan penghapusan diaktifkan, Anda harus menonaktifkannya terlebih dahulu sebelum dapat menghapus buku besar. Anda dapat menonaktifkannya dengan memanggil `UpdateLedger` operasi untuk mengatur parameter ini`false`.

Jenis: Boolean

EncryptionDescription

Informasi tentang enkripsi data saat istirahat di buku besar. Ini termasuk status saat ini, AWS KMS kunci, dan ketika kunci menjadi tidak dapat diakses (dalam kasus kesalahan). Jika parameter ini tidak terdefinisi, buku besar menggunakan kunci KMS yang AWS dimiliki untuk enkripsi.

Tipe: Objek LedgerEncryptionDescription

### Name

Nama buku besar.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

### PermissionsMode

Mode izin buku besar.

Jenis: String

Nilai yang Valid: `ALLOW_ALL | STANDARD`

### State

Status buku besar saat ini.

Jenis: String

Nilai yang Valid: `CREATING | ACTIVE | DELETING | DELETED`

## Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

### InvalidParameterException

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

### ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

## Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## ExportJournalToS3

Layanan: Amazon QLDB

Mengekspor konten jurnal dalam rentang tanggal dan waktu dari buku besar ke bucket Amazon Simple Storage Service (Amazon S3) tertentu. Pekerjaan ekspor jurnal dapat menulis objek data baik dalam teks atau representasi biner format Amazon Ion, atau dalam format teks JSON Lines.

Jika buku besar dengan yang diberikan `Name` tidak ada, maka lempar`ResourceNotFoundException`.

Jika buku besar dengan yang `Name` diberikan dalam `CREATING` status, maka lempar`ResourcePreconditionNotMetException`.

Anda dapat memulai hingga dua permintaan ekspor jurnal bersamaan untuk setiap buku besar. Di luar batas ini, permintaan ekspor jurnal melempar`LimitExceededException`.

Minta Sintaks

```
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"
   }
}
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama buku besar.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

ExclusiveEndTime

Tanggal dan waktu akhir eksklusif untuk berbagai konten jurnal untuk diekspor.

`ExclusiveEndTime` harus berada dalam format tanggal dan waktu `ISO 8601` dan dalam Waktu Terkoordinasi Universal (UTC). Misalnya: `2019-06-13T21:36:34Z`.

`ExclusiveEndTime`Harus kurang dari atau sama dengan tanggal dan waktu UTC saat ini.

Tipe: Timestamp

Wajib: Ya

InclusiveStartTime

Tanggal dan waktu mulai inklusif untuk berbagai konten jurnal untuk diekspor.

`InclusiveStartTime` harus berada dalam format tanggal dan waktu `ISO 8601` dan dalam Waktu Terkoordinasi Universal (UTC). Misalnya: `2019-06-13T21:36:34Z`.

`InclusiveStartTime`Harus sebelumnya`ExclusiveEndTime`.

Jika Anda memberikan `InclusiveStartTime` yang sebelum buku besar`CreationDateTime`, Amazon QLDB mendefaultkannya ke buku besar. `CreationDateTime`

Tipe: Timestamp

Wajib: Ya

OutputFormat

Format output dari data jurnal yang diekspor. Pekerjaan ekspor jurnal dapat menulis objek data baik dalam teks atau representasi biner format Amazon Ion, atau dalam format teks JSON Lines.

Default: `ION_TEXT`

Dalam format JSON Lines, setiap blok jurnal dalam objek data yang diekspor adalah objek JSON yang valid yang dibatasi oleh baris baru. Anda dapat menggunakan format ini untuk mengintegrasikan ekspor JSON secara langsung dengan alat analitik seperti Amazon Athena AWS Glue dan karena layanan ini dapat mengurai JSON yang dibatasi baris baru secara otomatis.

Jenis: String

Nilai yang Valid: `ION_BINARY | ION_TEXT | JSON`

Wajib: Tidak

RoleArn

Nama Sumber Daya Amazon (ARN) dari peran IAM yang memberikan izin QLDB untuk pekerjaan ekspor jurnal untuk melakukan hal berikut:

- Tulis objek ke dalam ember Amazon S3 Anda.
- (Opsional) Gunakan kunci terkelola pelanggan Anda di AWS Key Management Service (AWS KMS) untuk enkripsi sisi server dari data yang Anda ekspor.

Untuk meneruskan peran ke QLDB saat meminta ekspor jurnal, Anda harus memiliki izin untuk melakukan `iam:PassRole` tindakan pada sumber daya peran IAM. Ini diperlukan untuk semua permintaan ekspor jurnal.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

Wajib: Ya

S3ExportConfiguration

Pengaturan konfigurasi tujuan bucket Amazon S3 untuk permintaan ekspor Anda.

Tipe: Objek S3ExportConfiguration

Wajib: Ya

## Sintaksis Respons

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "ExportId": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

ExportId

UUID (diwakili dalam teks yang disandikan Base62) yang diberikan QLDB untuk setiap pekerjaan ekspor jurnal.

Untuk menjelaskan permintaan ekspor Anda dan memeriksa status pekerjaan, Anda dapat menggunakan `ExportId` untuk meneleponDescribeJournalS3Export.

Jenis: String

Kendala Panjang: Panjang tetap 22.

Pola: `^[A-Za-z-0-9]+$`

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

ResourcePreconditionNotMetException

Operasi gagal karena kondisi tidak terpenuhi sebelumnya.

Kode Status HTTP: 412

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## GetBlock

Layanan: Amazon QLDB

Mengembalikan objek blok pada alamat tertentu dalam jurnal. Juga mengembalikan bukti blok yang ditentukan untuk verifikasi jika `DigestTipAddress` disediakan.

Untuk informasi tentang konten data dalam satu blok, lihat Konten jurnal di Panduan Pengembang QLDB Amazon.

Jika buku besar yang ditentukan tidak ada atau dalam `DELETING` status, maka melempar`ResourceNotFoundException`.

Jika buku besar yang ditentukan dalam `CREATING` status, maka lempar`ResourcePreconditionNotMetException`.

Jika tidak ada blok dengan alamat yang ditentukan, maka melempar`InvalidParameterException`.

Minta Sintaks

```
POST /ledgers/name/block HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama buku besar.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

### Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

#### BlockAddress

Lokasi blok yang ingin Anda minta. Alamat adalah struktur Amazon Ion yang memiliki dua bidang: `strandId` dan`sequenceNo`.

Misalnya: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`.

Tipe: Objek ValueHolder

Wajib: Ya

#### DigestTipAddress

Lokasi blok terbaru yang dicakup oleh intisari untuk meminta bukti. Alamat adalah struktur Amazon Ion yang memiliki dua bidang: `strandId` dan`sequenceNo`.

Misalnya: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:49}`.

Tipe: Objek ValueHolder

Wajib: Tidak

### Sintaksis Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "Block": {
      "IonText": "string"
   },
   "Proof": {
      "IonText": "string"
   }
```

```
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

Block

Objek data blok dalam format Amazon Ion.

Tipe: Objek ValueHolder

Proof

Objek bukti dalam format Amazon Ion dikembalikan oleh `GetBlock` permintaan. Bukti berisi daftar nilai hash yang diperlukan untuk menghitung ulang intisari yang ditentukan menggunakan pohon Merkle, dimulai dengan blok yang ditentukan.

Tipe: Objek ValueHolder

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidParameterException

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

ResourcePreconditionNotMetException

Operasi gagal karena kondisi tidak terpenuhi sebelumnya.

Kode Status HTTP: 412

## Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## GetDigest

Layanan: Amazon QLDB

Mengembalikan intisari buku besar di blok komitmen terbaru dalam jurnal. Responsnya mencakup nilai hash 256-bit dan alamat blok.

### Minta Sintaks

```
POST /ledgers/name/digest HTTP/1.1
```

### Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama buku besar.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

### Isi Permintaan

Permintaan tidak memiliki isi permintaan.

### Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "Digest": blob,
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

### Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

Digest

Nilai hash 256-bit yang mewakili intisari yang dikembalikan oleh permintaan. `GetDigest`

Tipe: Objek data biner dienkode Base64

Kendala Panjang: Panjang tetap 32.

DigestTipAddress

Lokasi blok terbaru yang dicakup oleh intisari yang Anda minta. Alamat adalah struktur Amazon Ion yang memiliki dua bidang: `strandId` dan`sequenceNo`.

Tipe: Objek ValueHolder

## Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidParameterException

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

ResourcePreconditionNotMetException

Operasi gagal karena kondisi tidak terpenuhi sebelumnya.

Kode Status HTTP: 412

## Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## GetRevision

Layanan: Amazon QLDB

Mengembalikan objek data revisi untuk ID dokumen tertentu dan alamat blok. Juga mengembalikan bukti revisi yang ditentukan untuk verifikasi jika `DigestTipAddress` disediakan.

### Minta Sintaks

```
POST /ledgers/name/revision HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   },
   "DocumentId": "string"
}
```

### Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

#### name

Nama buku besar.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

### Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

#### BlockAddress

Lokasi blok revisi dokumen yang akan diverifikasi. Alamat adalah struktur Amazon Ion yang memiliki dua bidang: `strandId` dan`sequenceNo`.

Misalnya: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`.

Tipe: Objek ValueHolder

Wajib: Ya

DigestTipAddress

Lokasi blok terbaru yang dicakup oleh intisari untuk meminta bukti. Alamat adalah struktur Amazon Ion yang memiliki dua bidang: `strandId` dan`sequenceNo`.

Misalnya: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:49}`.

Tipe: Objek ValueHolder

Wajib: Tidak

DocumentId

UUID (diwakili dalam teks yang disandikan Base62) dari dokumen yang akan diverifikasi.

Jenis: String

Kendala Panjang: Panjang tetap 22.

Pola: `^[A-Za-z-0-9]+$`

Diperlukan: Ya

## Sintaksis Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "Proof": {
      "IonText": "string"
   },
   "Revision": {
      "IonText": "string"
   }
}
```

### Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

#### Proof

Objek bukti dalam format Amazon Ion dikembalikan oleh `GetRevision` permintaan. Bukti berisi daftar nilai hash yang diperlukan untuk menghitung ulang intisari yang ditentukan menggunakan pohon Merkle, dimulai dengan revisi dokumen yang ditentukan.

Tipe: Objek ValueHolder

#### Revision

Objek data revisi dokumen dalam format Amazon Ion.

Tipe: Objek ValueHolder

### Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

#### InvalidParameterException

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

#### ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

#### ResourcePreconditionNotMetException

Operasi gagal karena kondisi tidak terpenuhi sebelumnya.

Kode Status HTTP: 412

### Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## ListJournalKinesisStreamsForLedger

Layanan: Amazon QLDB

Mengembalikan semua aliran jurnal QLDB Amazon untuk buku besar tertentu.

Tindakan ini tidak mengembalikan aliran jurnal yang kedaluwarsa. Untuk informasi selengkapnya, lihat Kedaluwarsa untuk aliran terminal di Panduan Pengembang QLDB Amazon.

Tindakan ini mengembalikan maksimum `MaxResults` item. Ini diberi halaman sehingga Anda dapat mengambil semua item dengan menelepon `ListJournalKinesisStreamsForLedger` beberapa kali.

Minta Sintaks

```
GET /ledgers/name/journal-kinesis-streams?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama buku besar.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

MaxResults

Jumlah maksimum hasil yang akan dikembalikan dalam satu `ListJournalKinesisStreamsForLedger` permintaan. (Jumlah sebenarnya dari hasil yang dikembalikan mungkin lebih sedikit.)

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 100.

NextToken

Token pagination, yang menunjukkan bahwa Anda ingin mengambil halaman hasil berikutnya. Jika Anda menerima nilai untuk `NextToken` dalam respons dari

`ListJournalKinesisStreamsForLedger` panggilan sebelumnya, Anda harus menggunakan nilai itu sebagai input di sini.

Batasan Panjang: Panjang minimum 4. Panjang maksimum 1024.

Pola: `^[A-Za-z-0-9+/=]+$`

### Isi Permintaan

Permintaan tidak memiliki isi permintaan.

### Sintaks Respons

```
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"
      }
   ]
}
```

### Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

NextToken

- Jika `NextToken` kosong, halaman terakhir hasil telah diproses dan tidak ada lagi hasil yang akan diambil.
- Jika `NextToken` tidak kosong, lebih banyak hasil yang tersedia. Untuk mengambil halaman hasil berikutnya, gunakan nilai `NextToken` dalam `ListJournalKinesisStreamsForLedger` panggilan berikutnya.

Jenis: String

Batasan Panjang: Panjang minimum 4. Panjang maksimum 1024.

Pola: `^[A-Za-z-0-9+/=]+$`

Streams

Aliran jurnal QLDB yang saat ini dikaitkan dengan buku besar yang diberikan.

Tipe: Array objek JournalKinesisStreamDescription

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidParameterException

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

ResourcePreconditionNotMetException

Operasi gagal karena kondisi tidak terpenuhi sebelumnya.

Kode Status HTTP: 412

### Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## ListJournalS3Exports

Layanan: Amazon QLDB

Mengembalikan semua pekerjaan ekspor jurnal untuk semua buku besar yang terkait dengan saat ini Akun AWS dan Wilayah.

Tindakan ini mengembalikan `MaxResults` item maksimum, dan diberi paginasi sehingga Anda dapat mengambil semua item dengan menelepon `ListJournalS3Exports` beberapa kali.

Tindakan ini tidak mengembalikan pekerjaan ekspor yang kedaluwarsa. Untuk informasi selengkapnya, lihat Ekspor kedaluwarsa lowongan kerja di Panduan Pengembang QLDB Amazon.

Minta Sintaks

```
GET /journal-s3-exports?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

MaxResults

Jumlah maksimum hasil untuk dikembalikan dalam satu `ListJournalS3Exports` permintaan. (Jumlah sebenarnya dari hasil yang dikembalikan mungkin lebih sedikit.)

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 100.

NextToken

Token pagination, yang menunjukkan bahwa Anda ingin mengambil halaman hasil berikutnya. Jika Anda menerima nilai untuk `NextToken` dalam respons dari `ListJournalS3Exports` panggilan sebelumnya, maka Anda harus menggunakan nilai itu sebagai input di sini.

Batasan Panjang: Panjang minimum 4. Panjang maksimum 1024.

Pola: `^[A-Za-z-0-9+/=]+$`

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
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"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

JournalS3Exports

Pekerjaan ekspor jurnal untuk semua buku besar yang terkait dengan saat ini Akun AWS dan Wilayah.

Tipe: Array objek JournalS3ExportDescription

NextToken

- Jika `NextToken` kosong, maka halaman terakhir hasil telah diproses dan tidak ada lagi hasil yang akan diambil.
- Jika `NextToken` tidak kosong, maka ada lebih banyak hasil yang tersedia. Untuk mengambil halaman hasil berikutnya, gunakan nilai `NextToken` dalam `ListJournalS3Exports` panggilan berikutnya.

Jenis: String

Batasan Panjang: Panjang minimum 4. Panjang maksimum 1024.

Pola: `^[A-Za-z-0-9+/=]+$`

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## ListJournalS3ExportsForLedger

Layanan: Amazon QLDB

Mengembalikan semua pekerjaan ekspor jurnal untuk buku besar tertentu.

Tindakan ini mengembalikan `MaxResults` item maksimum, dan diberi paginasi sehingga Anda dapat mengambil semua item dengan menelepon `ListJournalS3ExportsForLedger` beberapa kali.

Tindakan ini tidak mengembalikan pekerjaan ekspor yang kedaluwarsa. Untuk informasi selengkapnya, lihat Ekspor kedaluwarsa lowongan kerja di Panduan Pengembang QLDB Amazon.

Minta Sintaks

```
GET /ledgers/name/journal-s3-exports?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

MaxResults

Jumlah maksimum hasil untuk dikembalikan dalam satu `ListJournalS3ExportsForLedger` permintaan. (Jumlah sebenarnya dari hasil yang dikembalikan mungkin lebih sedikit.)

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 100.

name

Nama buku besar.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

NextToken

Token pagination, yang menunjukkan bahwa Anda ingin mengambil halaman hasil berikutnya. Jika Anda menerima nilai untuk `NextToken` dalam respons dari `ListJournalS3ExportsForLedger` panggilan sebelumnya, maka Anda harus menggunakan nilai itu sebagai input di sini.

Batasan Panjang: Panjang minimum 4. Panjang maksimum 1024.

Pola: `^[A-Za-z-0-9+/=]+$`

### Isi Permintaan

Permintaan tidak memiliki isi permintaan.

### Sintaks Respons

```
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"
}
```

### Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

JournalS3Exports

Pekerjaan ekspor jurnal yang saat ini terkait dengan buku besar yang ditentukan.

Tipe: Array objek JournalS3ExportDescription

NextToken

- Jika `NextToken` kosong, maka halaman terakhir hasil telah diproses dan tidak ada lagi hasil yang akan diambil.
- Jika `NextToken` tidak kosong, maka ada lebih banyak hasil yang tersedia. Untuk mengambil halaman hasil berikutnya, gunakan nilai `NextToken` dalam `ListJournalS3ExportsForLedger` panggilan berikutnya.

Jenis: String

Batasan Panjang: Panjang minimum 4. Panjang maksimum 1024.

Pola: `^[A-Za-z-0-9+/=]+$`

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## ListLedgers

Layanan: Amazon QLDB

Mengembalikan semua buku besar yang terkait dengan saat ini Akun AWS dan Wilayah.

Tindakan ini mengembalikan maksimum `MaxResults` item dan diberi paginasi sehingga Anda dapat mengambil semua item dengan menelepon `ListLedgers` beberapa kali.

### Minta Sintaks

```
GET /ledgers?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

### Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

#### MaxResults

Jumlah maksimum hasil yang akan dikembalikan dalam satu `ListLedgers` permintaan. (Jumlah sebenarnya dari hasil yang dikembalikan mungkin lebih sedikit.)

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 100.

#### NextToken

Token pagination, yang menunjukkan bahwa Anda ingin mengambil halaman hasil berikutnya. Jika Anda menerima nilai untuk `NextToken` dalam respons dari `ListLedgers` panggilan sebelumnya, maka Anda harus menggunakan nilai itu sebagai input di sini.

Batasan Panjang: Panjang minimum 4. Panjang maksimum 1024.

Pola: `^[A-Za-z-0-9+/=]+$`

### Isi Permintaan

Permintaan tidak memiliki isi permintaan.

### Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "Ledgers": [
      {
         "CreationDateTime": number,
         "Name": "string",
         "State": "string"
      }
   ],
   "NextToken": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

Ledgers

Buku besar yang terkait dengan saat ini Akun AWS dan Wilayah.

Tipe: Array objek LedgerSummary

NextToken

Token pagination, yang menunjukkan apakah ada lebih banyak hasil yang tersedia:

- Jika `NextToken` kosong, maka halaman terakhir hasil telah diproses dan tidak ada lagi hasil yang akan diambil.
- Jika `NextToken` tidak kosong, maka ada lebih banyak hasil yang tersedia. Untuk mengambil halaman hasil berikutnya, gunakan nilai `NextToken` dalam `ListLedgers` panggilan berikutnya.

Jenis: String

Batasan Panjang: Panjang minimum 4. Panjang maksimum 1024.

Pola: `^[A-Za-z-0-9+/=]+$`

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

### Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## ListTagsForResource

Layanan: Amazon QLDB

Mengembalikan semua tag untuk sumber daya QLDB Amazon tertentu.

### Minta Sintaks

```
GET /tags/resourceArn HTTP/1.1
```

### Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

resourceArn

Nama Sumber Daya Amazon (ARN) untuk mencantumkan tag. Sebagai contoh:

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

Wajib: Ya

### Isi Permintaan

Permintaan tidak memiliki isi permintaan.

### Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

### Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

Tags

Tag yang saat ini dikaitkan dengan sumber daya QLDB Amazon yang ditentukan.

Tipe: Peta string ke string

Entri Peta: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Batasan Panjang Kunci: Panjang minimum 1. Panjang maksimum 128.

Batasan Panjang Nilai: Panjang minimum 0. Panjang maksimum 256.

## Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidParameterException

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

## Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript

- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## StreamJournalToKinesis

Layanan: Amazon QLDB

Membuat aliran jurnal untuk buku besar Amazon QLDB tertentu. Pengaliran menangkap setiap revisi dokumen yang dilakukan ke jurnal buku besar dan mengirimkan data ke sumber daya Amazon Kinesis Data Streams yang ditentukan.

### Minta Sintaks

```
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"
   }
}
```

### Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

#### name

Nama buku besar.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

### Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

ExclusiveEndTime

Tanggal dan waktu eksklusif yang menentukan kapan pengaliran berakhir. Jika Anda tidak menentukan parameter ini, pengaliran berjalan tanpa batas waktu hingga Anda membatalkannya.

`ExclusiveEndTime` harus berada dalam format tanggal dan waktu `ISO 8601` dan dalam Waktu Terkoordinasi Universal (UTC). Misalnya: `2019-06-13T21:36:34Z`.

Tipe: Timestamp

Wajib: Tidak

InclusiveStartTime

Tanggal dan waktu mulai inklusif untuk memulai data jurnal streaming. Parameter ini harus berada dalam format tanggal dan waktu `ISO 8601` dan dalam Waktu Terkoordinasi Universal (UTC). Misalnya: `2019-06-13T21:36:34Z`.

`InclusiveStartTime` tidak bisa untuk masa depan dan harus sebelum `ExclusiveEndTime`.

Jika Anda menyediakan `InclusiveStartTime` yang ada sebelum `CreationDateTime` buku besar, QLDB secara efektif men-default ke `CreationDateTime` buku besar.

Tipe: Timestamp

Wajib: Ya

KinesisConfiguration

Pengaturan konfigurasi tujuan Kinesis Data Streams untuk permintaan pengaliran Anda.

Tipe: Objek KinesisConfiguration

Wajib: Ya

RoleArn

Amazon Resource Name (ARN) dari IAM role yang memberikan QLDB izin untuk pengaliran jurnal untuk menulis catatan data ke sumber daya Kinesis Data Streams.

Untuk meneruskan peran ke QLDB saat meminta aliran jurnal, Anda harus memiliki izin untuk melakukan `iam:PassRole` tindakan pada sumber daya peran IAM. Ini diperlukan untuk semua permintaan aliran jurnal.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

Wajib: Ya

StreamName

Nama yang ingin Anda tetapkan ke pengaliran jurnal QLDB. Nama yang ditentukan pengguna dapat membantu mengidentifikasi dan menunjukkan tujuan pengaliran.

Nama pengaliran Anda harus unik di antara pengaliran aktif lainnya untuk buku besar yang ditentukan. Nama pengaliran memiliki batasan penamaan yang sama dengan nama buku besar, sebagaimana ditentukan di Kuota di Amazon QLDB di Panduan Developer Amazon QLDB.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

Tags

Pasangan kunci-nilai untuk ditambahkan sebagai tag ke aliran yang ingin Anda buat. Kunci tag peka huruf besar dan kecil. Nilai tag peka huruf besar/kecil dan bisa null.

Tipe: Peta string ke string

Entri Peta: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Batasan Panjang Kunci: Panjang minimum 1. Panjang maksimum 128.

Batasan Panjang Nilai: Panjang minimum 0. Panjang maksimum 256.

Wajib: Tidak

## Sintaksis Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "StreamId": "string"
}
```

### Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

[StreamId](#)

UUID (diwakili dalam teks yang disandikan Base62) yang diberikan QLDB ke setiap aliran jurnal QLDB.

Jenis: String

Kendala Panjang: Panjang tetap 22.

Pola: `^[A-Za-z-0-9]+$`

### Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidParameterException

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

ResourcePreconditionNotMetException

Operasi gagal karena kondisi tidak terpenuhi sebelumnya.

Kode Status HTTP: 412

### Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## TagResource

Layanan: Amazon QLDB

Menambahkan satu atau beberapa tag ke sumber daya QLDB Amazon tertentu.

Sumber daya dapat memiliki hingga 50 tag. Jika Anda mencoba membuat lebih dari 50 tag untuk sumber daya, permintaan Anda gagal dan mengembalikan kesalahan.

Minta Sintaks

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

resourceArn

Nama Sumber Daya Amazon (ARN) yang ingin Anda tambahkan tag. Sebagai contoh:

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

Wajib: Ya

Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

Tags

Pasangan kunci-nilai untuk ditambahkan sebagai tag ke sumber daya QLDB yang ditentukan. Kunci tag peka huruf besar dan kecil. Jika Anda menentukan kunci yang sudah ada untuk sumber

daya, permintaan Anda gagal dan mengembalikan kesalahan. Nilai tag peka huruf besar/kecil dan bisa null.

Tipe: Peta string ke string

Entri Peta: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Batasan Panjang Kunci: Panjang minimum 1. Panjang maksimum 128.

Batasan Panjang Nilai: Panjang minimum 0. Panjang maksimum 256.

Wajib: Ya

### Sintaksis Respons

```
HTTP/1.1 200
```

### Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

### Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidParameterException

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

### Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## UntagResource

Layanan: Amazon QLDB

Menghapus satu atau beberapa tag dari sumber daya QLDB Amazon tertentu. Anda dapat menentukan hingga 50 kunci tag untuk dihapus.

### Minta Sintaks

```
DELETE /tags/resourceArn?tagKeys=TagKeys HTTP/1.1
```

### Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

resourceArn

Nama Sumber Daya Amazon (ARN) untuk menghapus tag. Sebagai contoh:

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

Wajib: Ya

TagKeys

Daftar kunci tag untuk dihapus.

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Wajib: Ya

### Isi Permintaan

Permintaan tidak memiliki isi permintaan.

### Sintaks Respons

```
HTTP/1.1 200
```

### Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan isi HTTP kosong.

### Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidParameterException

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

### Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## UpdateLedger

Layanan: Amazon QLDB

Memperbarui properti pada buku besar.

### Minta Sintaks

```
PATCH /ledgers/name HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string"
}
```

### Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

#### name

Nama buku besar.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

### Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

#### DeletionProtection

Menentukan apakah buku besar dilindungi dari dihapus oleh setiap pengguna. Jika tidak ditentukan selama pembuatan buku besar, fitur ini diaktifkan (`true`) secara default.

Jika perlindungan penghapusan diaktifkan, Anda harus menonaktifkannya terlebih dahulu sebelum dapat menghapus buku besar. Anda dapat menonaktifkannya dengan memanggil `UpdateLedger` operasi untuk mengatur parameter ini`false`.

Tipe: Boolean

Wajib: Tidak

KmsKey

Kunci in AWS Key Management Service (AWS KMS) yang akan digunakan untuk enkripsi data saat istirahat di buku besar. Untuk informasi selengkapnya, lihat Enkripsi saat istirahat di Panduan Pengembang QLDB Amazon.

Gunakan salah satu opsi berikut untuk menentukan parameter ini:

- `AWS_OWNED_KMS_KEY`: Gunakan AWS KMS kunci yang dimiliki dan dikelola oleh AWS atas nama Anda.
- Undefined: Jangan membuat perubahan pada kunci KMS buku besar.
- Kunci KMS terkelola pelanggan simetris yang valid: Gunakan kunci KMS enkripsi simetris yang ditentukan di akun yang Anda buat, miliki, dan kelola.

  Amazon QLDB tidak mendukung kunci asimetris. Untuk informasi selengkapnya, lihat Menggunakan kunci simetris dan asimetris di Panduan AWS Key Management Service Pengembang.

Untuk menentukan kunci KMS yang dikelola pelanggan, Anda dapat menggunakan ID kunci, Nama Sumber Daya Amazon (ARN), nama alias, atau alias ARN. Saat menggunakan nama alias, awali dengan. `"alias/"` Untuk menentukan kunci yang berbeda Akun AWS, Anda harus menggunakan kunci ARN atau alias ARN.

Sebagai contoh:

- ID Kunci: `1234abcd-12ab-34cd-56ef-1234567890ab`
- ARN kunci: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Nama alias: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

Untuk informasi selengkapnya, lihat Pengidentifikasi kunci (KeyId) di Panduan AWS Key Management Service Pengembang.

Jenis: String

Kendala Panjang: Panjang maksimum 1600.

Wajib: Tidak

Sintaksis Respons

```
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"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

Arn

Nama Sumber Daya Amazon (ARN) untuk buku besar.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

CreationDateTime

Tanggal dan waktu, dalam format waktu epoch, saat buku besar dibuat. (Format waktu zaman adalah jumlah detik yang berlalu sejak 12:00:00 1 Januari 1970 UTC.)

Tipe: Timestamp

DeletionProtection

Menentukan apakah buku besar dilindungi dari dihapus oleh setiap pengguna. Jika tidak ditentukan selama pembuatan buku besar, fitur ini diaktifkan (`true`) secara default.

Jika perlindungan penghapusan diaktifkan, Anda harus menonaktifkannya terlebih dahulu sebelum dapat menghapus buku besar. Anda dapat menonaktifkannya dengan memanggil `UpdateLedger` operasi untuk mengatur parameter ini`false`.

Jenis: Boolean

EncryptionDescription

Informasi tentang enkripsi data saat istirahat di buku besar. Ini termasuk status saat ini, AWS KMS kunci, dan ketika kunci menjadi tidak dapat diakses (dalam kasus kesalahan).

Tipe: Objek LedgerEncryptionDescription

Name

Nama buku besar.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

State

Status buku besar saat ini.

Jenis: String

Nilai yang Valid: `CREATING | ACTIVE | DELETING | DELETED`

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidParameterException

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

### Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

# UpdateLedgerPermissionsMode

Layanan: Amazon QLDB

Memperbarui mode izin buku besar.

> **Important**
>
> Sebelum beralih ke mode `STANDARD` izin, Anda harus terlebih dahulu membuat semua kebijakan IAM dan tag tabel yang diperlukan untuk menghindari gangguan pada pengguna Anda. Untuk mempelajari selengkapnya, lihat Memigrasi ke mode izin standar di Panduan Pengembang QLDB Amazon.

## Minta Sintaks

```
PATCH /ledgers/name/permissions-mode HTTP/1.1
Content-type: application/json

{
   "PermissionsMode": "string"
}
```

## Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

### name

Nama buku besar.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

## Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

PermissionsMode

Mode izin untuk menetapkan ke buku besar. parameter ini dapat memiliki salah satu nilai berikut:

- `ALLOW_ALL`: Mode izin warisan yang memungkinkan kontrol akses dengan rincian tingkat API untuk buku besar.

  Mode ini memungkinkan pengguna yang memiliki izin API `SendCommand` untuk buku besar ini untuk menjalankan semua perintah PartiQL (maka, `ALLOW_ALL`) pada setiap tabel dalam buku besar yang ditentukan. Mode ini mengabaikan setiap kebijakan izin IAM tingkat tabel atau tingkat perintah yang Anda buat untuk buku besar.

- `STANDARD`: (Direkomendasikan) Mode perizinan yang memungkinkan kontrol akses dengan rincian yang lebih halus untuk buku besar, tabel, dan perintah PartiQL.

  Secara default, mode ini menyangkal semua permintaan pengguna untuk menjalankan perintah PartiQL pada setiap tabel dalam buku besar ini. Untuk mengizinkan perintah PartiQL untuk berjalan, Anda harus membuat kebijakan izin IAM untuk sumber daya tabel tertentu dan tindakan PartiQL, selain izin API `SendCommand` untuk buku besar. Untuk informasi, lihat Memulai dengan mode izin standar dalam Panduan Developer Amazon QLDB.

> **Note**
>
> Kami sangat merekomendasikan untuk menggunakan mode izin `STANDARD` untuk memaksimalkan keamanan data buku besar Anda.

Jenis: String

Nilai yang Valid: `ALLOW_ALL | STANDARD`

Wajib: Ya

Sintaksis Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "Name": "string",
```

```
   "PermissionsMode": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

Arn

Nama Sumber Daya Amazon (ARN) untuk buku besar.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

Name

Nama buku besar.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

Mode izin saat ini dari buku besar.

Jenis: String

Nilai yang Valid: `ALLOW_ALL | STANDARD`

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidParameterException

Satu atau beberapa parameter dalam permintaan tidak valid.

Kode Status HTTP: 400

ResourceNotFoundException

Sumber daya yang ditentukan tidak ada.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

## Sesi Amazon QLDB

Tindakan berikut didukung oleh Sesi Amazon QLDB:

- SendCommand

## SendCommand

Layanan: Amazon QLDB Session

Mengirim perintah ke buku besar QLDB Amazon.

> **Note**
>
> Alih-alih berinteraksi langsung dengan API ini, sebaiknya gunakan driver QLDB atau shell QLDB untuk menjalankan transaksi data pada buku besar.
>
> - Jika Anda bekerja dengan AWS SDK, gunakan driver QLDB. Driver menyediakan lapisan abstraksi tingkat tinggi di atas API data Sesi QLDB ini dan mengelola operasi untuk Anda. `SendCommand` Untuk informasi dan daftar bahasa pemrograman yang didukung, lihat Memulai driver di Panduan Pengembang QLDB Amazon.
> - Jika Anda bekerja dengan AWS Command Line Interface (AWS CLI), gunakan shell QLDB. Shell adalah antarmuka baris perintah yang menggunakan driver QLDB untuk berinteraksi dengan buku besar. Untuk selengkapnya, lihat Mengakses Amazon QLDB menggunakan shell QLDB.

### Sintaksis Permintaan

```
{
   "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": {
    }
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam format JSON.

AbortTransaction

Perintah untuk membatalkan transaksi saat ini.

Tipe: Objek AbortTransactionRequest

Wajib: Tidak

CommitTransaction

Perintah untuk melakukan transaksi yang ditentukan.

Tipe: Objek CommitTransactionRequest

Wajib: Tidak

EndSession

Perintah untuk mengakhiri sesi saat ini.

Tipe: Objek EndSessionRequest

Wajib: Tidak

ExecuteStatement

Perintah untuk mengeksekusi pernyataan dalam transaksi yang ditentukan.

Tipe: Objek ExecuteStatementRequest

Wajib: Tidak

FetchPage

Perintah untuk mengambil halaman.

Tipe: Objek FetchPageRequest

Wajib: Tidak

SessionToken

Menentukan token sesi untuk perintah saat ini. Token sesi konstan sepanjang masa sesi.

Untuk mendapatkan token sesi, jalankan `StartSession` perintah. Ini `SessionToken` diperlukan untuk setiap perintah berikutnya yang dikeluarkan selama sesi saat ini.

Jenis: String

Batasan Panjang: Panjang minimum 4. Panjang maksimum 1024.

Pola: `^[A-Za-z-0-9+/=]+$`

Wajib: Tidak

StartSession

Perintah untuk memulai sesi baru. Token sesi diperoleh sebagai bagian dari respons.

Tipe: Objek StartSessionRequest

Wajib: Tidak

StartTransaction

Perintah untuk memulai transaksi baru.

Tipe: Objek StartTransactionRequest

Wajib: Tidak

### Sintaksis Respons

```
{
   "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"
   }
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

AbortTransaction

Berisi rincian transaksi yang dibatalkan.

Tipe: Objek AbortTransactionResult

CommitTransaction

Berisi rincian transaksi yang dilakukan.

Tipe: Objek CommitTransactionResult

EndSession

Berisi rincian sesi yang berakhir.

Tipe: Objek EndSessionResult

ExecuteStatement

Berisi rincian pernyataan yang dieksekusi.

Tipe: Objek ExecuteStatementResult

FetchPage

Berisi detail halaman yang diambil.

Tipe: Objek FetchPageResult

StartSession

Berisi detail sesi yang dimulai yang mencakup token sesi. Ini `SessionToken` diperlukan untuk setiap perintah berikutnya yang dikeluarkan selama sesi saat ini.

Tipe: Objek StartSessionResult

StartTransaction

Berisi rincian transaksi yang dimulai.

Tipe: Objek StartTransactionResult

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

BadRequestException

Dikembalikan jika permintaan salah bentuk atau berisi kesalahan seperti nilai parameter yang tidak valid atau parameter wajib yang hilang.

Kode Status HTTP: 400

CapacityExceededException

Dikembalikan ketika permintaan melebihi kapasitas pemrosesan buku besar.

Kode Status HTTP: 400

InvalidSessionException

Dikembalikan jika sesi tidak ada lagi karena waktunya habis atau kedaluwarsa.

Kode Status HTTP: 400

LimitExceededException

Dikembalikan jika batas sumber daya seperti jumlah sesi aktif terlampaui.

Kode Status HTTP: 400

OccConflictException

Dikembalikan ketika transaksi tidak dapat ditulis ke jurnal karena kegagalan dalam fase verifikasi kontrol konkurensi optimis (OCC).

Kode Status HTTP: 400

RateExceededException

Dikembalikan ketika tingkat permintaan melebihi throughput yang diizinkan.

Kode Status HTTP: 400

## Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

# Tipe Data

tipe data berikut didukung oleh Amazon QLDB:

- JournalKinesisStreamDescription
- JournalS3ExportDescription
- KinesisConfiguration
- LedgerEncryptionDescription
- LedgerSummary
- S3EncryptionConfiguration
- S3ExportConfiguration
- ValueHolder

tipe data berikut didukung oleh Sesi Amazon QLDB:

- 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

Layanan: Amazon QLDB

Informasi tentang aliran jurnal QLDB Amazon, termasuk Nama Sumber Daya Amazon (ARN), nama aliran, waktu pembuatan, status saat ini, dan parameter permintaan pembuatan aliran asli.

Daftar Isi

KinesisConfiguration

Pengaturan konfigurasi tujuan Amazon Kinesis Data Streams untuk aliran jurnal QLDB.

Tipe: Objek KinesisConfiguration

Wajib: Ya

LedgerName

Nama buku besar.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

RoleArn

Amazon Resource Name (ARN) dari IAM role yang memberikan QLDB izin untuk pengaliran jurnal untuk menulis catatan data ke sumber daya Kinesis Data Streams.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

Wajib: Ya

Status

Keadaan aliran jurnal QLDB saat ini.

Jenis: String

Nilai yang Valid: `ACTIVE | COMPLETED | CANCELED | FAILED | IMPAIRED`

Wajib: Ya

StreamId

UUID (diwakili dalam teks yang disandikan Base62) dari aliran jurnal QLDB.

Jenis: String

Kendala Panjang: Panjang tetap 22.

Pola: `^[A-Za-z-0-9]+$`

Wajib: Ya

StreamName

Nama yang ditentukan pengguna dari aliran jurnal QLDB.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

Arn

Amazon Resource Name (ARN) dari pengaliran jurnal QLDB.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

Wajib: Tidak

CreationTime

Tanggal dan waktu, dalam format waktu zaman, ketika aliran jurnal QLDB dibuat. (Format waktu zaman adalah jumlah detik yang berlalu sejak 12:00:00 1 Januari 1970 UTC.)

Tipe: Timestamp

Wajib: Tidak

ErrorCause

Pesan kesalahan yang menjelaskan alasan bahwa aliran memiliki status `IMPAIRED` atau`FAILED`. Ini tidak berlaku untuk aliran yang memiliki nilai status lainnya.

Jenis: String

Nilai yang Valid: `KINESIS_STREAM_NOT_FOUND | IAM_PERMISSION_REVOKED`

Wajib: Tidak

ExclusiveEndTime

Tanggal dan waktu eksklusif yang menentukan kapan pengaliran berakhir. Jika parameter ini tidak terdefinisi, aliran berjalan tanpa batas hingga Anda membatalkannya.

Tipe: Timestamp

Wajib: Tidak

InclusiveStartTime

Tanggal dan waktu mulai inklusif untuk memulai data jurnal streaming.

Tipe: Timestamp

Wajib: Tidak

## Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## JournalS3ExportDescription

Layanan: Amazon QLDB

Informasi tentang pekerjaan ekspor jurnal, termasuk nama buku besar, ID ekspor, waktu pembuatan, status saat ini, dan parameter permintaan pembuatan ekspor asli.

Daftar Isi

ExclusiveEndTime

Tanggal dan waktu akhir eksklusif untuk rentang isi jurnal yang ditentukan dalam permintaan ekspor asli.

Tipe: Timestamp

Wajib: Ya

ExportCreationTime

Tanggal dan waktu, dalam format waktu epoch, saat pekerjaan ekspor dibuat. (Format waktu zaman adalah jumlah detik yang berlalu sejak 12:00:00 1 Januari 1970 UTC.)

Tipe: Timestamp

Wajib: Ya

ExportId

UUID (diwakili dalam teks yang disandikan Base62) dari pekerjaan ekspor jurnal.

Jenis: String

Kendala Panjang: Panjang tetap 22.

Pola: `^[A-Za-z-0-9]+$`

Wajib: Ya

InclusiveStartTime

Tanggal dan waktu mulai inklusif untuk rentang isi jurnal yang ditentukan dalam permintaan ekspor asli.

Tipe: Timestamp

Wajib: Ya

LedgerName

Nama buku besar.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Ya

RoleArn

Nama Sumber Daya Amazon (ARN) dari peran IAM yang memberikan izin QLDB untuk pekerjaan ekspor jurnal untuk melakukan hal berikut:

- Tulis objek ke dalam bucket Amazon Simple Storage Service (Amazon S3).
- (Opsional) Gunakan kunci terkelola pelanggan Anda di AWS Key Management Service (AWS KMS) untuk enkripsi sisi server dari data yang Anda ekspor.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

Wajib: Ya

S3ExportConfiguration

Lokasi bucket Amazon Simple Storage Service (Amazon S3) tempat pekerjaan ekspor jurnal menulis konten jurnal.

Tipe: Objek S3ExportConfiguration

Wajib: Ya

Status

Keadaan pekerjaan ekspor jurnal saat ini.

Jenis: String

Nilai yang Valid: `IN_PROGRESS | COMPLETED | CANCELLED`

Wajib: Ya

OutputFormat

Format output dari data jurnal yang diekspor.

Jenis: String

Nilai yang Valid: `ION_BINARY | ION_TEXT | JSON`

Wajib: Tidak

## Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## KinesisConfiguration

Layanan: Amazon QLDB

Pengaturan konfigurasi tujuan Amazon Kinesis Data Streams tujuan untuk pengaliran jurnal Amazon QLDB.

Daftar Isi

StreamArn

Amazon Resource Name (ARN) sumber daya Kinesis Data Streams.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

Wajib: Ya

AggregationEnabled

Memungkinkan QLDB untuk memublikasikan beberapa catatan data dalam satu catatan Kinesis Data Streams, meningkatkan jumlah rekaman yang dikirim per panggilan API.

Default: `True`

> **Important**
>
> Agregasi rekaman memiliki implikasi penting untuk memproses catatan dan membutuhkan de-agregasi di pelanggan pengaliran Anda. Untuk mempelajari lebih lanjut, lihat Konsep kunci KPL dan Konsumen de-agregasi di Panduan Developer Amazon Kinesis Data Streams.

Tipe: Boolean

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## LedgerEncryptionDescription

Layanan: Amazon QLDB

Informasi tentang enkripsi data saat istirahat di buku besar Amazon QLDB. Ini termasuk status saat ini, kunci di AWS Key Management Service (AWS KMS), dan ketika kunci menjadi tidak dapat diakses (dalam kasus kesalahan).

Untuk informasi selengkapnya, lihat Enkripsi saat istirahat di Panduan Pengembang QLDB Amazon.

Daftar Isi

EncryptionStatus

Keadaan enkripsi saat ini saat istirahat untuk buku besar. Ini dapat berupa salah satu dari nilai berikut:

- `ENABLED`: Enkripsi sepenuhnya diaktifkan menggunakan kunci yang ditentukan.
- `UPDATING`: Buku besar secara aktif memproses perubahan kunci yang ditentukan.

  Perubahan utama dalam QLDB bersifat asinkron. Buku besar dapat diakses sepenuhnya tanpa dampak kinerja apa pun saat perubahan kunci sedang diproses. Jumlah waktu yang diperlukan untuk memperbarui kunci bervariasi tergantung pada ukuran buku besar.
- `KMS_KEY_INACCESSIBLE`: Kunci KMS yang dikelola pelanggan yang ditentukan tidak dapat diakses, dan buku besar terganggu. Entah kunci dinonaktifkan atau dihapus, atau hibah pada kunci dicabut. Ketika buku besar terganggu, buku itu tidak dapat diakses dan tidak menerima permintaan baca atau tulis apa pun.

  Buku besar yang rusak secara otomatis kembali ke status aktif setelah Anda mengembalikan hibah pada kunci, atau mengaktifkan kembali kunci yang dinonaktifkan. Namun, menghapus kunci KMS yang dikelola pelanggan tidak dapat diubah. Setelah kunci dihapus, Anda tidak dapat lagi mengakses buku besar yang dilindungi dengan kunci itu, dan data menjadi tidak dapat dipulihkan secara permanen.

Jenis: String

Nilai yang Valid: `ENABLED | UPDATING | KMS_KEY_INACCESSIBLE`

Wajib: Ya

KmsKeyArn

Nama Sumber Daya Amazon (ARN) dari kunci KMS yang dikelola pelanggan yang digunakan buku besar untuk enkripsi saat istirahat. Jika parameter ini tidak terdefinisi, buku besar menggunakan kunci KMS yang AWS dimiliki untuk enkripsi. Ini akan ditampilkan `AWS_OWNED_KMS_KEY` saat memperbarui konfigurasi enkripsi buku besar ke kunci KMS yang AWS dimiliki.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

Wajib: Ya

InaccessibleKmsKeyDateTime

Tanggal dan waktu, dalam format waktu zaman, ketika AWS KMS kunci pertama kali menjadi tidak dapat diakses, dalam kasus kesalahan. (Format waktu zaman adalah jumlah detik yang telah berlalu sejak 12:00:00 1 Januari 1970 UTC.)

Parameter ini tidak ditentukan jika AWS KMS kunci dapat diakses.

Tipe: Timestamp

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## LedgerSummary

Layanan: Amazon QLDB

Informasi tentang buku besar, termasuk namanya, negara bagian, dan kapan buku itu dibuat.

Daftar Isi

CreationDateTime

Tanggal dan waktu, dalam format waktu epoch, saat buku besar dibuat. (Format waktu zaman adalah jumlah detik yang berlalu sejak 12:00:00 1 Januari 1970 UTC.)

Tipe: Timestamp

Wajib: Tidak

Name

Nama buku besar.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Wajib: Tidak

State

Status buku besar saat ini.

Jenis: String

Nilai yang Valid: `CREATING | ACTIVE | DELETING | DELETED`

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

## S3EncryptionConfiguration

Layanan: Amazon QLDB

Pengaturan enkripsi yang digunakan oleh pekerjaan ekspor jurnal untuk menulis data dalam bucket Amazon Simple Storage Service (Amazon S3).

Daftar Isi

ObjectEncryptionType

Jenis enkripsi objek Amazon S3.

Untuk mempelajari lebih lanjut tentang opsi enkripsi sisi server di Amazon S3, lihat Melindungi Data Menggunakan Enkripsi Sisi Server di Panduan Pengembang Amazon S3.

Jenis: String

Nilai yang Valid: `SSE_KMS | SSE_S3 | NO_ENCRYPTION`

Wajib: Ya

KmsKeyArn

Nama Sumber Daya Amazon (ARN) dari kunci enkripsi simetris di AWS Key Management Service ().AWS KMS Amazon S3 tidak mendukung kunci KMS asimetris.

Anda harus memberikan `KmsKeyArn` jika Anda menentukan `SSE_KMS` sebagai`ObjectEncryptionType`.

`KmsKeyArn`tidak diperlukan jika Anda menentukan `SSE_S3` sebagai`ObjectEncryptionType`.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1600.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

## S3ExportConfiguration

Layanan: Amazon QLDB

Lokasi bucket Amazon Simple Storage Service (Amazon S3) tempat pekerjaan ekspor jurnal menulis konten jurnal.

Daftar Isi

Bucket

Nama bucket Amazon S3 di mana pekerjaan ekspor jurnal menulis konten jurnal.

Nama bucket harus sesuai dengan konvensi penamaan bucket Amazon S3. Untuk informasi selengkapnya, lihat Pembatasan dan Batasan Bucket di Panduan Pengembang Amazon S3.

Jenis: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 255.

Pola: `^[A-Za-z-0-9-_.]+$`

Wajib: Ya

EncryptionConfiguration

Pengaturan enkripsi yang digunakan oleh pekerjaan ekspor jurnal untuk menulis data dalam bucket Amazon S3.

Tipe: Objek S3EncryptionConfiguration

Wajib: Ya

Prefix

Awalan untuk bucket Amazon S3 di mana pekerjaan ekspor jurnal menulis konten jurnal.

Awalan harus mematuhi aturan dan batasan penamaan kunci Amazon S3. Untuk informasi selengkapnya, lihat Object Key dan Metadata di Panduan Pengembang Amazon S3.

Berikut ini adalah contoh `Prefix` nilai yang valid:

- `JournalExports-ForMyLedger/Testing/`
- `JournalExports`
- `My:Tests/`

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 128.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

### ValueHolder

Layanan: Amazon QLDB

Struktur yang dapat berisi nilai dalam beberapa format pengkodean.

Daftar Isi

IonText

Nilai plaintext Amazon Ion yang terkandung dalam struktur. `ValueHolder`

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1048576.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## Pengsesi Amazon QLDB

tipe data berikut didukung oleh Pengsesi Amazon QLDB:

- AbortTransactionRequest
- AbortTransactionResult
- CommitTransactionRequest
- CommitTransactionResult
- EndSessionRequest
- EndSessionResult
- ExecuteStatementRequest
- ExecuteStatementResult

- FetchPageRequest
- FetchPageResult
- IOUsage
- Page
- StartSessionRequest
- StartSessionResult
- StartTransactionRequest
- StartTransactionResult
- TimingInformation
- ValueHolder

## AbortTransactionRequest

Layanan: Amazon QLDB Session

Berisi rincian transaksi yang akan dibatalkan.

Daftar Isi

Anggota struktur pengecualian ini bergantung pada konteks.

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## AbortTransactionResult

Layanan: Amazon QLDB Session

Berisi rincian transaksi yang dibatalkan.

Daftar Isi

TimingInformation

Berisi informasi kinerja sisi server untuk perintah.

Tipe: Objek TimingInformation

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## CommitTransactionRequest

Layanan: Amazon QLDB Session

Berisi rincian transaksi yang akan dilakukan.

Daftar Isi

CommitDigest

Menentukan intisari komit untuk transaksi untuk melakukan. Untuk setiap transaksi aktif, intisari komit harus dilewati. QLDB `CommitDigest` memvalidasi dan menolak komit dengan kesalahan jika intisari yang dihitung pada klien tidak cocok dengan intisari yang dihitung oleh QLDB.

Tujuan dari `CommitDigest` parameter ini adalah untuk memastikan bahwa QLDB melakukan transaksi jika dan hanya jika server telah memproses kumpulan pernyataan yang dikirim oleh klien, dalam urutan yang sama dengan yang dikirim klien, dan tanpa duplikat.

Tipe: Objek data biner dienkode Base64

Wajib: Ya

TransactionId

Menentukan ID transaksi transaksi untuk melakukan.

Jenis: String

Kendala Panjang: Panjang tetap 22.

Pola: `^[A-Za-z-0-9]+$`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## CommitTransactionResult

Layanan: Amazon QLDB Session

Berisi rincian transaksi yang dilakukan.

Daftar Isi

CommitDigest

Intisari komit dari transaksi yang dilakukan.

Tipe: Objek data biner berkode Base64

Wajib: Tidak

ConsumedIOs

Berisi metrik tentang jumlah permintaan I/O yang digunakan.

Tipe: Objek IOUsage

Wajib: Tidak

TimingInformation

Berisi informasi kinerja sisi server untuk perintah.

Tipe: Objek TimingInformation

Wajib: Tidak

TransactionId

ID transaksi dari transaksi yang dilakukan.

Jenis: String

Kendala Panjang: Panjang tetap 22.

Pola: `^[A-Za-z-0-9]+$`

Diperlukan: Tidak

### Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## EndSessionRequest

Layanan: Amazon QLDB Session

Menentukan permintaan untuk mengakhiri sesi.

Daftar Isi

Anggota struktur pengecualian ini bergantung pada konteks.

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## EndSessionResult

Layanan: Amazon QLDB Session

Berisi rincian sesi yang berakhir.

Daftar Isi

TimingInformation

Berisi informasi kinerja sisi server untuk perintah.

Tipe: Objek TimingInformation

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## ExecuteStatementRequest

Layanan: Amazon QLDB Session

Menentukan permintaan untuk mengeksekusi pernyataan.

Daftar Isi

Statement

Menentukan pernyataan permintaan.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100000.

Wajib: Ya

TransactionId

Menentukan ID transaksi permintaan.

Jenis: String

Kendala Panjang: Panjang tetap 22.

Pola: `^[A-Za-z-0-9]+$`

Wajib: Ya

Parameters

Menentukan parameter untuk pernyataan parameterized dalam permintaan.

Tipe: Array objek ValueHolder

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

## ExecuteStatementResult

Layanan: Amazon QLDB Session

Berisi rincian pernyataan yang dieksekusi.

### Daftar Isi

**ConsumedIOs**

Berisi metrik tentang jumlah permintaan I/O yang digunakan.

Tipe: Objek IOUsage

Wajib: Tidak

**FirstPage**

Berisi detail halaman pertama yang diambil.

Tipe: Objek Page

Wajib: Tidak

**TimingInformation**

Berisi informasi kinerja sisi server untuk perintah.

Tipe: Objek TimingInformation

Wajib: Tidak

### Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## FetchPageRequest

Layanan: Amazon QLDB Session

Menentukan rincian halaman yang akan diambil.

### Daftar Isi

NextPageToken

Menentukan token halaman berikutnya dari halaman yang akan diambil.

Jenis: String

Batasan Panjang: Panjang minimum 4. Panjang maksimum 1024.

Pola: `^[A-Za-z-0-9+/=]+$`

Wajib: Ya

TransactionId

Menentukan ID transaksi dari halaman yang akan diambil.

Jenis: String

Kendala Panjang: Panjang tetap 22.

Pola: `^[A-Za-z-0-9]+$`

Diperlukan: Ya

### Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## FetchPageResult

Layanan: Amazon QLDB Session

Berisi halaman yang diambil.

Daftar Isi

ConsumedIOs

Berisi metrik tentang jumlah permintaan I/O yang digunakan.

Tipe: Objek IOUsage

Wajib: Tidak

Page

Berisi rincian halaman yang diambil.

Tipe: Objek Page

Wajib: Tidak

TimingInformation

Berisi informasi kinerja sisi server untuk perintah.

Tipe: Objek TimingInformation

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## IOUsage

Layanan: Amazon QLDB Session

Berisi metrik penggunaan I/O untuk perintah yang dipanggil.

Daftar Isi

ReadIOs

Jumlah permintaan I/O baca yang dibuat perintah.

Tipe: Panjang

Wajib: Tidak

WriteIOs

Jumlah permintaan I/O tulis yang dibuat perintah.

Tipe: Panjang

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## Page

Layanan: Amazon QLDB Session

Berisi rincian halaman yang diambil.

Daftar Isi

NextPageToken

Token dari halaman berikutnya.

Jenis: String

Batasan Panjang: Panjang minimum 4. Panjang maksimum 1024.

Pola: `^[A-Za-z-0-9+/=]+$`

Wajib: Tidak

Values

Struktur yang berisi nilai dalam berbagai format pengkodean.

Tipe: Array objek ValueHolder

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## StartSessionRequest

Layanan: Amazon QLDB Session

Menentukan permintaan untuk memulai sesi baru.

Daftar Isi

LedgerName

Nama buku besar untuk memulai sesi baru melawan.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 32.

Pola: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## StartSessionResult

Layanan: Amazon QLDB Session

Berisi rincian sesi yang dimulai.

Daftar Isi

SessionToken

Token sesi dari sesi yang dimulai. Ini `SessionToken` diperlukan untuk setiap perintah berikutnya yang dikeluarkan selama sesi saat ini.

Jenis: String

Batasan Panjang: Panjang minimum 4. Panjang maksimum 1024.

Pola: `^[A-Za-z-0-9+/=]+$`

Wajib: Tidak

TimingInformation

Berisi informasi kinerja sisi server untuk perintah.

Tipe: Objek TimingInformation

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## StartTransactionRequest

Layanan: Amazon QLDB Session

Menentukan permintaan untuk memulai transaksi.

Daftar Isi

Anggota struktur pengecualian ini bergantung pada konteks.

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## StartTransactionResult

Layanan: Amazon QLDB Session

Berisi rincian transaksi yang dimulai.

Daftar Isi

TimingInformation

Berisi informasi kinerja sisi server untuk perintah.

Tipe: Objek TimingInformation

Wajib: Tidak

TransactionId

ID transaksi dari transaksi yang dimulai.

Jenis: String

Kendala Panjang: Panjang tetap 22.

Pola: `^[A-Za-z-0-9]+$`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## TimingInformation

Layanan: Amazon QLDB Session

Berisi informasi kinerja sisi server untuk sebuah perintah. Amazon QLDB menangkap informasi waktu antara waktu ketika menerima permintaan dan ketika mengirim respons yang sesuai.

Daftar Isi

ProcessingTimeMilliseconds

Jumlah waktu yang dihabiskan QLDB untuk memproses perintah, diukur dalam milidetik.

Tipe: Panjang

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

### ValueHolder

Layanan: Amazon QLDB Session

Struktur yang dapat berisi nilai dalam beberapa format pengkodean.

Daftar Isi

IonBinary

Nilai biner Amazon Ion yang terkandung dalam suatu `ValueHolder` struktur.

Tipe: Objek data biner dienkode Base64

Panjang Batasan: Panjang minimum 1. Panjang maksimum 131072.

Wajib: Tidak

IonText

Nilai plaintext Amazon Ion yang terkandung dalam struktur. `ValueHolder`

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1048576.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

## Kesalahan Umum

Bagian ini berisi daftar kesalahan yang umum terjadi pada tindakan API dari semua layanan AWS. Untuk kesalahan khusus pada tindakan API untuk layanan ini, lihat topik untuk tindakan API tersebut.

AccessDeniedException

Anda tidak memiliki akses yang memadai untuk melakukan tindakan ini.

Kode Status HTTP: 400

IncompleteSignature

Tanda tangan permintaan tidak sesuai dengan standar AWS.

Kode Status HTTP: 400

InternalFailure

Pemrosesan permintaan telah gagal karena kesalahan yang tidak diketahui, pengecualian atau kegagalan.

Kode Status HTTP: 500

InvalidAction

Tindakan atau operasi yang diminta tidak valid. Verifikasi bahwa tindakan diketik dengan benar.

Kode Status HTTP: 400

InvalidClientTokenId

Sertifikat X.509 atau access key ID AWS yang diberikan tidak ada dalam catatan kami.

Kode Status HTTP: 403

NotAuthorized

Anda tidak memiliki izin untuk melakukan tindakan ini.

Kode Status HTTP: 400

OptInRequired

Access key ID AWS membutuhkan berlangganan untuk layanan.

Kode Status HTTP: 403

RequestExpired

Permintaan menjangkau layanan lebih dari 15 menit setelah stempel tanggal pada permintaan atau lebih dari 15 menit setelah tanggal kedaluwarsa permintaan (seperti untuk URL pre-signed), atau stempel tanggal pada permintaan lebih dari 15 menit di masa mendatang.

Kode Status HTTP: 400

ServiceUnavailable

Permintaan telah gagal karena kegagalan sementara server.

Kode Status HTTP: 503

ThrottlingException

Permintaan ditolak karena throttling permintaan.

Kode Status HTTP: 400

ValidationError

Input gagal untuk memenuhi batasan yang ditentukan oleh layanan AWS.

Kode Status HTTP: 400

# Parameter Umum

Daftar berikut berisi parameter yang digunakan semua tindakan untuk menandatangani permintaan Tanda Tangan Versi 4 dengan string kueri. Setiap parameter khusus tindakan tercantum dalam topik untuk tindakan tersebut. Untuk informasi selengkapnya tentang Signature Versi 4, lihat Menandatangani permintaanAWS API di Panduan Pengguna IAM.

Action

Tindakan yang harus dilakukan.

Tipe: string

Wajib: Ya

Version

Versi API yang ditulis dalam permintaan, dinyatakan dalam format HH-BB-TTTT.

Tipe: string

Wajib: Ya

X-Amz-Algorithm

Algoritme hash yang Anda gunakan untuk membuat tanda tangan permintaan.

Syarat: Tentukan parameter ini ketika Anda menyertakan informasi autentikasi dalam string kueri alih-alih di header otorisasi HTTP.

Tipe: string

Nilai yang Valid: `AWS4-HMAC-SHA256`

Diperlukan: Kondisional

X-Amz-Credential

Nilai lingkup kredensial, yang merupakan string yang menyertakan access key Anda, tanggal, wilayah yang Anda targetkan, layanan yang Anda minta, dan string penghentian ("aws4_request"). Nilai dinyatakan dalam format berikut: access_key/HHBBTTTT/wilayah/layanan/aws4_request.

Untuk informasi selengkapnya, lihat Membuat permintaanAWS API yang ditandatangani di Panduan Pengguna IAM.

Syarat: Tentukan parameter ini ketika Anda menyertakan informasi autentikasi dalam string kueri alih-alih di header otorisasi HTTP.

Tipe: string

Diperlukan: Kondisional

X-Amz-Date

Tanggal yang digunakan untuk membuat tanda tangan. Format harus berupa format dasar ISO 8601 (YYYYMMDD'T'HMMSS'Z'). Misalnya, waktu tanggal berikut adalah nilai X-Amz-Date yang valid: `20120325T120000Z`.

Syarat: X-Amz-Date bersifat opsional untuk semua permintaan; nilai ini dapat digunakan untuk mengganti tanggal yang digunakan untuk menandatangani permintaan. Jika header Tanggal ditentukan dalam format dasar ISO 8601, X-Amz-Date tidak diperlukan. Ketika X-Amz-Date digunakan, ia selalu mengganti nilai header Tanggal. Untuk informasi selengkapnya, lihat Elemen tanda tangan permintaanAWS API di Panduan Pengguna IAM.

Tipe: string

Diperlukan: Kondisional

X-Amz-Security-Token

Token keamanan sementara yang diperoleh melalui panggilan keAWS Security Token Service (AWS STS). Untuk daftar layanan yang mendukung kredenial keamanan sementaraAWS STS, lihat Layanan AWSbahwa bekerja dengan IAM dalam Panduan Pengguna IAM.

Syarat: Jika Anda menggunakan kredenial keamanan.AWS STS

Tipe: string

Diperlukan: Kondisional

X-Amz-Signature

Menentukan tanda tangan yang dikodekan oleh hex yang dihitung dari string to sign dan kunci penandatanganan turunan.

Syarat: Tentukan parameter ini ketika Anda menyertakan informasi autentikasi dalam string kueri alih-alih di header otorisasi HTTP.

Tipe: string

Diperlukan: Kondisional

X-Amz-SignedHeaders

Menentukan semua header HTTP yang disertakan sebagai bagian dari permintaan kanonik. Untuk informasi selengkapnya tentang menentukan header yang ditandatangani, lihat Membuat permintaanAWS API yang ditandatangani di Panduan Pengguna IAM.

Syarat: Tentukan parameter ini ketika Anda menyertakan informasi autentikasi dalam string kueri alih-alih di header otorisasi HTTP.

Tipe: string

Diperlukan: Kondisional

# Kuota dan batasan di Amazon QLDB

Bagian ini menjelaskan kuota saat ini, juga disebut sebagai batasan, di Amazon QLDB.

Topik



## Kuota default

QLDB memiliki kuota default berikut, seperti yang juga tercantum di endpoint dan kuota Amazon QLDB di Referensi Umum AWS. Kuota iniAkun AWS per Wilayah. Untuk meminta kenaikan kuota untuk akun Anda di suatu Wilayah, gunakan konsol Service Quotas.

Masuk keAWS Management Console dan buka konsol Service Quotas di https://console.aws.amazon.com/servicequotas/.

| Resource | Kuota default |
| --- | --- |
| Jumlah maksimum buku besar aktif yang dapat Anda buat di akun di akun di akun di akun di akun di akun di akun di akun di akun di akun di akun di akun di akun di akun di akun di akun di akun di akun di akun ini. | 5 |
| Jumlah maksimum ekspor jurnal aktif ke Amazon S3 per buku besar | 2 |
| Jumlah maksimum pengaliran jurnal aktif ke Kinesis Data Streams per buku besar | 5 |

# kuota tetap

Selain kuota default, QLDB memiliki kuota tetap berikut per buku besar. Kuota ini tidak dapat ditingkatkan dengan menggunakan Service Quotas:

| Resource | Kuota tetap |
| --- | --- |
| Jumlah sesi aktif bersamaan | 1500 |
| Jumlah tabel aktif | 20 |
| Jumlah total tabel (aktif dan tidak aktif)<br><br>**Note**<br>Di QLDB, tabel yang dijatuhkan dianggap tidak aktif dan dihitung terhadap total kuota ini. | 40 |
| Jumlah indeks per tabel | 5 |
| Jumlah dokumen dalam suatu transaksi | 40 |
| Jumlah revisi yang akan disunting dalam suatu transaksi | 1 |
| Ukuran dokumen (dikodekan dalam `IonBinary` format) | 128 KB |
| Ukuran parameter pernyataan (`IonBinary` format) | 128 KB |
| Ukuran parameter pernyataan (`IonText` format) | 1 MB |
| Panjang string pernyataan | 100.000 karakter |
| Ukuran transaksi | 4 MB |

| Resource | Kuota tetap |
| --- | --- |
| Batas waktu transaksi | 30 detik |
| Periode kedaluwarsa untuk pekerjaan ekspor jurnal yang telah selesai | 7 hari |
| Periode kedaluwarsa untuk aliran jurnal terminal | 7 hari |

## Kuota buku besar

Untuk meminta kenaikan kuota buku besar untuk akun Anda di suatu Wilayah, Anda dapat menggunakan konsol Service Quotas.

Buka konsol Service Quotas di https://console.aws.amazon.com/servicequotas/.

Beberapa kasus penggunaan QLDB memerlukan perluasan jumlah buku besarAkun AWS per Wilayah berdasarkan pertumbuhan bisnis. Misalnya, Anda mungkin perlu membuat buku besar khusus untuk mengisolasi pelanggan atau data. Dalam hal ini, pertimbangkan untuk memanfaatkan arsitektur multi-akun untuk bekerja dengan kuota QLDB. Untuk informasi selengkapnya, lihat Isolasi Silo Akun diAWS whitepaper Strategi Isolasi Penyewa SaaS.

## Ukuran dokumen

Ukuran maksimum untuk dokumen yang dikodekan dalam`IonBinary` format adalah 128 KB. Kami tidak dapat memberikan batas yang tepat untuk ukuran dokumen dalam`IonText` format karena konversi dari teks ke biner bervariasi secara signifikan berdasarkan struktur setiap dokumen. QLDB mendukung dokumen dengan konten terbuka, sehingga setiap struktur dokumen yang unik mengubah perhitungan ukuran.

## Ukuran transaksi

Ukuran maksimum untuk transaksi di QLDB adalah 4 MB. Ukuran transaksi dihitung berdasarkan jumlah dari faktor-faktor berikut.

Delta

Perubahan dokumen yang dihasilkan oleh semua pernyataan dalam transaksi. Dalam transaksi yang memengaruhi beberapa dokumen, ukuran delta total adalah jumlah delta individu masing-masing dokumen yang terpengaruh.

Metadata

Metadata transaksi yang dihasilkan sistem yang dikaitkan dengan setiap dokumen yang terpengaruh.

Indeks

Jika indeks didefinisikan pada tabel yang dipengaruhi oleh transaksi, entri indeks terkait juga menghasilkan delta.

Riwayat

Karena semua revisi dokumen tetap ada di QLDB, semua transaksi juga ditambahkan ke sejarah.

Sisipan - Setiap dokumen yang dimasukkan ke dalam tabel juga memiliki salinan yang dimasukkan ke dalam tabel sejarahnya. Misalnya, dokumen 100 KB yang baru dimasukkan menghasilkan minimal 200 KB delta dalam transaksi. (Ini adalah perkiraan kasar yang tidak menyertakan metadata atau indeks.)

Pembaruan - Setiap pembaruan dokumen, bahkan untuk satu bidang, membuat revisi baru dari seluruh dokumen dalam sejarah, plus atau minus delta pembaruan. Ini berarti bahwa pembaruan kecil dalam dokumen besar masih akan menghasilkan delta transaksi besar. Misalnya, menambahkan 2 KB data dalam dokumen 100 KB yang ada membuat revisi 102 KB baru dalam sejarah. Ini menambahkan hingga setidaknya 104 KB dari total delta dalam suatu transaksi. (Sekali lagi, perkiraan ini tidak menyertakan metadata atau indeks.)

Menghapus - Mirip dengan pembaruan, setiap transaksi hapus membuat revisi dokumen baru dalam sejarah. Namun,`DELETE` revisi yang baru dibuat lebih kecil dari dokumen asli karena memiliki data pengguna null dan hanya berisi metadata.

# kendala penamaan

Tabel berikut menjelaskan batasan penamaan di Amazon QLDB.

| | |
|---|---|
| Nama buku besar | • Harus hanya mengandung dari 1—32 karakter alfanumerik atau tanda hubung. |

| | |
|---|---|
| Nama aliran jurnal | • Harus memiliki huruf atau angka untuk karakter pertama dan terakhir.<br>• Tidak boleh semua angka.<br>• Tidak dapat berisi dua tanda hubung berturut-turut.<br>• Apakah peka terhadap huruf besar dan kecil. |
| Nama tabel | • Harus hanya berisi 1-128 karakter alfanumerik atau garis bawah.<br>• Harus memiliki sebuah huruf atau garis bawah untuk karakter pertama.<br>• Dapat berisi kombinasi karakter alfanumerik dan garis bawah untuk karakter yang tersisa.<br>• Apakah peka terhadap huruf besar dan kecil.<br>• Tidak harus berupa kata QLDB PartiQL reserved. |

# Informasi terkait Amazon QLDB

Sumber daya terkait berikut dapat membantu Anda saat bekerja dengan layanan ini.

Topik



## Dokumen teknis

- FAQ Amazon QLDB — Pertanyaan yang sering diajukan tentang produk.
- Harga Amazon QLDB — informasiAWS harga dan contoh.
- AWS re:Post- forumAWS komunitas untuk pertanyaan dan jawaban (Tanya Jawab).
- Amazon Ion — Panduan pengembang, panduan pengguna, dan referensi untuk format data Amazon Ion.
- PartiQL - Dokumen spesifikasi dan tutorial umum untuk bahasa kueri PartiQL.
- Lokakarya QLDB — Lokakarya yang memberikan contoh praktis penggunaan Amazon QLDB untuk membangun system-of-record aplikasi, termasuk laboratorium berikut:
  - Dasar-dasar QLDB
  - Bekerja dengan Amazon Ion, dan mengonversi Ion ke dan dari JSON (Java)
  - MenggunakanAWS Glue dan Amazon Athena untuk mengaktifkan data QLDB untuk data lake
  - Menautkan instans DB Amazon Aurora MyLDB
- Data Kualitas Bukti Tamper Menggunakan Amazon QLDB — ImplementasiAWS Solusi yang menunjukkan cara mencegah penyerang merusak data berkualitas dengan menggunakan QLDB untuk mempertahankan riwayat perubahan data yang akurat. AWS Implementasi Solusi membantu Anda memecahkan masalah umum dan membangun lebih cepat menggunakanAWS.

# GitHub repositori

Pengemudi

- .NET driver - Sebuah implementasi .NET dari driver QLDB.
- Go driver - Sebuah implementasi Go driver QLDB.
- Java driver - Sebuah implementasi Java dari driver QLDB.
- Node.js driver - Sebuah implementasi Node.js dari driver QLDB.
- Python driver - Sebuah implementasi Python dari driver QLDB.

Shell baris perintah

- Shell QLDB - Implementasi Python dan Rust dari antarmuka baris perintah untuk API data transaksional QLDB.

Aplikasi sampel

- Aplikasi Java DMV - Aplikasi tutorial yang didasarkan pada kasus penggunaan departemen kendaraan bermotor (DMV). Ini menunjukkan operasi dasar dan praktik terbaik untuk menggunakan QLDB dan driver QLDB untuk Java.
- Aplikasi .NET DMV - Sebuah aplikasi tutorial berbasis DMV yang menunjukkan operasi dasar dan praktik terbaik untuk menggunakan QLDB dan driver QLDB untuk .NET.
- Aplikasi DMV Node.js - Aplikasi tutorial berbasis DMV yang menunjukkan operasi dasar dan praktik terbaik untuk menggunakan QLDB dan driver QLDB untuk Node.js.
- Aplikasi Python DMV - Aplikasi tutorial berbasis DMV yang menunjukkan operasi dasar dan praktik terbaik untuk menggunakan QLDB dan driver QLDB untuk Python.
- Pemuat buku besar — Kerangka kerja Java untuk memuat data secara asinkron ke buku besar QLDB dengan kecepatan tinggi menggunakan saluran pengiriman yang didukung (AWS DMS, Amazon SQS, Amazon SNS, Kinesis Data Streams, Amazon MSK, atau EventBridge).
- Prosesor ekspor — Kerangka kerja Java yang dapat diperluas yang menangani pekerjaan pemrosesan ekspor QLDB di Amazon S3 dengan membaca output ekspor dan melakukan iterasi melalui blok yang diekspor secara berurutan.
- QLDB mengalirkan sampel Lambda dengan Python — Aplikasi yang menunjukkan cara mengonsumsi aliran QLDB dengan menggunakanAWS Lambda fungsi untuk mengirim data QLDB ke topik Amazon SNS, yang memiliki antrian Amazon SQS yang berlangganan.

- Sampel OpenSearch integrasi aliran QLDB — Aplikasi Python yang menunjukkan cara mengintegrasikan Amazon OpenSearch Service dengan QLDB dengan menggunakan stream.
- Aplikasi double-entry - Sebuah aplikasi Java yang menunjukkan bagaimana memodelkan aplikasi buku besar keuangan double-entry menggunakan QLDB.
- QLDB KVS untuk Node.js - Pustaka antarmuka penyimpanan kunci-nilai sederhana untuk QLDB dengan fungsi tambahan untuk verifikasi dokumen.

Amazon Ion dan PartiQL

- Pustaka Amazon Ion — Pustaka, alat, dan dokumentasi yang didukung tim Ion.
- implementasi PartiQL - Implementasi, spesifikasi, dan tutorial untuk PartiQL.

## AWSposting blog dan artikel

- Bagaimana Earnin membangun layanan buku besar mereka menggunakan Amazon QLDB (16 Februari 2023) - Menjelaskan bagaimana Earnin.com menggunakan QLDB untuk membangun layanan buku besar guna menyediakan aplikasi keuangan seluler modern dan berfitur lengkap kepada penggunanya.
- Ekspor dan analisis data jurnal Amazon QLDB menggunakanAWS Glue dan Amazon Athena (19 Desember 2022) — Membahas bagaimana Anda dapat menggunakan fitur ekspor di QLDB denganAWS Glue dan Athena untuk menyediakan kemampuan pelaporan dan analitis ke arsitektur berbasis buku besar Anda.
- Bagaimana Shinsegae International meningkatkan pengalaman pelanggan dan mencegah pemalsuan dengan Amazon QLDB (3 Agustus 2022) - Menjelaskan bagaimana Shinsegae International membangun layanan verifikasi keaslian digital menggunakan Amazon QLDB untuk memberi tahu pelanggan tentang keaslian barang mewah, dan menyediakan riwayat pembelian dan distribusi produk.
- BungKusit menggunakan teknologi Amazon QLDB dan ISV VeriDoc Global untuk meningkatkan pengalaman pelanggan dan agen pengiriman (29 April 2022) - Menunjukkan bagaimana satu perusahaan logistik, BungKusit, menggunakan QLDB untuk menerapkan platform terpusat dan transparan untuk komunikasi antar industri dengan log transaksi yang tidak dapat diubah dan dapat diverifikasi secara kriptografis.
- Gunakan Amazon QLDB sebagai penyimpanan nilai kunci yang tidak dapat diubah dengan REST API dan JSON (14 Februari 2022) — Memperkenalkan cara untuk bekerja dengan QLDB sebagai penyimpanan nilai kunci yang tidak dapat diubah dan menggunakan fitur audit melalui REST API.

- Membangun sistem perbankan inti dengan Amazon QLDB (21 Januari 2022) - Menunjukkan cara menggunakan QLDB untuk membangun sistem buku besar perbankan inti. Ini juga menunjukkan mengapa QLDB adalah fit-for-purpose database yang baik yang sesuai dengan kebutuhan industri jasa keuangan.
- Cara Specright menggunakan Amazon QLDB untuk membuat jaringan rantai pasokan yang dapat dilacak (21 Januari 2022) - Menunjukkan bagaimana Specright menggunakan QLDB untuk membuat jaringan rantai pasokan yang dapat dilacak yang memungkinkan merek grosir, pengecer, dan produsen untuk berbagi data rantai pasokan penting dan data spesifikasi pengemasan.
- Bagaimana FeMr Menghadirkan Data Medis yang Aman dan Dapat Diverifikasi secara Kriptografis dengan Amazon QLDB (23 Desember 2021) - Mengeksplorasi bagaimana Tim FeMr menggunakan QLDB dan layananAWS terkelola lainnya untuk memungkinkan upaya bantuan mereka.
- Pantau pola akses kueri Amazon QLDB (8 November 2021) — Menunjukkan cara mengaitkan transaksi QLDB dan pernyataan PartiQL yang dijalankan dalam transaksi dengan permintaan API yang diterima melalui Amazon API Gateway.
- Buat operasi CRUD sederhana dan aliran data di Amazon QLDB menggunakanAWS Lambda (28 September 2021) — Menunjukkan cara melakukan operasi CRUD (membuat, membaca, memperbarui, dan menghapus) pada QLDB menggunakanAWS Lambda fungsi.
- Verifikasi kriptografi dunia nyata dengan Amazon QLDB (26 Agustus 2021) - Membahas nilai verifikasi kriptografi dalam buku besar QLDB dalam konteks kasus penggunaan yang realistis.
- Verifikasi kondisi pengiriman dengan Accord Project dan Amazon QLDB: Bagian 1 , Bagian 2 (28 Juni 2021) - Membahas cara menerapkan teknologi kontrak hukum cerdas untuk memverifikasi kondisi pengiriman dengan Proyek Accord sumber terbuka dan QLDB.
- Streaming data Amazon QLDB melaluiAWS CDK (7 Juni 2021) — Menunjukkan cara menggunakanAWS Cloud Development Kit (AWS CDK) untuk mengatur QLDB, mengisi data QLDB menggunakanAWS Lambda fungsi, dan menyiapkan streaming QLDB untuk memberikan ketahanan data data buku besar.
- Demo kontrol akses berbutir halus di QLDB (1 Juni 2021) - Menunjukkan cara memulai dengan izin berbutir halusAWS Identity and Access Management (IAM) untuk buku besar QLDB.
- Pemrosesan Stream dengan DynamoDB Streams dan QLDB Streams (23 November 2020) - Memberikan perbandingan singkat antara aliran DynamoDB dan aliran QLDB, dan tips untuk memulainya.

- Streaming data dari Amazon QLDB ke OpenSearch (19 Agustus 2020) — Menjelaskan cara melakukan streaming data dari QLDB ke Amazon OpenSearch Service untuk mendukung pencarian teks kaya dan analisis hilir, seperti agregasi atau metrik di seluruh catatan.
- Cara saya mengalirkan data dari Amazon QLDB ke DynamoDB menggunakan Nodejs dalam waktu hampir nyata (7 Juli 2020) — Menjelaskan cara melakukan streaming data dari QLDB ke DynamoDB untuk mendukung latensi satu digit dan pertanyaan nilai kunci yang dapat diskalakan tanpa batas.
- Membangun antarmuka GraphQL ke Amazon QLDB denganAWS AppSync: Bagian 1 , Bagian 2 (4 Mei 2020) - Membahas cara mengintegrasikan QLDB danAWS AppSync menyediakan API serbaguna yang didukung GraphQL di atas buku besar QLDB.

# Media

AWSvideo

- AWSTutorial & Demo: QLDB ke Aurora Streaming (17 Maret 2023; 21 menit) - Video ini menjelaskan konsep dasar untuk menerapkan kemampuan streaming QLDB, dan menunjukkan cara mengatur streaming data dari QLDB ke DB hilir Amazon Aurora MySQL.
- ArcBlock: Memanfaatkan Amazon QLDB untuk Membangun Solusi Identitas Terdesentralisasi (31 Mei 2022; 4 menit) - ArcBlock berjalan melalui solusi identitas terdesentralisasi yang mereka bangun dengan menggunakan QLDB.
- AWSRE:Invent 2020: Menggunakan Amazon QLDB sebagai system-of-trust database untuk aplikasi bisnis inti (5 Februari 2021; 30 menit) - Pelajari bagaimana pengguna Amazon QLDB awal telah menerapkan properti unik basis data buku besar untuk asal data dan verifikasi kriptografi untuk mengimplementasikan sistem rekaman dengan integritas data bawaan.
- AWSRe: Invent 2020: Membangun aplikasi tanpa server dengan Amazon QLDB (5 Februari 2021; 28 menit) - Pelajari cara membuat, menguji, dan mengoptimalkan aplikasi tanpa server yang berfungsi penuh dengan menggabungkan Amazon QLDB dengan layanan sepertiAWS Lambda, Amazon Kinesis, dan Amazon DynamoDB.
- AWSRe: Invent 2020: Membangun aplikasi berbasis audit yang menjaga integritas data dengan Amazon QLDB (5 Februari 2021; 18 menit) - Sesi ini membahas masalah yang dapat diselesaikan Amazon QLDB, menjawab pertanyaan Anda tentang kapan dan mengapa Anda akan menggunakan basis data buku besar, dan membagikan kasus penggunaan dari pelanggan seperti Osano.

- Cara Menyimpan Log yang Tidak Dapat Diubah secara Terpusat menggunakan Amazon QLDB dengan Aplikasi .NET (7 Desember 2020; 10 menit) - Demonstrasi cara menggunakan QLDB dengan aplikasi .NET untuk menyimpan log yang tidak dapat diubah secara terpusat.
- Lokakarya Virtual: Membangun Sistem Rekaman Berbasis Buku Besar dengan QLDB dan Java - Pembicaraan TeknologiAWS Online (29 Juli 2020; 87 menit) - Lokakarya yang dipimpin instruktur yang berjalan melalui laboratorium Working With Ion Immersion Day untuk membangun program pemuat data menggunakan perpustakaan Amazon Ion dan driver QLDB untuk Jawa.
- Workshop Pengembang: Menggunakan QLDB Database Buku Besar yang Tidak DapatAWS Diubah di ABT Node (22 Juni 2020; 34 menit) - step-by-step Panduan tentang cara mengatur dan mengkonfigurasi QLDB di ABT Node. Ini juga menjelaskan cara menginstal dan mengkonfigurasi ArcBlock Blockchain Explorer dan Boarding Gate Blocklets untuk terhubung ke QLDB.
- AWSTech Talk: Perspektif Pelanggan tentang Membangun Aplikasi System-of-Record yang Dipicu Acara dengan Amazon QLDB (31 Maret 2020; 29 menit) - Pembicaraan teknologi oleh Matt Lewis - PahlawanAWS Data dan Kepala Arsitek Agen Lisensi Pengemudi dan Kendaraan (DVLA) di Inggris — yang menjelaskan kasus penggunaan DVLA untuk QLDB dan arsitektur berbasis acara aplikasi mereka.
- AWSRE: Invent 2019: Mengapa Anda membutuhkan basis data buku besar: BMW, DVLA, & Sage membahas kasus penggunaan (5 Desember 2019; 47 menit) - Presentasi oleh pelanggan BMW, organisasi pemerintah Inggris DVLA, dan Sage yang membahas alasan untuk menggunakan basis data buku besar dan membagikan kasus penggunaannya untuk QLDB.
- AWSRE: Invent 2019: Amazon QLDB: Penyelaman mendalam seorang insinyur tentang mengapa ini adalah pengubah permainan (5 Desember 2019; 50 menit) - Presentasi oleh Andrew Tertentu (AWSDistinguished Engineer) yang membahas arsitektur unik dan jurnal pertama QLDB, bersama dengan berbagai inovasinya. Ini termasuk hashing kriptografi, pohon Merkle, beberapa replikasi Availability Zone, dan dukungan PartiQL.
- Membangun Aplikasi dengan Amazon QLDB, Database Buku Besar Pertama-of-Jenisnya - Pembicaraan TeknologiAWS Online (19 November 2019; 51 menit) - Pembicaraan teknologi mendalam yang menjelaskan fitur unik QLDB dan spesifik tentang cara menggunakan fungsionalitas inti. Ini termasuk menanyakan riwayat lengkap data Anda, memverifikasi dokumen secara kriptografis, dan merancang model data.

Podcast

- Mengapa pelanggan memilih Amazon QLDB? (5 Juli 2020; 33 menit) - Diskusi yang menjelaskan definisi basis data buku besar, bagaimana perbedaannya dengan blockchain, dan bagaimana pelanggan menggunakannya saat ini.

# Sumber daya AWS umum

- Kelas & Lokakarya — Tautan ke kursus specialty dan berbasis peran, selain lab mandiri untuk membantu mempertajamAWS keterampilan Anda dan mendapatkan pengalaman praktis.
- AWSPusat Pengembang - Jelajahi tutorial, alat unduh, dan pelajari tentang acaraAWS pengembang.
- AWSAlat Pengembang — Tautan ke alat developer, SDK, toolkit IDE, dan alat baris perintah untuk mengembangkan dan mengelolaAWS aplikasi.
- Memulai Pusat Sumber Daya — Pelajari cara mengaturAkun AWS, bergabung denganAWS komunitas, dan meluncurkan aplikasi pertama Anda.
- Tutorial Hands-On - Ikuti step-by-step tutorial untuk meluncurkan aplikasi pertama AndaAWS.
- AWSLaporan Resmi — Tautan ke daftar lengkapAWS laporan resmi teknis, yang mencakup topik seperti arsitektur, keamanan, dan ekonomi dan ditulis oleh ArsitekAWS Solusi atau ahli teknis lainnya.
- AWS SupportCenter — Hub untuk membuat dan mengelolaAWS Support kasus Anda. Juga mencakup tautan ke sumber daya yang bermanfaat lainnya, seperti forum, FAQ teknis, status kondisi layanan, dan AWS Trusted Advisor.
- AWS Support— Halaman web utama untuk informasi tentangAWS Support, saluran dukungan respons cepat cepat untuk membantu Anda membangun dan menjalankan aplikasi di cloud. one-on-one
- Kontak Kami – Titik kontak pusat untuk pertanyaan tentang tagihan AWS, akun, peristiwa, penyalahgunaan, dan masalah lainnya.
- AWSPersyaratan Situs – Informasi detail tentang hak cipta dan merek dagang kami; akun, lisensi, dan akses situs Anda; serta topik lainnya.

# Riwayat rilis Amazon QLDB

Tabel berikut menjelaskan tentang perubahan penting dalam setiap perilisan Amazon QLDB dan pembaruan terkait dalam Panduan Developer Amazon QLDB. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan ke umpan RSS.

- Versi API: 2019-01-02
- Update dokumentasi terbaru: 3 Januari 2023

| Perubahan | Deskripsi | Tanggal |
| --- | --- | --- |
| Panduan IAM yang diperbarui | Panduan yang diperbarui untuk menyelaraskan dengan praktik terbaik IAM. Untuk informasi selengkapnya, lihat Praktik terbaik keamanan dalam IAM. | 3 Januari 2023 |
| Pembaruan ke kebijakanAWS terkelola | Amazon QLDB memperbarui kebijakanAWS terkelola yang adaAmazonQLDBFullAccess danAmazonQLDBConsoleFullAccess . Kebijakan ini memiliki izin baru untuk memungkinkan prinsipal untuk menyuntin g revisi dokumen dengan menggunakan prosedur tersimpan PartiQL. Untuk informasi selengkapnya, lihat kebijakanAWS terkelola untuk Amazon QLDB. | 04 November 2022 |
| redaksi data | Amazon QLDB sekarang mendukung prosedur tersimpanREDACT_RE | 3 November 2022 |

| | | |
|---|---|---|
| | `VISION` PartiQL untuk buku besar yang dibuat pada atau setelah 22 Juli 2021. Dengan menggunakan prosedur tersimpan ini, Anda dapat menghapus revisi dokumen yang tidak aktif secara permanen dalam riwayat dan tetap mempertahankan integritas data keseluruh an buku besar Anda. Untuk informasi selengkapnya, lihat Mengedit revisi dokumen. | |
| Node.js sopir v3 | Driver Amazon QLDB untuk versi 3.0 sekarang tersedia secara umum. Node.js Versi ini memperkenalkan dukungan untukAWS SDK for JavaScrip t v3. Untuk catatan rilis, lihat awslabs/ GitHub repositor iamazon-qldb-driver-nodejs. | 26 September 2022 |
| Pergi sopir v3 | Driver Amazon QLDB untuk versi versi 3.0 sekarang tersedia secara umum. Versi ini memperkenalkan dukungan untukAWS SDK for Go v2. Untuk catatan rilis, lihat awslabs/ GitHub repositor iamazon-qldb-driver-go. | 11 Agustus 2022 |

| | | |
|---|---|---|
| Editor kueri PartiQL baru | Editor kueri PartiQL baru di konsol Amazon QLDB sekarang tersedia secara umum. Editor QLDB PartiQL baru menyediakan antarmuka yang ditingkatkan untuk authoring query, debugging transaksi, dan mengeksplorasi hasil. Untuk informasi tentang membuka dan menggunak an editor, lihat Mengakses Amazon QLDB menggunakan konsol. | 22 Juni 2022 |
| Ion objek mapper untuk driver .NET | Driver Amazon QLDB untuk .NET versi 1.3 memperkenalkan dukungan untuk pemetaan objek Ion. Fitur ini memungkinkan Anda sepenuhnya melewati kebutuhan untuk mengonver si secara manual antara jenis Amazon Ion dan tipe C # asli. Untuk riwayat perubahan penuh dari pemetaan objek Ion, lihat file ChangeLog.md di GitHub repositoriamzn/ion-object-mapper-dotn et . | 19 Januari 2022 |

| | | |
|---|---|---|
| Format ekspor jurnal JSON | Amazon QLDB sekarang mendukung format keluaran JSON Lines untuk ekspor jurnal. Untuk informasi selengkapnya, lihat Mengekspor data jurnal dari Amazon QLDB. | 21 Desember 2021 |
| Pemecahan Masalah Amazon QLDB | Menambahkan topik Pemecahan Masalah baru yang memberikan panduan untuk daftar agregat kesalahan umum yang mungkin Anda temui saat menggunakan Amazon QLDB. | 8 Desember 2021 |
| Peluncuran Wilayah Baru | Amazon QLDB kini tersedia di Wilayah Canada (Central). Untuk daftar lengkap Wilayah yang tersedia, lihat titik akhir dan kuota Amazon QLDB di bagian Referensi Umum Amazon Web Services. | 11 November 2021 |
| Pencegahan wakil bingung lintas layanan | Amazon QLDB sekarang mendukung penggunaa `naws:SourceArn` dan kunci konteks kondisi`aws:SourceAccount` global dalam kebijakan sumber daya IAM untuk mencegah masalah wakil yang membingungkan. Untuk informasi lebih lanjut, lihat Cross-service bingung wakil pencegahan. | 8 November 2021 |

| | | |
|---|---|---|
| Cangkang Amazon QLDB v2 | Versi 2.0 dari shell Amazon QLDB, yang ditulis dalam Rust, sekarang tersedia secara umum. Untuk catatan rilis, lihat awslabs/ GitHub repositoriamazon-qldb-shell. | 14 Oktober 2021 |
| Pembaruan ke kebijakanAWS terkelola | Amazon QLDB memperbarui kebijakanAWS terkelola yang ada`AmazonQLDBFullAccess` dan`AmazonQLDBConsoleFullAccess` . Kebijakan ini memiliki izin yang baru ditambahkan untuk memungkinkan prinsipal meneruskan sumber daya peran IAM apa pun di akun Anda ke layanan QLDB. Ini diperlukan untuk semua permintaan ekspor dan aliran jurnal. Untuk informasi selengkapnya, lihat kebijakan AWS terkelola untuk Amazon QLDB. | 2 September 2021 |
| AWS KMSKunci yang dikelola pelanggan | Amazon QLDB sekarang mendukung enkripsi saat istirahat menggunakan kunci yang dikelola pelanggan inAWS Key Management Service (AWS KMS) untuk sumber daya buku besar baru. Untuk informasi selengkapnya, lihat Enkripsi at rest di Amazon QLDB. | 22 Juli 2021 |

| | | |
|---|---|---|
| Pembaruan ke kebijakanAWS terkelola | Amazon QLDB memperbarui kebijakanAWS terkelola yang adaAmazonQLDBReadOnly untuk menghapusqldb:GetBlock tindakan duplikat yang sebelumnya terdaftar dua kali. Untuk informasi selengkapnya, lihat kebijakanAWS terkelola untuk Amazon QLDB. | 1 Juli 2021 |
| Peluncuran Wilayah Baru | Amazon QLDB kini tersedia di Wilayah Eropa (London). Untuk daftar lengkap Wilayah yang tersedia, lihat titik akhir dan kuota Amazon QLDB di bagian Referensi Umum Amazon Web Services. | 24 Juni 2021 |
| Pembaruan ke kebijakanAWS terkelola | Amazon QLDB memperbarui kebijakanAWS terkelola yang adaAmazonQLDBFullAccess danAmazonQLDBConsoleFullAccess . Kebijakan ini memiliki izin yang baru ditambahkan untuk memungkinkan prinsipal memperbarui mode izin di semua buku besar, dan untuk menjalankan semua perintah PartiQL di semua buku besarSTANDARD izin. Untuk informasi selengkapnya, lihat kebijakanAWS terkelola untuk Amazon QLDB. | 27 Mei 2021 |

| | | |
|---|---|---|
| Mode izin standar | Amazon QLDB sekarang mendukung mode`STANDARD` izin untuk sumber daya buku besar. Dengan mode perizinan standar, Anda dapat mengontrol akses dengan rincian yang lebih halus untuk buku, dan perintah PartiQL. Untuk informasi selengkapnya, lihat Memulai dengan mode izin standar. | 27 Mei 2021 |
| Statistik pernyataan PartiQL | Fitur statistik pernyataan PartiQL sekarang tersedia di editor Query di konsol Amazon QLDB. Untuk informasi selengkapnya, lihat Mendapatkan statistik pernyataan. | 24 Mei 2021 |
| Tutorial: Memverifikasi data menggunakanAWS SDK | Menambahkan step-by-s tep tutorial dengan contoh kode yang menunjukkan cara memverifikasi hash revisi dan hash blok di Amazon QLDB menggunakan QLDB API melaluiAWS SDK. Untuk mempelajari lebih lanjut, lihat Tutorial: Memverifikasi data menggunakanAWS SDK. | 6 Mei 2021 |

| | | |
|---|---|---|
| .NET driver v1.2 | Driver Amazon QLDB untuk versi 1.1.2 sekarang tersedia secara umum. Versi ini memperkenalkan API asinkron. Untuk catatan rilis, lihat awslabs/ GitHub repositoriamazon-qldb-driver-dotnet. | 1 April 2021 |
| `DROP INDEX`Pernyataan PartiQL | PartiQL di Amazon QLDB sekarang mendukung pernyataan DROP INDEX. Untuk informasi selengkapnya, lihat Menjatuhkan indeks. | 3 Maret 2021 |
| Statistik pernyataan PartiQL | Fitur statistik pernyataan PartiQL sekarang tersedia dalam versi terbaru driver Amazon QLDB untuk semua bahasa yang didukung, termasuk .NET, Go, dan Python. Untuk informasi selengkapnya, lihat Mendapatkan statistik pernyataan. | 25 Februari 2021 |
| TypeScript tutorial mulai cepat | Menambahkan contoh TypeScript kode dalam tutorial Quick start untuk driver Amazon QLDB untuk Node.js. | 28 Desember 2020 |

| | | |
|---|---|---|
| Statistik pernyataan PartiQL | Versi terbaru driver Amazon QLDB untuk Java dan Node.js sekarang menyediakan statistik eksekusi pernyataan yang dapat membantu Anda menjalankan pernyataan PartiQL yang lebih efisien. Untuk informasi selengkapnya, lihat Mendapatkan statistik pernyataan. | 22 Desember 2020 |
| Referensi buku masak driver dan tutorial mulai cepat | Ditambahkan referensi buku masak untuk driver Go dan Node.js, tutorial mulai cepat untuk driver Java dan Python, dan panduan untuk bekerja dengan Amazon Ion di QLDB. Untuk informasi selengkapnya, lihat Memulai dengan driver Amazon QLDB. | 24 November 2020 |
| Amazon QLDB QLDB | Versi 1.1 shell Amazon QLDB sekarang tersedia secara umum. Untuk catatan rilis, lihat awslabs/ GitHub repositor iamazon-qldb-shell. | 26 Oktober 2020 |

| | | |
|---|---|---|
| Driver QLDB Amazon untuk Go | Driver Amazon QLDB untuk Go sekarang tersedia secara umum. Driver ini open source GitHub dan memungkinkan Anda untuk menggunak anAWS SDK for Go untuk berinteraksi dengan API data transaksional QLDB. Untuk catatan rilis, lihat awslabs/ GitHub repositoriamazon-qldb-driver-go. | 20 Oktober 2020 |
| Rekomendasi pengaturan driver Node.js | Menambahkan bagian baru yang menyediakan rekomendasi Pengaturan untuk driver Amazon QLDB untuk Node.js, termasuk cara mengurangi latensi dengan menggunakan kembali koneksi dengan keep-alive. | 16 Oktober 2020 |
| Pembuatan indeks pada tabel yang tidak kosong | Amazon QLDB sekarang mendukung pembuatan indeks pada tabel yang tidak kosong. Untuk informasi selengkapnya, lihat Mengelola indeks. | 30 September 2020 |
| Mengoptimalkan kinerja kueri | Menambahkan bagian baru yang menjelaskan batasan kueri di Amazon QLDB dan memberikan panduan untuk Mengoptimalkan kinerja kueri dengan batasan ini. | 18 September 2020 |

| | | |
|---|---|---|
| Referensi buku masak untuk driver Java | Menambahkan referensi Cookbook yang menunjukkan contoh kode kasus penggunaan umum untuk driver Amazon QLDB untuk Java. | 3 September 2020 |
| Manajemen sesi dengan pengemudi | Menambahkan bagian baru yang memberikan ikhtisar manajemen Sesi dengan driver di Amazon QLDB, dan menjelaskan bagaimana driver QLDB menangani sesi saat menjalankan transaksi data. | 1 September 2020 |
| Java driver v2.0 | Driver Amazon QLDB untuk versi 2.0 sekarang tersedia secara umum. Untuk catatan rilis, lihat awslabs/ GitHub repositoriamazon-qldb-driver-java. | 28 Agustus 2020 |
| Node.js sopir v2.0 | Driver Amazon QLDB untuk versi 2.0 sekarang tersedia secara umum. Node.js Untuk catatan rilis, lihat awslabs/ GitHub repositoriamazon-qldb-driver-nodejs. | 27 Agustus 2020 |
| Informasi terkait | Menambahkan topik baru yang berisi tautan untuk informasi terkait dan sumber daya tambahan untuk membantu Anda memahami dan bekerja dengan Amazon QLDB. | 24 Agustus 2020 |

| | | |
|---|---|---|
| Python sopir v3.0 | Driver Amazon QLDB untuk Python versi 3.0 sekarang tersedia secara umum. Untuk catatan rilis, lihat awslabs/ GitHub repositoriamazon-q ldb-driver-python. | 20 Agustus 2020 |
| Driver QLDB Amazon untuk Go | Rilis pratinjau driver QLDB Amazon untuk Go sekarang tersedia. Driver ini memungkin kan Anda untuk menggunak anAWS SDK for Go untuk berinteraksi dengan API data transaksional QLDB. Untuk informasi selengkapnya, lihat driver Amazon QLDB untuk Go (pratinjau). | 6 Agustus 2020 |
| Referensi buku masak untuk driver Python | Menambahkan referensi Cookbook yang menunjukkan contoh kode kasus penggunaa n umum untuk driver Amazon QLDB untuk Python. | 24 Juli 2020 |
| ID unik di Amazon QLDB | Menambahkan bagian baru yang menjelaskan properti dan pedoman penggunaan ID Unik di Amazon QLDB. | 9 Juli 2020 |

| | | |
|---|---|---|
| AWS CloudFormationsumber daya untuk aliran jurnal | Amazon QLDB sekarang mendukung sumber daya untuk membuat aliran jurnal menggunakanAWS CloudFormation template. Untuk informasi selengkap nya, lihat AWS::QLDB ::Streamsumber daya di PanduanAWS CloudFormation Pengguna. | 9 Juli 2020 |
| Referensi buku masak untuk driver .NET | Menambahkan referensi Cookbook yang menunjukkan contoh kode kasus penggunaa n umum untuk driver Amazon QLDB untuk .NET. | 1 Juli 2020 |
| Driver Amazon QLDB untuk .NET | Driver Amazon QLDB untuk .NET sekarang tersedia secara umum. Driver ini open source GitHub dan memungkinkan Anda untuk menggunakanAWS SDK for .NET untuk berinteraksi dengan API data transaksional QLDB. Untuk catatan rilis, lihat awslabs/ GitHub repositor iamazon-qldb-driver-dotnet. | 26 Juni 2020 |

| | | |
|---|---|---|
| Driver QLDB Amazon untuk Node.js | Driver Amazon QLDB untuk Node.js sekarang tersedia secara umum. Driver ini aktif open source GitHub dan memungkinkan Anda untuk menggunakanAWS SDK untuk JavaScript di Node.js untuk berinteraksi dengan API data transaksional QLDB. Untuk catatan rilis, lihat awslabs/amazon-qldb-driver-nodejs GitHub repositori. | 5 Juni 2020 |
| Aliran jurnal Amazon QLDB | Amazon QLDB sekarang memungkinkan Anda membuat stream yang menangkap setiap revisi dokumen yang berkomitm en pada jurnal Anda dan mengirimkan data ini ke Amazon Kinesis Data Streams dalam waktu yang hampir nyata. Untuk informasi selengkapnya, lihat Streaming data jurnal dari Amazon QLDB. | 19 Mei 2020 |

| | | |
|---|---|---|
| Contoh kode Amazon Ion | Menambahkan contoh kode yang mengkueri dan memproses data Amazon Ion dalam buku besar Amazon QLDB dengan menggunakan driver QLDB untuk Java, Node.js, dan Python. Untuk informasi selengkapnya, lihat contoh kode ion di Amazon QLDB. | 12 Mei 2020 |
| Model konkurensi dan konten jurnal | Memperluas topik model konkurensi Amazon QLDB untuk menambahkan informasi tentang penggunaan indeks untuk membatasi konflik kontrol konkurensi (OCC) yang optimis. Menambahkan bagian baru yang menjelaskan konten Jurnal di Amazon QLDB. | 4 Mei 2020 |
| Panduan mulai cepat untuk driver Node.js | Menambahkan panduan mulai cepat untuk driver Amazon QLDB untuk Node.js. | Selasa, 01 Mei 2020 |
| Cangkang Amazon QLDB | Shell Amazon QLDB sekarang tersedia secara umum. Shell ini open source GitHub dan menyediakan antarmuka baris perintah yang memungkinkan Anda untuk mengeksekusi pernyataan PartiQL pada data buku besar. Untuk catatan rilis, lihat awslabs/amazon-qldb-shell GitHub repositori. | 20 April 2020 |

| | | |
|---|---|---|
| Sertifikasi kepatuhan | Amazon QLDB sekarang disertifikasi untuk programAWS kepatuhan, termasuk HIPAA dan ISO. Untuk informasi selengkapnya, lihat Validasi kepatuhan untuk Amazon QLDB. | 3 April 2020 |
| Rekomendasi pengemudi yang diperluas | Memperluas bagian rekomendasi driver QLDB Amazon untuk membuatnya berlaku untuk semua bahasa pemrograman yang didukung. | 2 April 2020 |
| Cangkang Amazon QLDB | Rilis pratinjau shell Amazon QLDB sekarang tersedia. Shell ini menyediakan antarmuka baris perintah yang memungkinkan Anda untuk mengeksekusi pernyataan PartiQL pada data buku besar. Untuk informasi selengkapnya, lihat Mengakses Amazon QLDB menggunakan shell QLDB (hanya API data) (pratinjau). | 23 Maret 2020 |
| Driver Java v1.1 | Driver Amazon QLDB untuk versi Java 1.1 sekarang tersedia secara umum. Untuk catatan rilis, lihat awslabs/amazon-qldb-driver-java GitHub repositori. | 20 Maret 2020 |

| | | |
|---|---|---|
| Driver Amazon QLDB untuk .NET | Amazon QLDB sekarang menyediakan rilis pratinjau driver .NET. Driver ini memungkinkan Anda untuk menggunakanAWS SDK for .NET untuk berinteraksi dengan API data transaksi onal QLDB. Untuk informasi selengkapnya, lihat driver Amazon QLDB untuk .NET (pratinjau). | 13 Maret 2020 |
| Driver Amazon QLDB untuk Python | Driver Amazon QLDB untuk Python sekarang tersedia secara umum. Driver ini open source GitHub dan memungkinkan Anda untuk menggunakanAWS SDK for Python (Boto3) untuk berintera ksi dengan API data transaksi onal QLDB. Untuk catatan rilis, lihat awslabs/amazon-qldb-driver-python GitHub repositor i. | 11 Maret 2020 |

| | | |
|---|---|---|
| `UNDROP TABLE`Pernyataan PartiQL dan DML-bersarang | PartiQL di Amazon QLDB sekarang mendukung pernyataan bahasa manipulasi data (DHTML) di mana koleksi bersarang ditentukan sebagai sumber dalam`FROM` klausa. Untuk informasi lebih lanjut, lihat pernyataan FROM di referensi PartiQL. QLDB PartiQL juga mendukung pernyataan UNDROP TABLE. Untuk informasi selengkapnya, lihat tabel undropping. | 26 Februari 2020 |
| Topik intro yang diperluas | Memperluas pengantar Apa itu Amazon QLDB? topik untuk menyertakan bagian baru Ikhtisar Amazon QLDB dan Dari relasional ke buku besar. | 24 Januari 2020 |
| Referensi PartiQL untuk fungsi yang didukung | Memperluas referensi Amazon QLDB PartiQL untuk menyertakan informasi penggunaan terperinci tentang fungsi SQL yang didukung. Untuk informasi lebih lanjut, lihat fungsi PartiQL. | 2 Januari 2020 |
| Rekomendasi pengemudi dan kesalahan umum | Menambahkan bagian baru yang menjelaskan rekomendasi Driver untuk driver Amazon QLDB untuk kesalahan Java dan Common untuk semua bahasa driver. | 2 Januari 2020 |

| | | |
|---|---|---|
| Peluncuran Wilayah Baru | Amazon QLDB kini tersedia di Wilayah Asia Pacific (Seoul), Singapore), Asia Pacific (Sydney). Untuk daftar lengkap Wilayah yang tersedia, lihat titik akhir dan kuota Amazon QLDB di bagian Referensi Umum Amazon Web Services. | 19 November 2019 |
| Driver QLDB Amazon untuk Node.js | Amazon QLDB sekarang menyediakan rilis pratinjau driver Node.js. Driver ini memungkinkan Anda untuk menggunakanAWS SDK untuk JavaScript di Node.js untuk berinteraksi dengan API data transaksional QLDB. Untuk informasi selengkapnya, lihat driver Amazon QLDB untuk Node.js (pratinjau). | 13 November 2019 |
| Driver Amazon QLDB untuk Python | Amazon QLDB sekarang menyediakan rilis pratinjau driver Python. Driver ini memungkinkan Anda untuk menggunakanAWS SDK for Python (Boto3) untuk berinteraksi dengan API data transaksional QLDB. Untuk informasi selengkapnya, lihat driver Amazon QLDB untuk Python (pratinjau). | 29 Oktober 2019 |

| | | |
|---|---|---|
| Rilis publik | Ini adalah rilis publik awal Amazon QLDB. Rilis ini mencakup Panduan Pengembang dan referensi API manajemen buku besar terintegrasi. | 10 September 2019 |

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.