aws

開發人員指南

Amazon Quantum Ledger Database (Amazon QLDB)

Amazon Quantum Ledger Database (Amazon QLDB): 開發人員指南

Table of Contents

什麼是 Amazon QLDB？

Amazon Quantum QLDB 是一個全受管分類帳資料庫，提供透明、不可變且以密碼編譯方式驗證的交易日誌，這些交易日誌為集中的受信任授權單位所擁有。您可以使用 Amazon QLDB 追蹤所有的應用程式資料變更，維護隨著時間變更的完整且可驗證的歷史記錄。要進一步了解 Amazon Web Services 上可用的各種資料庫選項，請參閱選擇適合您組織的資料庫（詳見）AWS。

分類帳通常用於記錄組織中經濟與財務活動的歷史記錄。許多組織會建置具有類似分類帳功能的應用程式，因為他們想要維護其應用程式資料的準確歷程記錄。例如，他們可能想要追蹤銀行交易中貸方與借方的歷史記錄、核對保險索賠的資料歷程，或追蹤供應鏈網路中料號的移動情況。總帳應用程式通常是使用關聯式資料庫中建立的自訂稽核表或稽核追蹤來實作。

Amazon QLDB 是一種新的資料庫類別，可協助您不再需要從事複雜的開發工作來建立自己的類似總帳的應用程式。使用 QLDB 時，您的資料變更歷程記錄是不可變的，無法就地覆寫或變更。使用密碼編譯，您可以驗證應用程序的數據沒有意外更改。QLDB 使用不可變的事務日誌，稱為日誌。該日誌是僅附加的，由一組包含您提交的數據的已排序和散列鏈式塊組成。

Amazon QLDB

有關 Amazon QLDB 的概觀以及它如何為您帶來好處，請觀看以下的 QLDB 概觀視頻 YouTube。

Amazon QLDB 的 Amazon QLDB

使用 Amazon QLDB 後，您只需按實際用量付費，而沒有最低收費或強制性的服務。您只需為分類帳資料庫所耗用的資源付費，而且不需預先佈建。

如需詳細資訊，請參閱 Amazon QLDB Amazon QLDB 定價。

QLDB 入門

建議您一開始先閱讀下列主題：

- Amazon QLDB— 要取得 QLDB 的高階概觀
- 亞馬遜 QLDB 中的核心概念和術語— 學習基本的 QLDB 概念和術語。
- 訪問 Amazon QLDB— 瞭解如何使用AWS Management Console、API 或AWS Command Line Interface (AWS CLI) 存取 QLDB。

- Amazon QLDB 如何與 IAM 合作— 瞭解如何使用AWS Identity and Access Management (IAM) 控制對 QLDB 的存取權限。

若要快速開始使用 QLDB 主控台，請參閱Amazon QLDB 主控台。

若要瞭解如何使用AWS提供的驅動程式使用 QLDB 進行開發，請參閱開始使用 Amazon QLDB 驅動程式。

Amazon QLDB

下列各節提供 Amazon QLDB 服務元件及其互動方式之高階概觀。

主題

- 第一期刊
- 固定
- 可以加密驗證
- 類似 SQL 和文檔靈活
- 開源開發者
- 無伺服器和高可用性
- 企業級

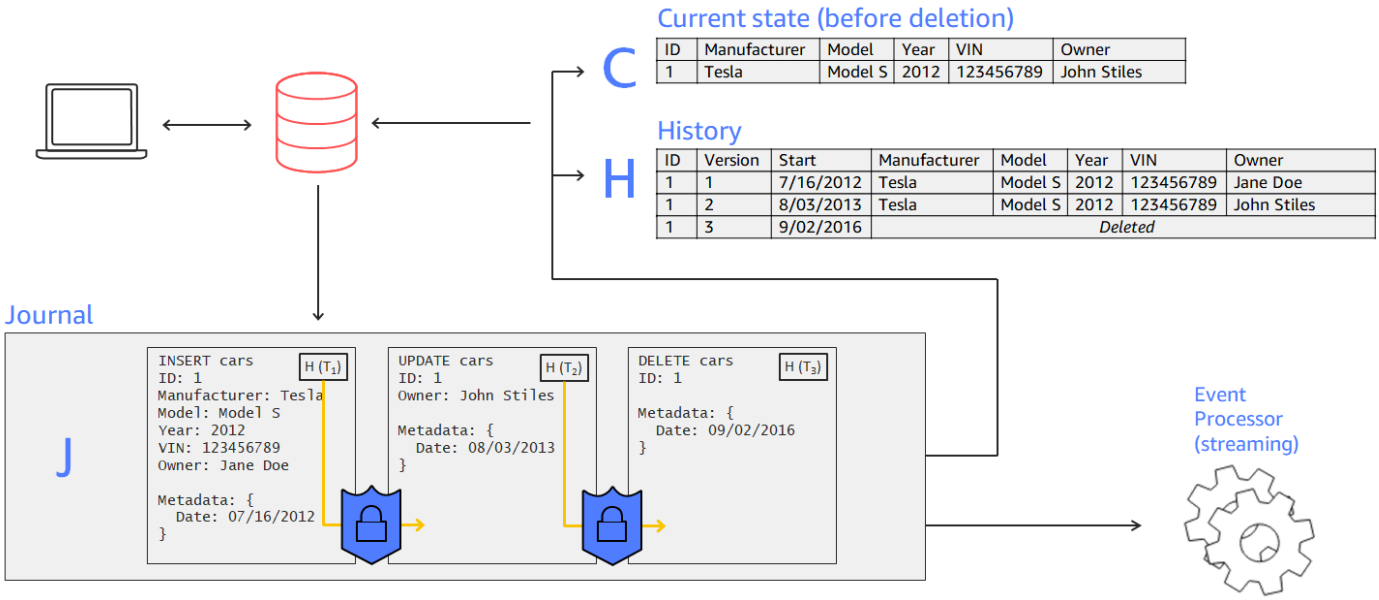
第一期刊

在傳統的數據庫體系結構，你通常寫在表中的數據作為一個事務的一部分。交易記錄 (通常是內部實作) 會記錄所有的交易，以及它們所做的資料庫修改。交易記錄檔是資料庫的重要元件。在發生系統故障、嚴重損壞修復或資料複寫時，您需要記錄才能重新顯示交易。不過，資料庫交易記錄檔不是不可變的，而且不是為了提供使用者直接且輕鬆的存取而設計。

在亞馬遜 QLDB 中，日誌是資料庫的核心。在結構上與交易記錄檔類似，日誌是不可變的、僅附加的資料結構，可儲存應用程式資料以及相關聯的中繼資料。所有寫入交易 (包括更新和刪除) 都會先確認至日誌。

QLDB 會將分類帳資料實現為可查詢的使用者定義表格，藉此使用分錄來判斷分類帳資料的目前狀態。這些表格也提供所有交易資料的可存取歷史記錄，包括文件修訂版和中繼資料。此外，該日誌還處理分類帳資料的並行、排序、密碼編譯驗證和可用性。

下圖說明 QLDB 日誌的架構。



- 在此範例中，應用程式會連線至分類帳，並執行將文件插入、更新及刪除名為的表格中的交易`cars`。
- 資料會先以排序順序寫入日誌。
- 然後將數據實現到具有內置視圖的表中。這些視圖使您可以查詢 Car 的當前狀態和完整歷史記錄，每個版本都分配了一個版本號。
- 您也可以直接從日誌匯出或串流資料。

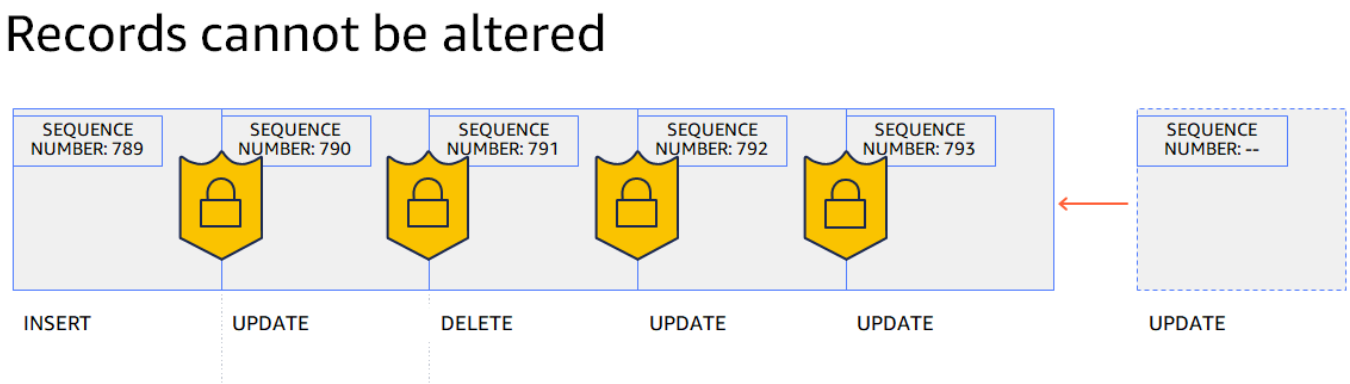
固定

由於 QLDB 日誌是僅附加的，因此會保留無法修改或覆寫資料的所有變更的完整記錄。沒有 API 或其他方法可以更改任何已提交的數據。此分錄結構可讓您存取並查詢分類帳的完整歷史記錄。

> **Note**
>
> QLDB 支援的不變性的唯一例外是資料編輯。使用此功能，您可以遵守法規，例如歐盟的一般資料保護規範 (GDPR) 和加州消費者隱私法 (CCPA)。
> QLDB 提供編輯操作，可讓您永久刪除表格歷程記錄中非使用中的文件修訂版本。此作業只會刪除指定修訂版本中的使用者資料，並保持分錄順序與文件中繼資料不變。這樣可以維護分類帳的整體資料完整性。如需詳細資訊，請參閱編輯文件修訂版本。

QLDB 會將一個區塊寫入交易中的日誌。每個區塊都包含項目物件，這些物件代表您插入、更新和刪除的文件，以及您執行來認可它們的陳述式。這些塊經過排序和哈希鏈接，以確保數據的完整性。

下圖說明此分錄結構。

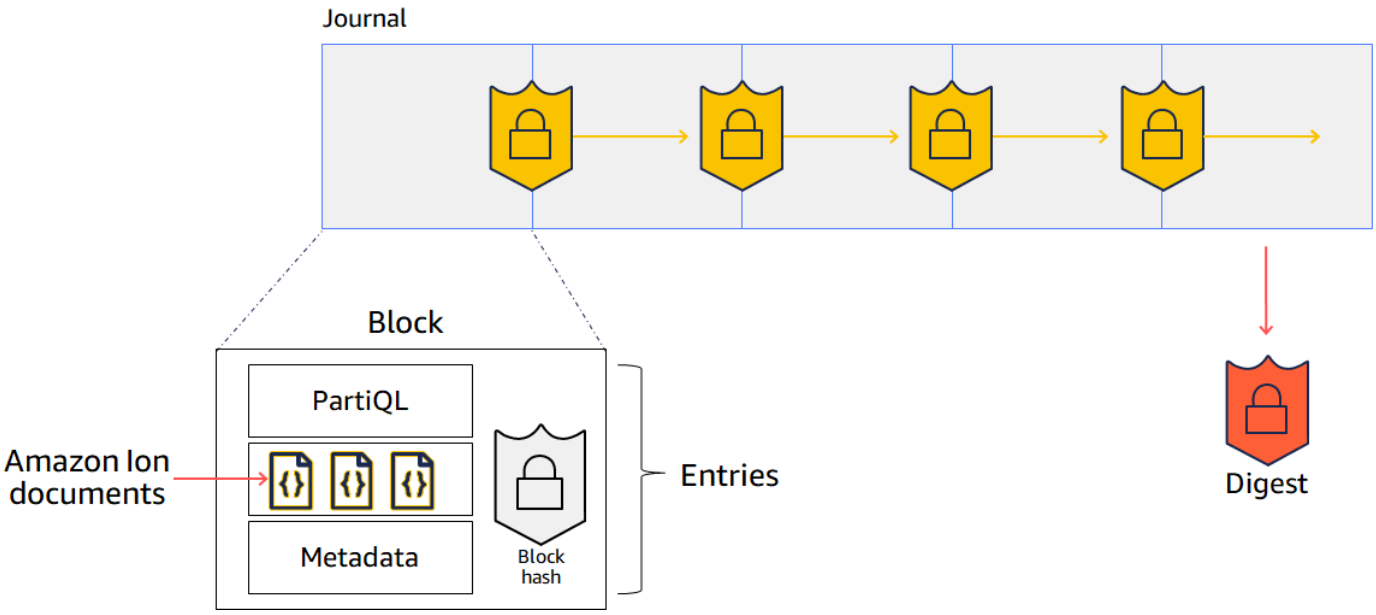


該圖表顯示交易作為哈希鏈接進行驗證的塊認可到日誌。每個塊都有一個序列號來指定其地址。

可以加密驗證

日誌塊與加密哈希技術進行排序並鏈接在一起，類似於區塊鏈。QLDB 使用日誌的雜湊鏈來提供使用密碼編譯驗證方法的交易資料完整性。使用摘要（表示截至某個時間點日誌的完整散列鏈的哈希值）和 Merkle 審核證明（一種證明二進制哈希樹中任何節點有效性的機制），您可以驗證隨時對數據沒有意外更改。

下圖顯示在某個時間點涵蓋日誌的完整雜湊鏈的摘要。



在此圖中，日誌區塊會使用 SHA-256 密碼編譯雜湊函式進行雜湊處理，並依序鏈結至後續區塊。每個區塊都包含包含在交易中執行的資料文件、中繼資料和 PartiQL 陳述式的項目。

如需詳細資訊，請參閱Amazon QLDB 中的數據驗證。

類似 SQL 和文檔靈活

QLDB 使用 PartiQL 作為其查詢語言，而亞馬遜離子作為其文件導向的資料模型。PartiQL 是一種開放原始碼、SQL 相容的查詢語言，已經擴充為與 Ion 搭配使用。使用 PartiQL，您可以使用熟悉的 SQL 運算子插入、查詢和管理資料。當您查詢平面文件時，語法與使用 SQL 查詢關聯式資料表的語法相同。若要進一步了解 PartiQL 的 QLDB 實作方式，請參閱《》Amazon QLDB 參考。

亞馬遜離子是 JSON 的超集。Ion 是一種開放原始碼、以文件為基礎的資料格式，可讓您靈活地儲存和處理結構化、半結構化和巢狀資料。若要進一步了解 QLDB 中的離子，請參閱《》亞馬遜 QLDB 中的亞馬遜離子數據格式參考。

如需傳統關聯式資料庫中核心元件和功能與 QLDB 的高階比較，請參閱從關聯式到分類帳。

開源開發者

為了簡化應用程式開發，QLDB 提供各種程式語言的開放原始碼驅動程式。您可以在分類帳上執行 PartiQL 陳述式並處理這些陳述式的結果，使用這些驅動程式與交易資料 API 互動。如需目前支援之驅動程式語言的資訊和自學課程，請參閱〈〉開始使用 Amazon QLDB 驅動程式。

Amazon Ion 也提供可為您處理 Ion 資料的用戶端程式庫。如需處理 Ion 資料的開發人員指南和程式碼範例，請參閱上的 Amazon Ion 文件 GitHub。

無伺服器和高可用性

QLDB 是全受管、無伺服器且高可用性。服務會自動擴展以支援應用程式的需求，而且您不需要佈建執行個體或容量。您的資料的多個副本會在可用區域內複寫，並在AWS 區域.

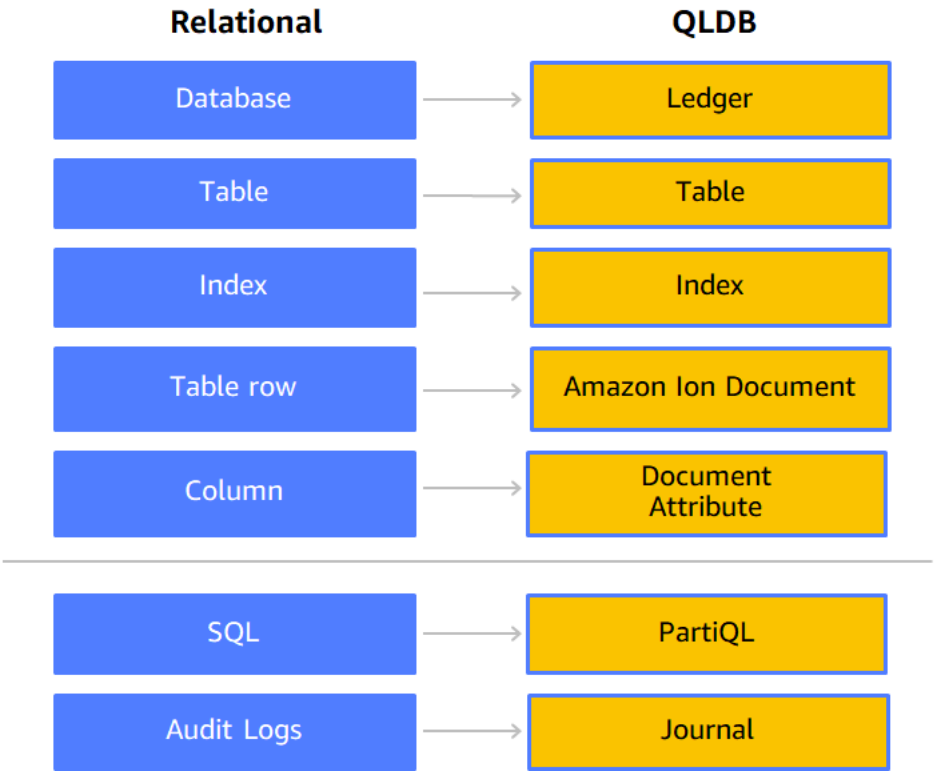
企業級

QLDB 交易完全符合不可分割性、一致性、一致性、一致性、一致性、一致性、一致性、一致性、一致性、QLDB 使用樂觀並發控制（OCC），並且交易以完全序列化（最高級別的隔離）進行操作。這意味著沒有看到幻象讀取，骯髒讀取，寫入歪斜或其他類似並發問題的風險。如需詳細資訊，請參閱Amazon QLLDB 並行模型。

從關聯式到分類帳

如果您是應用程序開發人員，則可能有使用關係數據庫管理系統（RDBMS）和結構化查詢語言（SQL）的一些經驗。開始使用 Amazon QLDB 後，您可能會發現許多相似點。當您進入更進階的主題時，您也會遇到 QLDB 建立在 RDBMS 基礎上的強大新功能。本節說明常見的資料庫元件和操作，並比較和比對方式及其相對應的 QLDB。

下圖顯示了傳統的 RDBMS 和亞馬遜 QLDB 之間的核心元件的映射結構。



下表顯示了傳統的 RDBMS 和 QLDB 之間的內置操作功能的主要高級相似性和差異。

操作	RDBMS	QLDB
建立資料表	CREATE TABLE定義所有列名和數據類型的語句	CREATE TABLE未定義任何資料表屬性或資料類型的陳述式，以允許無結構描述和開啟內容
建立索引	CREATE INDEX 陳述式	CREATE INDEX表格上任何頂層欄位的陳述式
插入資料	INSERT陳述式，指定新資料列或元組內的值，此陳述式會依附於資料表所定義的結構描述	INSERT以任何有效 Amazon Ion 格式指定新文件內的值的陳述式，不論表格中的現有文件為何

操作	RDBMS	QLDB
查詢資料	SELECT-FROM-WHERE 陳述式	SELECT-FROM-WHERE 查詢平面文件時，使用與 SQL 相同語法的陳述式
更新資料	UPDATE-SET-WHERE 陳述式	UPDATE-SET-WHERE 更新平面文檔時的語法與 SQL 相同的語句
刪除資料	DELETE-FROM-WHERE 陳述式	DELETE-FROM-WHERE 刪除平面文檔時的語法與 SQL 相同的語句
嵌套和半結構化數據	僅平面行或元組	可以具有 Amazon Ion 資料格式和 PartiQL 查詢語言支援的任何結構化、半結構化或巢狀資料的文件
查詢元數據	無內建的元資料	SELECT從表的內置提交視圖查詢的語句
查詢修訂歷史	沒有內建的資料歷史	SELECT從內置歷史記錄函數查詢的語句
密碼編譯	沒有內置的密碼學或不變性	傳回期刊摘要的 API，以及驗證任何文件修訂版本相對於該摘要的完整性的證明

如需 QLDB 中核心概念和術語的概觀，請參閱核心概念。

如需在分類帳中建立、查詢及管理資料的程序的詳細資訊，請參閱使用資料和歷程記錄。

亞馬遜 QLDB 中的核心概念和術語

本節提供 Amazon QLDB 核心概念和術語的概觀，包括分類帳結構以及分類帳管理資料的方式。作為總帳數據庫，QLDB 與其他面向文檔的數據庫不同，當涉及到以下關鍵概念時。

主題

QLDB 資料物件模型

亞馬遜 QLDB 中的基本數據對象模型描述如下：

1. 分類帳

   您的第一個步驟是建立分類帳，分類帳是 QLDB 中的主要AWS資源型態。若要瞭解如何建立分類帳，請參閱〈步驟 1：建立新的分類帳開始使用主控台〉或Amazon QLDB 分類帳的基本操作。

   對於分類帳的ALLOW_ALL和STANDARD許可模式，您可以建立AWS Identity and Access Management (IAM) 政策，以授與在此分類帳資源上執行 API 作業的權限。

   分類帳本格式式式式式式

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

2. 期刊和表格

   若要開始在 QLDB 分類帳中寫入資料，您必須先建立具有基本CREATE TABLE陳述式的資料表。分類帳資料包含確認至分類帳分錄的文件修訂。您可以在使用者定義表格的內容中，將文件版次確認至分類帳。在 QLDB 中，表格代表分錄中文件修訂集合的具體化視觀表。

   在分類帳的STANDARD許可模式中，您必須建立 IAM 政策，以授與權限，才能在此表格資源上執行 PartiQL 陳述式。透過表格資源的權限，您可以執行存取資料表目前狀態的陳述式。您還可以使用內置history()函數查詢表格的修訂歷史記錄。

   資料表 ARN 式式式的資料表

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

   如需有關授與分類帳及其相關資訊，請參閱Amazon QLDB 如何與 IAM 合作。

3. 文件

   資料表包含的修訂版QLDB 文件，這些修訂是 Amazon Ionstruct 格式的資料集。文件版本修訂代表以唯一文件 ID 識別的一系列文件的單一版本。

   QLDB 會儲存已提交文件的完整變更記錄。表格可讓您查詢其文件的目前狀態，而history()函數則可讓您查詢表格文件的整個修訂歷程記錄。如需查詢和寫入修訂的詳細資訊，請參閱使用資料和歷程記錄。

4. 系統目錄

   每個分類帳還提供系統定義的型錄資源，您可以查詢此資源，以列出分類帳中的所有表格與索引。在分類帳的STANDARD權限模式中，您需要此目錄資源的權限qldb:PartiQLSelect才能執行下列作業：

   - 在系統目錄表格資訊上執SELECT行陳述式。
   - 在 QLDB 主控台的分類帳詳細資訊頁面上檢視表格和索引資訊。
   - 在 QLDB 主控台的 PartiQL 編輯器中檢視表格和索引的清單。

   目錄的資料 ARN 式式式式式

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

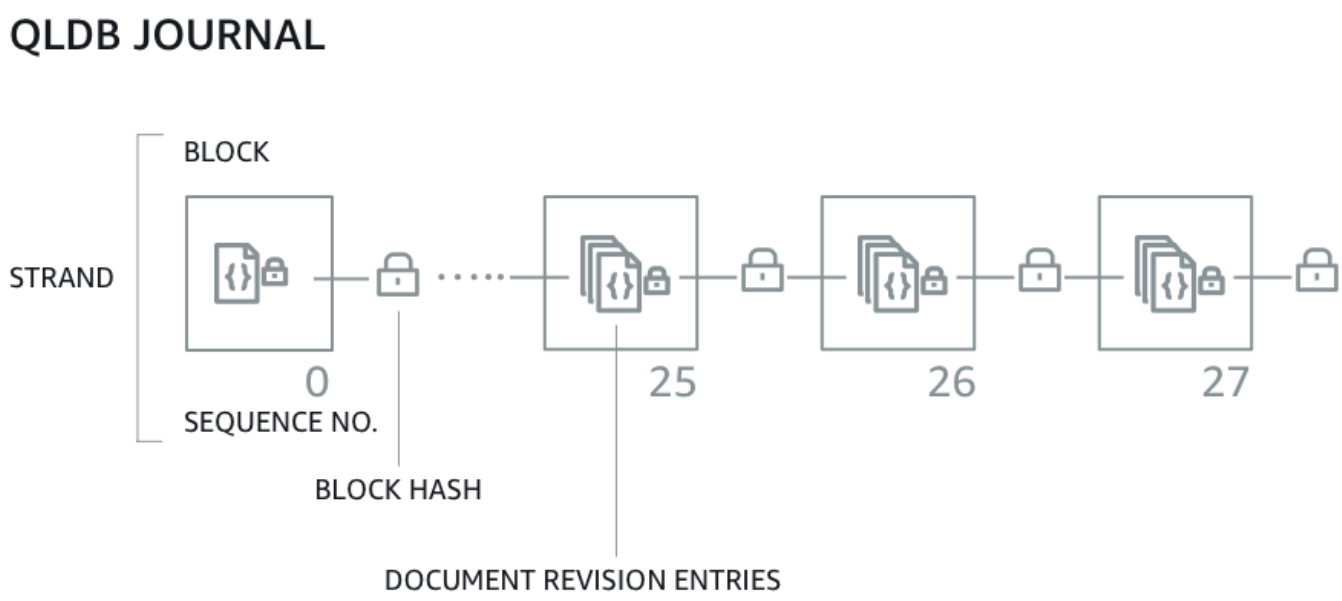
日誌優先的交易

當應用程序讀取或寫入 QLDB 分類帳中的數據時，它會在數據庫事務中執行此操作。所有交易均受到中定義的限制亞馬遜 QLDB 中的配額和限制。在交易中，QLDB 執行以下步驟：

1. 從分類帳中讀取資料的目前狀態。
2. 執行交易中提供的陳述式，然後使用樂觀並行控制 (OCC) 檢查是否有任何衝突，以確保完全可序列化隔離。
3. 如果找不到 OCC 衝突，請依照下列方式傳回交易結果：
   - 對於讀取，返回結果集，並以僅附加的方式將SELECT語句提交到日誌。
   - 對於寫入，請以僅附加的方式將任何更新、刪除或新插入的資料提交至日誌。

該日誌代表了對數據的所有更改的完整和不可變的歷史記錄。QLDB 會將一個鏈結的區塊寫入交易中的日誌。每個區塊都包含項目物件，這些物件代表您插入、更新和刪除的文件修訂版，以及認可它們的 PartiQL 陳述式。

下圖說明此分錄結構的說明此分錄結構



此圖表顯示交易已確認為包含文件修訂分錄的區塊。每個塊都被散列並鏈接到後續塊以進行驗證。每個塊都有一個序列號來指定其在鏈中的地址。

> **Note**
>
> 在 Amazon QLDB 中，鏈條是分類帳期刊的分區。QLDB 目前僅支援單鏈的期刊。

若要取得有關圖塊中資料內容的資訊，請參閱〈〉亞馬遜 QLDB 中的期刊內容。

查詢您的資料

QLDB 的目的想要高效能線上交易處理 (OLTP) 工作負載。分類帳會根據確認至分錄的交易資訊，提供資料的可查詢表格檢視表。在 QLDB 表視圖是在一個表中的數據的子集。視圖是實時維護的，因此它們始終可供應用程序查詢。

您可以使用 PartiQLSELECT 陳述式查詢下列系統定義的檢視：

- 「用戶」— 只有您在表中寫入的數據（即用戶數據的當前狀態）的最新活動版本。這是 QLDB 中的預設檢視。
- 已提交 — 使用者資料和系統產生的中繼資料的最新作用中修訂版本。這是直接對應至您的使用者資料表的完整系統定義表格。

除了這些可查詢的視圖之外，您還可以使用內置的查詢數據的修訂歷史記錄歷史功能。history 函數返回您的用戶數據和相同的模式中的提交視圖相關的元數據。

資料儲存體

有兩種類型的資料儲存體有兩種類型的 QLDB 儲存體：

- 日誌儲存 — 分類帳日誌所使用的磁碟空間。該日誌僅附加，其中包含對數據的所有更改的完整，不可變和可驗證的歷史記錄。
- 索引儲存 — 分類帳表格、索引和索引歷史記錄所使用的磁碟空間。索引儲存包含針對高效能查詢最佳化的總帳資料。

將資料送達日誌之後，就會實現到您定義的資料表中。這些表格經過最佳化，可提供更快、更有效率的查詢 當應用程式使用交易資料 API 讀取資料時，會存取儲存在索引儲存體中的資料表和索引。

QLDB API 模式

QLDB 提供兩種類型的 API，您的應用程式程式碼可以與之互動：

- 亞馬遜 QLDB — QLDB 資源管理 API (也稱為控制平面)。此 API 僅用於管理分類帳資源和非交易資料作業。您可以使用這些作來建立、刪除、描述、列出及更新分類帳。您也可以以密碼方式驗證資料，以及匯出或串流日誌區塊。
- 亞馬遜 QLDB 工作階段 — QLDB 交易資料 API。您可以使用此 API 在具有 PartiQL 陳述式的分類帳上執行資料交易。

> **Important**
>
> 我們建議您不要直接與 QLDB 工作階段 API 互動，而是使用 QLDB 驅動程式或 QLDB 命令介面在分類帳上執行資料交易。
>
> - 如果您使用的是AWS SDK，請使用 QLDB 驅動程式。驅動程式會在 QLDB 工作階段資料 API 上方提供高階層抽象層，並為您管理SendCommand作業。如需相關資訊和支援的程式設計語言清單，請參閱開始使用驅動程式。

- 如果您正在使用AWS CLI，請使用 QLDB 命令介面。殼層是使用 QLDB 驅動程式與總帳互動的命令列介面。如需相關資訊，請參閱 使用亞馬遜 QLDB 外殼 (僅限資料 API)。

如需有關這些 API 操作的詳細資訊，請參閱Amazon QLDB API 參考參考.

後續步驟

若要瞭解如何將分類帳與資料搭配使用，請參閱使用亞馬遜 QLDB 中的數據和歷史記錄並遵循說明建立資料表、插入資料和執行基本查詢程序的範例。本指南使用範例資料和上下文的查詢範例，說明這些概念如何深入運作。

若要透過使用 QLDB 主控台的範例應用程式教學課程快速入門，請參閱Amazon QLDB 主控台。

如需本節所述的主要術語和定義清單，請參閱Amazon QLDB 字彙。

亞馬遜 QLDB 中的期刊內容

在 Amazon QLDB 中，日誌是不可變的交易日誌，用於儲存資料所有變更的完整且可驗證的歷史記錄。該日誌是僅附加的，由一組包含已提交的數據和其他系統元數據的循序和散列鏈式塊組成。QLDB 會將一個鏈結的區塊寫入交易中的日誌。

本節提供包含範例資料的日誌區塊範例，並說明區塊的內容。

主題

區塊範例

日誌區塊包含交易中繼資料，以及代表交易中認可之文件修訂的項目，以及認可它們的 PartiQL 陳述式。

以下是具有範例資料的區塊範例。

> **Note**
>
> 此區塊範例僅供參考。顯示的哈希值不是實際計算的哈希值。

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

在revisions欄位中，某些版本修訂物件可能只包含hash值，而不包含其他屬性。這些是不包含使用者資料的僅限內部系統修訂。這些修訂的雜湊是日誌的完整雜湊鏈的一部分，這是密碼編譯驗證所需的。

區塊內容

日誌區塊欄位如下：

blockAddress

區塊在分錄中的位置。地址是具有兩個字段的 Amazon 離子結構：strandId和sequenceNo。

例如：{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}

transactionId

認可區塊之交易的唯一識別碼。

blockTimestamp

區塊認可至日誌的時間戳記。

blockHash

256 位元雜湊值，唯一表示區塊。這是entriesHash和的串連的雜湊值previousBlockHash。

entriesHash

代表區塊內所有項目的雜湊值，包括僅限內部的系統項目。這是默克爾樹的根散列，其中葉節點由中的所有哈希組成entriesHashList。

previousBlockHash

日誌中上一個鏈接塊的哈希值。

entriesHashList

代表塊中每個條目的哈希列表。此清單可以包含下列條目雜湊值：

- 代表的離子哈希值transactionInfo。這個值是透過取整個transactionInfo結構的離子雜湊值來計算。
- 默克爾樹的根散列，其中葉節點由中的所有哈希組成revisions。
- 代表的離子哈希值redactionInfo。此雜湊只存在於密文交易所認可的區塊中。它的值是通過採取整個redactionInfo結構的離子散列來計算的。
- 代表僅限內部系統中繼資料的雜湊。這些雜湊可能不存在於所有區塊中。

transactionInfo

一種 Amazon Ion 結構，其中包含有關提交區塊之交易中陳述式的資訊。此架構包含下列欄位：

- statements— PartiQL 陳述式的清單以及它們開始執行的startTime時間。每個語句都有一個statementDigest散列，這是計算transactionInfo結構的散列所需的。
- documents— 由陳述式更新的文件 ID。每個文件都包tableId含它所屬的和，以及每個更新它之陳述式的索引。tableName

revisions

已在區塊中確認的文件修訂清單。每個修訂版本結構都包含修訂已提交檢視中的所有欄位。

這也可以包括代表作為日誌完整雜湊鏈一部分的僅限內部系統修訂的雜湊。

已修訂修訂

在 Amazon QLDB 中，DELETE陳述式只會以邏輯方式刪除文件，方法是建立將文件標記為已刪除的新修訂。QLDB 也支援資料編輯作業，可讓您永久刪除表格歷程記錄中非使用中的文件修訂版本。

密文作業只會刪除指定修訂版本中的使用者資料，並保持分錄序列與文件中繼資料不變。這樣可以維護分類帳的整體資料完整性。如需詳細資訊和密文操作範例，請參閱編輯文件修訂版本。

已修訂修訂範例

考慮前面的塊示例。在此區塊中，假設您密文的文件 ID 為HwVFkn8IMRa0xjze5xcgga且版本號碼為的修訂版本0。

完成密文之後，修訂版本中的使用者資料 (由結data構表示) 會被新dataHash欄位取代。此欄位的值是已移除data結構的離子雜湊值。因此，分類帳會維護其整體資料完整性，並透過現有的驗證 API 作業維持密碼編譯驗證。

下列修訂版範例顯示此密文的結果，新dataHash欄位會以###體反白顯示。

> **Note**
>
> 此修訂版範例僅供參考。顯示的哈希值不是實際計算的哈希值。

```
...
{
  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 也會將新區塊附加至已完成的密文請求的日誌。此區塊內含其他redactionInfo條目，其內含交易內已編輯修訂之修訂清單的修訂清單，如下列範例所示。

```
...
```

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

範例應用程式

如需使用匯出資料驗證日誌雜湊鏈的 Java 程式碼範例，請參閱 GitHub 儲存庫 aws-Samples/amazon-qldb-dmv-sample-java。此範例應用程式包含下列類別檔案：

- ValidateQldbHashChain.java — 包含從分類帳匯出分錄區塊的教學課程程式碼，並使用匯出的資料來驗證區塊之間的雜湊鏈。
- JournalBlock.java — 包含名為的方法，`verifyBlockHash()`該方法示範如何計算區塊內的每個個別雜湊元件。此方法由中的教學課程程式碼呼叫`ValidateQldbHashChain.java`。

如需如何下載和安裝此完整範例應用程式的指示，請參閱安裝亞馬遜 QLDB Java 範例應用程式。在您執行教學課程程式碼之前，請確定遵循中的步驟 1-3Java 教學 來設定範例分類帳，並使用範例資料載入分類帳。

另請參閱

如需 QLDB 分錄內分錄的詳細資訊，請參閱下列主題：

- 從 Amazon QLDB 匯出日誌資料— 了解如何將日誌資料匯出到 Amazon SSimple Storage Service (Amazon S3) torage Service
- 從 Amazon QLDB 串流日誌資料— 了解如何將日誌 Data Streams Data Streams Streams Data Streams StreAmazon Kinesis Data Streams
- Amazon QLDB 中的數據驗證— 瞭解日誌資料的密碼編譯驗證。

Amazon QLDB 字彙

以下是在使用 Amazon QLDB 時可能有用的關鍵字詞定義。

區塊 | 消化 | 文件 | 文件識別碼 | 文件修訂 | 條目 | field | index | 索引儲存 | 日誌 | 分錄區塊 | 分錄儲存儲 | 日誌鏈 | 日誌提示 | 分類帳 | 證明 | 修訂版 | 工作階段 | 股 | table | 表格視圖 | 檢視

區塊

在交易中認可至日誌的物件。單一交易會在日誌中寫入一個區塊，因此區塊只能與一個交易相關聯。區塊包含代表交易中已確認之文件修訂版本的項目，以及認可它們的 PartiQL 陳述式。

每個塊還具有用於驗證的哈希值。塊散列是從該塊內的條目哈希結合先前鏈接塊的哈希值計算的。

消化

256 位哈希值，唯一地表示分類帳從某個時間點開始的整個文檔修訂歷史記錄。摘要雜湊是從日誌當時最新提交區塊開始計算的日誌完整雜湊鏈。

QLDB 可讓您產生摘要作為安全輸出檔案。然後，您可以使用該輸出檔案來驗證文件修訂版本相對於該雜湊的完整性。

文件

一組採用 Amazon Ion `struct` 格式的資料，可在表格中插入、更新和刪除。一個 QLDB 文檔可以具有結構化，半結構化，嵌套和無結構描述的數據。

文件識別碼

QLDB 指派給您插入表格中的每個文件的通用唯一識別碼 (UUID)。此識別碼是 128 位元數字，以 Base62 編碼的英數字元字串表示，固定長度為 22 個字元。

文件修訂

Ion 結構，代表由唯一文件 ID 識別之文件序列的單一版本。修訂版包括您的使用者資料 (也就是您在表格中寫入的資料) 和系統產生的中繼資料。每個修訂版本都與一個表格相關聯，並以文件 ID 和從零開始的版本編號組合來唯一識別。

條目

包含在圖塊中的物件。項目代表在交易中插入、更新和刪除的文件修訂版，以及認可它們的 PartiQL 陳述式。

每個項目還具有用於驗證的哈希值。條目哈希從修訂哈希或該條目內的語句哈希計算。

field

一個名稱-值組成 QLDB 文檔的每個屬性。名稱是符號標記，值不受限制。

index

您可以在資料表上建立以最佳化資料擷取作業效能的資料結構。如需 QLDB 中索引的相關資訊，請參閱CREATE INDEX亞馬遜 QLDB PartiQL 參考資料中的。

索引儲存

分類帳表格、索引和索引歷史記錄所使用的磁碟空間。索引儲存包含針對高效能查詢最佳化的總帳資料。

日誌

分類帳中提交的所有區塊的雜湊鏈集。分錄是僅附加的，代表分類帳資料所有變更的完整且不可變的歷史記錄。

分錄區塊

請參閱 區塊。

分錄儲存儲

分類帳日誌所使用的磁碟空間。

日誌鏈

請參閱 股。

日誌提示

在某個時間點的期刊中最新提交的區塊。

分類帳

Amazon QLDB 分類帳資料庫資源的執行個體。這是 QLDB 中的主要AWS資源類型。分類帳由日誌儲存和索引儲存組成。將分類帳資料提交至日誌之後，就可以在 Amazon Ion 文件修訂的表格中查詢。

證明

對於給定的摘要和文檔修訂，QLDB 返回 256 位哈希值的有序列表。它由 Merkle 樹模型將給定修訂散列鏈接到摘要哈希所需的哈希組成。您可以使用證明來驗證修訂版本相對於摘要的完整性。如需詳細資訊，請參閱Amazon QLDB 中的數據驗證。

修訂版

請參閱 文件修訂。

工作階段

管理資料交易請求相關資訊的物件，以及來自分類帳的回應。作用中的工作階段 (主動執行交易的工作階段) 代表與總帳的單一連線。QLDB 支援每個工作階段一個主動執行的交易。

股

日誌的分割區。QLDB 目前僅支援單鏈的期刊。

table

分類帳分錄中確認之文件修訂之無序集合的具體化視觀表。

表格視圖

資料表中可查詢的資料子集，以確認至分錄的交易為基礎。在 PartiQL 陳述式中，檢視表會以資料表名稱的前置詞限定詞 (開頭為_ql_) 來表示。

您可以使用SELECT陳述式查詢下列系統定義的檢視：

- 「用戶」— 只有您在表中寫入的數據（即用戶數據的當前狀態）的最新活動版本。這是 QLDB 中的預設檢視。
- 已提交 — 使用者資料和系統產生的中繼資料的最新作用中修訂版本。這是直接對應至您的使用者資料表的完整系統定義表格。例如：`_ql_committed_`*`TableName`*。

檢視

請參閱表格視圖。

訪問 Amazon QLDB

您可以使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 QLDB API 存取 Amazon QLDB。下列各節說明如何使用這些選項，以及使用這些選項的先決條件。

必要條件

AWS 帳戶 如果您尚未存取 QLDB，則必須先設定，才能存取 QLDB。

主題

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。您可以隨時登錄 https://aws.amazon.com/ 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶根使用者

1. 選擇 Root 使用者並輸入您的 AWS 帳戶 電子郵件地址，以帳戶擁有者身分登入。AWS Management Console在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入 使用者指南中的以根使用者身分登入。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶 根使用者啟用虛擬 MFA 裝置 (主控台)。

建立具有管理權限的使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM 身分中心中，將管理存取權授予使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。

以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

  如需使用 IAM 身分中心使用者登入的說明，請參閱使用AWS 登入 者指南中的登入 AWS 存取入口網站。

指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「建立權限集」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「新增群組」。

在 IAM 中管理 QLDB 許可

如需使用 AWS Identity and Access Management (IAM) 管理使用者之 QLDB 許可的相關資訊，請參閱。Amazon QLDB 如何與 IAM 合作

授予程式設計存取權限 (選用)

如果使用者想要與 AWS 之外的 AWS Management Console. 授與程式設計存 AWS取權的方式取決於正在存取的使用者類型。

若要授與使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需詳細資訊 AWS CLI，請參閱《使 AWS CLI 用AWS Command Line Interface者指南》 AWS IAM Identity Center中的〈配置使用〉。 • 如需 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 IAM 身分中心身分驗證。
IAM	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	遵循《IAM 使用者指南》中的〈將臨時登入資料搭配 AWS 資源使用〉中的指示
IAM	(不建議使用) 使用長期認證簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需相關資訊 AWS CLI，請參閱使用指南中的使用 IAM 使用者登入資料進行驗

哪個使用者需要程式設計存取權？	到	By
		證。AWS Command Line Interface • 對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參考指南中的使用長期憑據進行身份驗證。 • 如需 AWS API，請參閱 IAM 使用者指南中的管理 IAM 使用者的存取金鑰。

如何訪問 Amazon QLDB

完成設定的必要條件之後 AWS 帳戶，請參閱下列主題，以深入了解如何存取 QLDB：

- 使用主控台
- 使用 AWS CLI (僅限管理 API)
- 使用亞馬遜 QLDB 外殼 (僅限資料 API)
- 使用 API

使用主控台存取 Amazon QLDB

您可以訪問 AWS Management Console Amazon QLDB 在 https://console.aws.amazon.com/qldb.

您可以使用控制台在 QLDB 中執行以下操作：

- 建立、刪除、描述及列出分類帳。
- 使用編輯器執行 P artiQL 陳述式。
- 管理 QLDB 資源的標籤。
- 以密碼方式驗證日誌資料。
- 匯出或串流日誌區塊。

若要了解如何建立 Amazon QLDB 分類帳並使用範例應用程式資料進行設定，請參閱。Amazon QLDB 主控台

編輯 PartiQL 快速參考

Amazon QLDB 支援 PartiQL 的子集作為其查詢語言，而 Amazon Ion 則支援以文件為導向的資料格式。如需 PartiQL 之 QLDB 實作的完整指南和更多詳細資訊，請參閱. Amazon QLDB 參考

下列主題提供如何在 QLDB 中使用 PartiQL 的快速參考概觀。

主題

- QLDB 中的快速提示
- 命令
- 系統定義的檢視
- 基本語法規則
- 編 PartiQL 鍵盤快捷鍵

QLDB 中的快速提示

以下是在 QLDB 中使用 PartiQL 的秘訣和最佳作法的簡短摘要：

- 瞭解並行與交易限制 — 包括SELECT查詢在內的所有陳述式都受到樂觀的並行控制 (OCC) 衝突和交易限制，包括 30 秒的交易逾時。
- 使用索引 — 使用高基數索引並執行目標查詢，以最佳化陳述式並避免完整表格掃描。如需進一步了解，請參閱最佳化查詢效能。
- 使用相等述詞 — 索引查詢需要相等運算子 (=或IN)。不等式運算子 (<、>LIKE、BETWEEN) 不符合索引查詢的資格，因此會產生完整的資料表掃描。
- 僅使用內部聯結 — QLDB 僅支援內部聯結。最佳做法是聯結您要加入的每個資料表索引的欄位。為聯結準則和相等述詞選擇高基數索引。

命令

QLDB 支援下 PartiQL 令。

資料定義語言 (DDL)

Command	描述
CREATE INDEX	為表格上的頂層文件欄位建立索引。
CREATE TABLE	創建一個表。
DROP INDEX	從資料表中刪除索引。
DROP TABLE	停用現有表格。
取消刪除資料表	重新啟動非使用中的表格。

資料操作語言 (DML)

Command	描述
DELETE	透過建立文件的新最終修訂版，將使用中的文件標記為已刪除。
從 (插入、移除或設定)	在語義上與. UPDATE
INSERT	將一個或多個文件加入至表格中。
SELECT	從一個或多個表中檢索數據。
UPDATE	更新、插入或移除文件中的特定元素。

DML 陳述式範例

插入

```
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
}
```

更新插入

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

更新-刪除

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

選擇-相關子查詢

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

選擇-內部連接

```
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')
```

選擇-使用 BY 子句獲取文檔 ID

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

系統定義的檢視

QLDB 支援下列資料表的系統定義檢視。

檢視	描述
table_name	表格的預設使用者檢視，其中僅包含使用者資料的目前狀態。
_ql_committed_ *table_name*	資料表的完整系統定義認可檢視，其中包含使用者資料和系統產生的中繼資料 (例如文件 ID) 的目前狀態。
history(*table_name*)	內置歷史記錄函數，返回表的完整修訂歷史記錄。

基本語法規則

QLDB 支援 PartiQL 的下列基本語法規則。

字元	描述
'	單引號表示字符串值，或 Amazon 離子結構中的字段名稱。
"	雙引號表示帶引號的識別碼，例如作為資料表名稱使用的保留字。
`	反引號表示離子文字值。
.	點標記法會存取父結構的欄位名稱。
[]	方括號定義離子`list`，或表示現有列表的從零開始的序號。
{}	大括號定義一個離子`struct`。
<< >>	雙角括號定義了一個 PartiQL 包，它是一個無序集合。您可以使用包裝袋將多個文件插入表格中。
區分大小寫	所有 QLDB 系統物件名稱 (包括欄位名稱和資料表名稱) 都區分大小寫。

編 PartiQL 鍵盤快捷鍵

QLDB 主控台上的 PartiQL 編輯器支援下列鍵盤快速鍵。

動作	macOS	Windows
執行	Cmd+Return	Ctrl+Enter
註解	Cmd+/	Ctrl+/
Clear	Cmd+Shift+Delete	Ctrl+Shift+Delete

使用 (僅限管理 API AWS CLI) 存取 Amazon QLDB

您可以使用 AWS Command Line Interface (AWS CLI) AWS 服務 從命令行控制多個，並通過腳本自動化它們。您可以視需要使 AWS CLI 用進行一次性作業。您也可以使用它在公用程式指令碼中內嵌 Amazon QLDB 操作。

對於 CLI 存取，您需要存取金鑰 ID 和私密存取金鑰。盡可能使用臨時憑證，而不是長期存取金鑰。臨時憑證包含存取金鑰 ID、私密存取金鑰，以及指出憑證何時到期的安全符記。如需詳細資訊，請參閱 IAM 使用者指南中的將臨時登入資料與 AWS 資源搭配使用。

如需中 QLDB 可用之所有命令的完整清單和使用範例 AWS CLI，請參閱《指AWS CLI 令參考》。

> **Note**
>
> AWS CLI 僅支援中列出的`qldb`管理 API 作業Amazon QLDB API 參考參考。此 API 僅用於管理分類帳資源和非交易資料作業。
> 若要使用命令列介面使用 `qldb-session` API 執行資料交易，請參閱使用 QLDB 殼層存取亞馬遜 QLDB (僅限資料 API)。

主題

- 安裝和配置 AWS CLI
- AWS CLI 與 QLDB 搭配使用

安裝和配置 AWS CLI

在 AWS CLI Linux，macOS 系統或視窗上運行。若要安裝和設定它，請參閱「AWS Command Line Interface 使用者指南」中的以下指示：

1. 安裝或更新最新版本的 AWS CLI
2. 快速設定

AWS CLI 與 QLDB 搭配使用

命令列格式包含 Amazon QLDB 作業名稱，後面接著該作業的參數。除了 JSON 之外，還 AWS CLI 支援參數值的速記語法。

用於列help出 QLDB 中所有可用的命令：

```
aws qldb help
```

您還可以使用help來描述特定命令並了解有關其用法的更多信息：

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

例如，若要建立分類帳：

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

使用 QLDB 殼層存取亞馬遜 QLDB (僅限資料 API)

Amazon QLDB 提供用於與交易資料 API 互動的命令列殼層。使用 QLDB 命令介面，您可以在分類帳資料上執行 PartiQL 陳述式。

這個 shell 的最新版本是用 Rust 編寫的，是在默認main分支的 GitHub倉庫 awslabs/amazon-qldb-shell 中開源的。Python 版本 (v1) 仍然可以在master分支上的同一個存儲庫中使用。

> **Note**
>
> 亞馬遜 QLDB 殼層僅支援qldb-session交易資料 API。此 API 僅用於在 QLDB 分類帳上執行 PartiQL 陳述式。
> 若要使用命令列介面與qldb管理 API 作業互動，請參閱使用 (僅限管理 API AWS CLI) 存取 Amazon QLDB。

此工具不打算被納入應用程序中或用於生產目的。此工具的目標是讓您快速地嘗試使用 QLDB 和 PartiQL。

以下幾節描述如何開始使用 QLDB Shell。

主題

先決條件

您開始使用 QLDB Shell 前，您必須執行以下操作：

1. 請遵循中的AWS設定指示訪問 Amazon QLDB。這包含下列項目：
    1. 註冊 AWS。
    2. 建立具有適當 QLDB 許可的使用者。
    3. 授予程式設計存取權以供開發。
2. 設定您的AWS憑證和預設值AWS 區域。如需指示，請參閱《使用指南》中的AWS Command Line Interface〈組態基本〉。

    如需可用區域的完整清單，請參閱中的 Amazon QLDB 端點和配額AWS 一般參考。
3. 對於STANDARD許可模式下的任何分類帳，請建立 IAM 政策，以授與您在適當資料表上執行 PartiQL 陳述式的權限。若要了解如何建立這些原則，請參閱開始使用 Amazon QLDB 中的標準許可模式。

安裝 Shell

若要安裝最新版本的 QLDB Shell，請參閱中的 Readme.md 檔案 GitHub。QLDB 會在 GitHub 儲存庫的「版本」區段中，為 Linux、macOS 和視窗提供預先建置的二進位檔案。

對於 macOS，外殼與aws/tap自製水龍頭集成。若要使用自製程式在 macOS 上安裝 Shell，請執行以下命令。

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

組態

安裝之後，shell 會載入初始化$XDG_CONFIG_HOME/qldbshell/config.ion期間位於的預設組態檔案。Linux 和 macOS 上，此檔案通常位於~/.config/qldbshell/config.ion。如果這樣的檔案不存在，殼層會以預設設定執行。

您可以在安裝後手動建立config.ion檔案。此組態檔案使用 Amazon 離子資料格式。以下是最小config.ion檔案的範例。

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

如果default_ledger未在配置文件中設置，則在調用 shell 時需要該--ledger參數。如需組態選項的完整清單，請參閱中的 README.md 檔案 GitHub。

調用 shell

若要在指令列終端機上呼叫特定總帳的 QLDB Shell，請執行下列命令。替換為您*my-example-ledger*的分類帳名稱。

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

此指令會連接至您的預設值AWS 區域。若要明確指定「區域」，您可以使用--region或--qldb-session-endpoint參數執行命令，如下一節所述。

調用qldb shell 會話後，您可以輸入以下類型的輸入：

- 薄殼指令
- 單個 PartiQL 語句在單獨的事務
- 交易中的多個 PartiQL 陳述式

薄殼參數

如需可用旗標和叫用 shell 選項的完整清單，請使用`--help`旗標執行`qldb`命令，如下所示。

```
$ qldb --help
```

以下是`qldb`指令的一些主要旗標和選項。您可以新增這些選用參數來覆寫AWS 區域、認證設定檔、端點、結果格式和其他組態選項。

用途

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

旗

`-h`, `--help`

列印說明資訊。

`-v`, `--verbose`

設定記錄詳細資訊。根據預設，殼層只會記錄錯誤。若要提高詳細程度層級，請重複此引數 (例如，`-vv`)。最高層級是`-vvv`對應於`trace`詳細程度。

`-V`, `--version`

列印版本資訊。

OPTIONS

`-l`,`--ledger` *#####*

要連線分類帳的名稱。如果沒有在您的`config.ion`文件中設置，這`default_ledger`是必需的 shell 參數。在此檔案中，您可以設定其他選項，例如「區域」。

`-c`，`--config`*####*

您可以在其中定義任何 shell 配置選項的文件。如需格式化詳細資訊和組態選項的完整清單，請參閱中的 README.md 檔案 GitHub。

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

查詢結果的輸出格式。預設值為 `ion`。

-p、--profile*###*

用於驗證的AWS認證設定檔位置。

如果未提供，shell 會使用您的預設設AWS定檔，該設定檔位於~/.aws/credentials。

-r,--region *####*

要連接的 QLDB 分類帳的AWS 區域程式碼。例如：us-east-1。

如果未提供，shell 會連接到您的設AWS 區域定AWS檔中指定的預設值。

-s,--qldb-session-endpoint *QLDB #### _ ##*

要連線的qldb-session API 端點。

如需可用 QLDB 區域和端點的完整清單，請參閱中的 Amazon QLDB 端點和配額AWS 一般參考。

命令參考

在您叫用qldb工作階段之後，殼層支援下列金鑰和資料庫命令：

外殼鑰匙

索引鍵	功能描述
Enter	執行陳述式。
Escape+Enter (macOS 系統) Shift+Enter (視窗)	開始新行以輸入跨越多行的陳述式。您也可以複製帶有多行的輸入文本並將其粘貼到 shell 中。 如需在 macOS 中設定Option而非中Escape繼金鑰的指示，請參閱 OS X 每日網站。
Ctrl+C	取消目前的指令。
Ctrl+D	信號的文件結束（EOF），並退出殼的當前級別。如果不在使用中交易中，則結束 shell。在作用中的交易中，中止交易。

殼牌資料庫指令

命令	功能描述
`help`	顯示協助資訊。
`begin`	開始交易。
`start transaction`	
`commit`	將交易確認至分類帳的分錄。
`abort`	停止交易並拒絕您所做的任何變更。
`exit`	結束 Shell。
`quit`	

> **Note**
>
> 所有 QLDB Shell 命令不區分大小寫。

執行個別陳述式

除了 README.md 中列出的資料庫命令和殼層中繼命令外，殼層會將您輸入的每個命令解譯為個別的 PartiQL 陳述式。根據預設，Shell 啟用`auto-commit`模式。此模式是可配置的。

在此`auto-commit`模式中，殼層會以隱含方式在自己的交易中執行每個陳述式，並在找不到錯誤時自動認可交易。這表示您不必在每次執行陳述式時執行`start transaction` (或`begin`) 和`commit`手動執行。

管理交易

或者，QLDB 命令介面可讓您手動控制交易。您可以在交易中以互動方式執行多個陳述式，也可以按順序批次處理命令和陳述式，以非互動方式執行。

互動式交易

若要執行互動式交易，請執行以下步驟。

1. 若要開始交易，請輸入begin指令。

```
qldb> begin
```

在您開始交易之後，殼層會顯示下列命令提示字元。

```
qldb *>
```

2. 然後，您輸入的每個陳述式都會在相同的交易中執行。
   - 例如，您可以執行如下單一陳述式。

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

按下之後Enter，殼層會顯示陳述式的結果。

   - 您也可以輸入多個以分號 (;) 分隔符號分隔的陳述式或命令，如下所示。

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

3. 若要結束交易，請輸入以下其中一個命令。
   - 輸入commit指令，將交易確認至分類帳的分錄。

```
qldb *> commit
```

   - 輸入abort指令以停止交易並拒絕您所做的任何變更。

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

交易逾時限制

互動式交易會遵守 QLDB 的交易逾時限制。如果您未在啟動交易後 30 秒內提交交易，QLDB 會自動過期交易，並拒絕交易期間所做的任何變更。

接著，殼層會顯示到期錯誤訊息，並返回正常的命令提示字元，而不是顯示陳述式結果。若要重試，您必須再次輸入begin命令，才能開始新的交易。

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

非互動式交易

您可以依照下列順序批次處理命令和陳述式，以多個陳述式執行完整的交易。

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

您必須使用分號 (;) 分隔符號來分隔每個命令和陳述式。如果交易中的任何陳述式無效，殼層會自動拒絕交易。命令介面不會繼續處理您輸入的任何後續陳述式。

您也可以設定多個交易。

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

與前面的範例類似，如果交易失敗，shell 就不會繼續處理您輸入的任何後續交易或陳述式。

如果您不結束交易，殼層會切換到互動模式，並提示您輸入下一個命令或陳述式。

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

退出 Shell

若要結束目前的qldb殼層工作階段，請輸入exit或quit命令，或在 shell 不在交易中D時使用鍵盤快速鍵Ctrl +。

```
qldb> exit
$
```

```
qldb> quit
$
```

範例

如需有關在 QLDB 中撰寫 PartiQL 陳述式的詳細資訊，請參閱Amazon QLDB 參考.

Example

以下範例顯示一般的基本命令序列。

> **Note**
>
> QLDB 命令介面會在本身的交易中執行此範例中的每個 PartiQL 陳述式。
> 此範例假設分類帳`test-ledger`已存在且處於作用中狀態。

```
$ 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
```

使用 API 存取 Amazon QLDB

您可以使用 AWS Management Console 和 AWS Command Line Interface (AWS CLI) 以互動方式與 Amazon QLDB 工作。但是，為了充分利用 QLDB，您可以使用 QLDB 驅動程式或 AWS SDK 撰寫應用程式程式碼，以便使用 API 與總帳互動。

驅動程式可讓您的應用程式使用交易資料 API 與 QLDB 互動。 AWS SDK 支援與 QLDB 資源管理 API 的互動。如需這些 API 的詳細資訊，請參閱Amazon QLDB API 參考參考.

該驅動程序提供了 Java，淨，圍棋，Node.js 和 Python 中的 QLDB 支持。若要快速開始使用這些語言，請參閱 開始使用 Amazon QLDB 驅動程式。

您必須先授與程式設計存取權，才能在應用程式中使用 QLDB 驅動程式或 AWS SDK。如需更多詳細資訊，請參閱 授與程式設計存取權。

Amazon QLDB 主控台

本教學會引導您逐步建立第一個 Amazon QLDB 分類帳，並使用表格和範例資料填入分類帳。您在此案例中建立的範例分類帳是機動車輛部門 (DMV) 應用程式的資料庫，可追蹤車輛註冊的完整歷史資訊。

資產的歷史記錄是 QLDB 的常見使用案例，因為它涉及各種突出分類帳資料庫實用性的案例和作業。透過 QLDB，您可以在支援 SQL 類查詢功能的文件導向資料庫中，直接存取、查詢和驗證資料變更的完整歷程記錄。

在您完成本教學課程時，下列主題將說明如何新增車輛註冊、修改它們，以及檢視這些註冊的變更歷史記錄。本指南還向您展示如何以密碼方式驗證註冊文件，並透過清理資源和刪除範例分類帳來結束。

教學課程中的每個步驟都有使用AWS Management Console.

主題

教學專用主題和考量

開始本 Amazon QLDB 教學課程之前，請確定您已完成以下先決條件：

1. 如果您尚未這麼做訪問 Amazon QLDB，請依照中的指示AWS執行。這些步驟包括註冊AWS和建立管理使用者。
2. 遵循中設定許可的指示，為您的 QLDB 資源設定 IAM 許可。若要完成本教學課程的所有步驟，您需要透過分類帳的完整管理存取AWS Management Console。

> **Note**
>
> 如果您已以具完整AWS管理權限的使用者身分登入，則可略過此步驟。

3. (選擇性) QLDB 使用AWS Key Management Service () 中的金鑰加密靜態資AWS KMS料。您可以選擇下列其中一個類型AWS KMS keys：
   - AWS擁有的 KMS 金鑰 — 使用由代表您擁有和管理AWS的 KMS 金鑰。這是預設選項，不需要額外的設定。
   - 客戶受管 KMS 金鑰 — 使用在您帳戶中建立、擁有和管理的對稱加密 KMS 金鑰。QLDB 不支援非對稱金鑰。

     此選項需要您建立 KMS 金鑰或使用帳戶中的現有金鑰。如需建立客戶受管金鑰的指示，請參閱AWS Key Management Service開發人員指南中的建立對稱加密 KMS 金鑰。

     您可使用 ID、別名或 Amazon 資源名稱 (ARN) 指定客戶受管 KMS 金鑰。若要深入了解，請參閱AWS Key Management Service開發人員指南中的金鑰識別碼 (KeyId)。

     > **Note**
     >
     > 不支援跨區域金鑰。指定的 KMS 金鑰必須位於與總帳相AWS 區域同的 KMS 金鑰。

設定許可

在此步驟中，您可以透過主控台為. 中的所有 QLDB 資源設定完整存取權限AWS 帳戶。若要快速授予這些權限，請使用 AmazonQLDB 的AWS受管原則ConsoleFullAccess。

若要提供存取權，請新增許可到您的使用者、群組或角色：

- AWS IAM Identity Center 中的使用者和群組：

  建立許可集合。請遵循《AWS IAM Identity Center 使用者指南》的建立許可集合中的指示。
- 透過身分提供者在 IAM 中管理的使用者：

  建立聯合身分的角色。請遵循《IAM 使用者指南》的為第三方身分提供者 (聯合) 建立角色中的指示。
- IAM 使用者：
  - 建立您的使用者可擔任的角色。請遵循《IAM 使用者指南》的為 IAM 使用者建立角色中的指示。

- (不建議) 將政策直接連接至使用者，或將使用者新增至使用者群組。請遵循《IAM 使用者指南》的新增許可到使用者 (主控台)中的指示。

> **Important**
>
> 針對本教學課程的目的，您授與自己對所有 QLDB 資源的完整管理存取權。然而，對於生產使用案例，請遵循授予最低權限的安全性最佳作法，或者只授予執行任務所需的許可。如需範例，請參閱Amazon QLDB 以身分識別為基礎的政策範例。

若要建立名為的分類帳`vehicle-registration`，請繼續執行步驟 1：建立新的分類帳。

步驟 1：建立新的分類帳

在此步驟中，您會建立名為的新 Amazon QLDB 分類帳`vehicle-registration`。然後，您確認分類帳的狀態為「有效」。您也可以驗證新增至分類帳的任何標籤。

當您建立分類帳時，預設會啟用刪除保護功能。QLDB 中的刪除分類帳保護功能，可防止任何使用者刪除總帳。您可以在使用 QLDB API 或AWS Command Line Interface (AWS CLI) 建立分類帳時停用刪除保護。

建立新分類帳

1. 登入AWS Management Console，開啟位於 https://console.aws.amazon.com/qldb 的 Amazon QLDB 主控台。
2. 在導覽窗格中，選擇 Monitor (入門)。
3. 在「建立您的第一個總帳卡」 上，選擇 「建立分類帳」。
4. 在 「建立分類帳」 頁面上，執行下列動作：
   - 分類帳資訊 —「分類帳」名稱應預先植入**`vehicle-registration`**。
   - 許可模式 — 要指派給分類帳的許可模式。請選擇下列其中一個選項：
     - 允許全部允許可模式，可使用 API 層級精細程度啟用對分類帳的存取控制。

       此模式允許具有指定分類帳 SendCommand API 許可的使用者 (因此為 ALLOW_ALL)，在此分類帳中的任何資料表上執行所有 PartiQL 命令。此模式會忽略您為分類帳建立之任何資料表層級或命令層級 IAM 許可政策。

- 標準 — (建議) 一種許可模式，可使用更細的精細程度啟用對分類帳的存取控制。強烈建議您使用此許可模式來最大化分類帳資料的安全性。

  根據預設，此模式會拒絕所有在此分類帳的任何資料表上執行任何 PartiQL 命令的請求。若要允許使用 PartiQL 命令，您還必須為特定資料表資料表資源和 PartiQL 動作建立 IAM 許可之外，您還必須為特定資料表資SendCommand料表資源和 PartiQL 動作建 如需相關資訊，請參閱 開始使用 Amazon QLDB 中的標準許可模式。

- 靜態資料加密 —AWS Key Management Service (AWS KMS) 中的金鑰，可使用於加密靜態資料的金鑰。請選擇下列其中一個選項：
  - 使用AWS擁有的 KMS 金鑰 — 使用由代表您擁有和管理AWS的 KMS 金鑰。這是預設選項，不需要額外的設定。
  - 選擇不同的金AWS KMS鑰 — 使用在您帳戶中建立、擁有和管理的對稱加密 KMS 金鑰。

    若要使用AWS KMS主控台建立新金鑰，請選擇 [建立AWS KMS金鑰]。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的建立對稱加密 KMS 金鑰。

    若要使用現有的 KMS 金鑰，請從下拉式清單中選擇一個金鑰，或指定 KMS 金鑰 ARN。
- 標籤 — (選用) 藉由連接標籤作為索引鍵/值組，將中繼資料表資料表中繼資料表。您可以在總帳中新增標籤，協助整理和識別它們。如需詳細資訊，請參閱標記 Amazon QLDB 資源。

  選擇 [新增標籤]，然後視需要輸入任何索引鍵值配對。

5. 當您滿意設定後，選擇 Create (建立分類帳)。

> **Note**
>
> 當 QLDB 分類帳狀態為「有效」時，您可以存取 QLDB 分類帳。這可能需要幾分鐘的時間。

6. 在「分類帳」清單中，找出`vehicle-registration`並確認分類帳的狀態為「有效」。
7. (選擇性) 選擇分`vehicle-registration`類帳名稱。在車輛登記分類帳詳細資料頁面上，確認您新增至分類帳的任何標籤都會顯示在「標籤」卡片上。您也可以使用此主控台頁面編輯分類帳標籤。

若要在`vehicle-registration`分類帳中建立表格，請繼續執行步驟 2：在分類帳中建立資料表、索引和範例資料。

步驟 2：在分類帳中建立資料表、索引和範例資料

Amazon QLDB 分類帳處於作用中狀態時，您可以開始建立有關車輛、其擁有者及其註冊資訊的資料表。創建表和索引後，您可以使用數據加載它們。

在此步驟中，您可以在vehicle-registration分類帳中建立四個表格：

- VehicleRegistration
- Vehicle
- Person
- DriversLicense

您也可以建立下列索引。

資料表名稱	欄位
VehicleRegistration	VIN
VehicleRegistration	LicensePlateNumber
Vehicle	VIN
Person	GovId
DriversLicense	LicensePlateNumber
DriversLicense	PersonId

您可以使用 QLDB 主控台自動建立這些含索引的表格，並使用範例資料載入它們。或者，您也可以使用主控台上的 PartiQL 編輯器來手動執行每個 PartiQL 陳述式 step-by-step。

自動選項

若要建立表格、索引和範例資料

1. 開啟亞馬遜 QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在導覽窗格中，選擇 Monitor (入門)。

3. vehicle-registration在「範例應用程式資料卡」上的「自動」選項下，從分類帳清單中選擇。
4. 選擇「載入範例資料」。

   如果作業順利完成，主控台會顯示已載入的範例資料訊息。

   此指令碼會在單一交易中執行所有陳述式。如果交易的任何部分失敗，每個陳述式都會復原，並顯示適當的錯誤訊息。您可以在解決任何問題後重試此作業。

   > **Note**
   >
   > - 交易失敗的一個可能原因是嘗試建立重複的資料表。如果分類帳中已存在下列任一表格名稱，則載入範例資料的請求將會失敗：VehicleRegistrationVehiclePerson、、和DriversLicense。
   >
   >   請嘗試在空白分類帳中載入此範例資料。
   > - 此指令碼會執行參數化INSERT陳述式。因此，這些 PartiQL 陳述式會以繫結參數而非常值資料記錄在您的日誌區塊中。例如，您可能會在日誌區塊中看到下列陳述式，其中問號 (?) 是文件內容的變數預留位置。
   >
   >   ```
   >   INSERT INTO Vehicle ?
   >   ```

手動選項

您可以VehicleRegistration使用空白PrimaryOwner欄位插入文件，並DriversLicense使用空白PersonId欄位插入。稍後，您會使用Person表格id中的系統指派文件填入這些欄位。

> **Tip**
>
> 最佳作法是使用此文件id中繼資料欄位做為外部索引鍵。如需詳細資訊，請參閱查詢文件元資料。

若要建立表格、索引和範例資料

1. 開啟亞馬遜 QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在導覽窗格中，選擇 PartiQL 編輯器。

3. 選擇分vehicle-registration類帳。
4. 首先創建四個表。QLDB 支援開放式內容，不會強制執行結構定義，因此您不需要指定屬性或資料類型。

   在查詢編輯器視窗中，輸入下列陳述式，然後選擇 Run (執行)。若要執行陳述式，您也可以使用鍵盤快速鍵Ctrl + (適用Enter於 Windows) 或Cmd + (適用Return於 macOS)。如需更多鍵盤快速鍵，請參閱編 PartiQL 鍵盤快捷鍵。

   ```
   CREATE TABLE VehicleRegistration
   ```

   針對下每個步驟重複此步驟。

   ```
   CREATE TABLE Vehicle
   ```

   ```
   CREATE TABLE Person
   ```

   ```
   CREATE TABLE DriversLicense
   ```

5. 接下來，建立最佳化每個資料表查詢效能的索引。

   > **Important**
   >
   > QLDB 需要一個索引來有效地查找文檔。如果沒有索引，QLDB 需要在讀取文檔時進行全表掃描。這可能會造成大型資料表的效能問題，包括並行衝突和交易逾時。
   > 若要避免資料表掃描，您必須在索引欄位或文件 ID 上使用相等運算子 (=或IN) 執行具有述WHERE詞子句的陳述式。如需詳細資訊，請參閱最佳化查詢效能。

   在查詢編輯器視窗中，輸入下列陳述式，然後選擇 Run (執行)。

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

   針對以下步驟重複此步驟。

   ```
   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. 創建索引後，您可以開始將數據加載到表中。在此步驟中，將文檔插入到Person表格中，其中包含有關分類帳正在跟踪的車輛的所有者的個人信息。

   在查詢編輯器視窗中，輸入下列陳述式，然後選擇 Run (執行)。

```
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. 然後，在DriversLicense表格中填入包含每位車主的駕駛執照資訊的文件。

   在查詢編輯器視窗中，輸入下列陳述式，然後選擇 Run (執行)。

```
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. 現在，用車輛登記文件填充VehicleRegistration表格。這些文件包括儲存主要和次要擁有者的巢狀Owners結構。

   在查詢編輯器視窗中，輸入下列陳述式，然後選擇 Run (執行)。

```
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. 最後，在Vehicle表格中填入描述在分類帳中註冊的車輛的文件。

   在查詢編輯器視窗中，輸入下列陳述式，然後選擇 Run (執行)。

```
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'
} >>
```

接下來，您可以使用SELECT語句從vehicle-registration分類帳中的表中讀取數據。繼續執行「步驟 3：查詢分類帳中的表格」。

步驟 3：查詢分類帳中的表格

在 Amazon QLDB 分類帳中建立表格並將其載入資料後，您可以執行查詢以檢閱剛插入的車輛註冊資料。QLDB 使用 PartiQL 作為其查詢語言，而亞馬遜離子作為其文件導向的資料模型。

PartiQL 是一種開放原始碼、SQL 相容的查詢語言，已經擴充為與 Ion 搭配使用。使用 PartiQL，您可以使用熟悉的 SQL 運算子插入、查詢和管理資料。亞馬遜離子是 JSON 的超集合。Ion 是一種開放原始碼、以文件為基礎的資料格式，可讓您靈活地儲存和處理結構化、半結構化和巢狀資料。

在此步驟中，您可以使用SELECT陳述式從vehicle-registration分類帳中的表格讀取資料。

> **Warning**
>
> 當您在沒有索引查閱的情況下在 QLDB 中執行查詢時，它會叫用完整資料表掃描。PartiQL 支持這樣的查詢，因為它是 SQL 兼容。不過，請勿在 QLDB 中針對生產使用案例執行資料表掃描。資料表掃描可能會造成大型資料表的效能問題，包括並行衝突和交易逾時。
> 若要避免資料表掃描，您必須在索引欄位或文件 ID 上使用相等運算子來執行具有述WHERE詞子句的陳述式；例如，`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`。如需詳細資訊，請參閱最佳化查詢效能。

若要查詢資料表

1. 開啟亞馬遜 QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在導覽窗格中，選擇 PartiQL 編輯器。
3. 選擇分vehicle-registration類帳。
4. 在查詢編輯器視窗中，輸入下列陳述式，以查詢Vehicle表格中新增至分類帳的特定車輛識別碼 (VIN)，然後選擇「執行」。

   若要執行陳述式，您也可以使用鍵盤快速鍵Ctrl + (適用Enter於 Windows) 或Cmd + (適用Return於 macOS)。如需更多鍵盤快速鍵，請參閱編 PartiQL 鍵盤快捷鍵。

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

5. 您可以編寫內部聯接查詢。此查詢範例會連Vehicle接VehicleRegistration並傳回指定車輛的註冊資訊，以及已註冊車輛的屬性VIN。

   輸入下列陳述式，然後選擇 [執行]。

   ```
   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
   ```

   您也可以聯結Person和DriversLicense表格，以查看與新增至分類帳之動因相關的屬性。

   針對以下步驟重複此步驟。

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

若要瞭解如何在vehicle-registration分類帳中修改表格中的文件，請參閱步驟 4：修改分類帳中的文件。

步驟 4：修改分類帳中的文件

既然您有資料需要使用，就可以開始變更 Amazon QLDB 中vehicle-registration分類帳中的文件。例如，考慮帶有 VIN 的奧迪 A51N4AL11D75C109151。這輛車最初由華盛頓州西雅圖的一名叫勞爾·劉易斯的司機擁有。

假設勞爾將汽車出售給埃弗里特，華盛頓州名為布倫特·洛根的居民。然後，布倫特和亞歷克西斯·佩納決定結婚。布倫特希望添加亞歷克西斯作為註冊的第二所有者。在此步驟中，下列資料處理語言 (DML) 陳述式示範如何在分類帳中進行適當的變更以反映這些事件。

> **Tip**
>
> 最佳作法是使用文件的系統指派id為外部索引鍵。雖然您可以定義要做為唯一識別碼的欄位 (例如，車輛的 VIN)，但文件的真正唯一識別碼就是其id。此欄位包含在文件的中繼資料中，您可以在認可的檢視 (資料表的系統定義檢視) 中進行查詢。
> 如需 QLDB 中視圖的詳細資訊，請參閱核心概念。若要進一步了解中繼資料，請參閱查詢文件元資料。

若要修改文件

1. 開啟亞馬遜 QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在導覽窗格中，選擇 PartiQL 編輯器。
3. 選擇分vehicle-registration類帳。

> **Note**
>
> 如果您使用主控台的自動載入範例資料功能設定分類帳，請跳至步驟 6。

4. 如果您手動執行INSERT陳述式來載入範例資料，請繼續執行下列步驟。

   首先要將勞爾註冊為這輛車的車主，首先在Person桌子中找到他的系統分配id的文件。此欄位包含在文件的中繼資料中，您可以在資料表的系統定義檢視中進行查詢，稱為已提交的檢視。

   在查詢編輯器視窗中，輸入下列陳述式，然後選擇 Run (執行)。

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

   前綴_ql_committed_是保留的前綴，表示您要查詢Person表的提交視圖。在此檢視中，您的資料以巢狀方式嵌套在data欄位中，而中繼資料則以巢狀方式嵌套在metadata欄位中。
5. 現在，在UPDATE語句id中使用它來修改VehicleRegistration表格中的適當文檔。輸入下列陳述式，然後選擇 [執行]。

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

   發出此聲明，確認您已修改此Owners欄位。

   ```
   SELECT r.Owners FROM VehicleRegistration AS r
   WHERE r.VIN = '1N4AL11D75C109151'
   ```
6. 要將車輛的所有權轉移到埃弗里特市的布倫特，請首先使用以下語句id從Person表中找到他。

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

接下來，使id用它來更新VehicleRegistration表City中的PrimaryOwner和。

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

發出此聲明，確認您已修改PrimaryOwner和City欄位。

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

7. 要添加亞歷克西斯作為汽車的二級所有者，找到她Person id。

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

然後，使用下SecondaryOwners列 FROM-插入 DML 陳述式id將其插入到清單中。

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

發出本聲明以確認您已修改SecondaryOwners。

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

若要複查vehicle-registration分類帳中的這些變更，請參閱步驟 5：檢視文件的修訂歷程。

步驟 5：檢視文件的修訂歷程

使用 VIN 修改汽車的註冊數據後 1N4AL11D75C109151，您可以查詢其所有註冊車主的歷史記錄以及任何其他更新的字段。您可以查詢內建版本，查看您插入、更新和刪除的文件的所有修訂版本歷史功能。

history 函數從表的提交視圖返回修訂，其中包括應用程序數據和關聯的元數據。元數據顯示每個修訂的確切時間，以何種順序以及提交它們的交易。

在此步驟中，您會查詢vehicle-registration分類帳VehicleRegistration表格中文件的修訂歷史記錄。

若要檢視修訂歷史記錄

1. 開啟亞馬遜 QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在導覽窗格中，選擇 PartiQL 編輯器。
3. 選擇分vehicle-registration類帳。
4. 要查詢文檔的歷史記錄，首先找到它的唯一性id。除了查詢已提交的檢視之外，取得文件的另一種方式id是在資料表的預設使用者檢視中使用BY關鍵字。如需進一步了解，請參閱 使用 BY 子句來查詢文件 ID。

   在查詢編輯器視窗中，輸入下列陳述式，然後選擇 Run (執行)。

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

5. 接下來，您可以使用此id值來查詢歷史記錄功能。輸入下列陳述式，然後選擇 [執行]。視情請務必使用自己的文件 ID 取代此id值。

   ```
   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**
   >
   > 為了符合本教學的目的，此歷程查詢會傳回文件 ID 的所有修訂ADR2LQq48kB9neZDupQrMm。但是，最佳作法是使用文件 ID 和日期範圍 (開始時間和結束時間) 來限定歷史記錄查詢。
   > 在 QLDB 中，每個SELECT查詢都會在交易中處理，並受到交易逾時限制。包含開始時間和結束時間的歷史查詢可獲得日期範圍限定的好處。如需詳細資訊，請參閱歷史功能。

   歷史函數返回與提交視圖相同的模式中的文檔。此範例會投影您修改過的車輛登記資料。輸出應看起來如下列內容。

VIN	城市	擁有者
"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**
>
> 歷史記錄查詢不一定會以順序傳回文件修訂版本。

檢閱輸出並確認變更是否反映您在中所做的操作步驟 4：修改分類帳中的文件。

6. 然後，您可以檢查每個修訂版的文檔元數據。輸入下列陳述式，然後選擇 [執行]。同樣地，請務必視情況使用您自己的文件 ID 取代該id值。

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

輸出應看起來如下列內容。

版本	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"

這些中繼資料欄位會提供每個項目的修改時間以及哪些交易的詳細資訊。從此資料中，您可以推斷下列內容：

- 文件由其系統指定的id:唯一識別ADR2LQq48kB9neZDupQrMm。這是一個通用唯一標識符（UUID），以 Base62 編碼的字符串表示。
- txTime顯示文件 (版本0) 的初始版本修訂是在建立的2019-05-23T19:20:360d-3Z。
- 每個後續異動都會建立具有相同文件id、遞增的版本編號以及更新的和的新txId修訂版本txTime。

若要以密碼方式驗證vehicle-registration分類帳中的文件版次，請繼續執行步驟 6：驗證分類帳中的文件。

步驟 6：驗證分類帳中的文件

使用 Amazon QLDB，您可以使用搭配 SHA-256 的加密雜湊，有效地驗證分類帳日誌中文件的完整性。在這個例子中，亞歷克西斯和布倫特決定通過在汽車經銷商與 VIN1N4AL11D75C109151 交易車輛升級到一個新的車型。經銷商通過與註冊辦事處驗證車輛的所有權來啟動該過程。

若要深入瞭解驗證和加密雜湊如何在 QLDB 中運作，請參閱Amazon QLDB 中的數據驗證。

在此步驟中，您會核對vehicle-registration分類帳中的文件版次。首先，您要求摘要，該摘要會以輸出檔案的形式傳回，並做為分類帳整個變更歷史記錄的簽章。然後，您要求相對於該摘要的修訂版本的證明。使用此證明，如果所有驗證檢查都通過，則會驗證修訂的完整性。

要求摘要

1. 開啟亞馬遜 QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在導覽窗格中，選擇分類帳。
3. 在分類帳清單中，選取vehicle-registration。
4. 選擇取得摘要。「取得摘要」對話方塊會顯示下列摘要詳細資訊：
   - 摘要 — 您要求之摘要的 SHA-256 雜湊值。
   - 摘要提示位址 — 您要求摘要涵蓋的日誌中最新的區塊位置。地址有下列兩個欄位：
     - strandId— 包含區塊之日誌鏈的唯一 ID。
     - sequenceNo— 指定圖塊在鏈中位置的索引號碼。
   - 分類帳 — 您請求摘要的分類帳名稱。
   - 日期 — 您請求摘要時的時間戳記。
5. 檢閱摘要資訊。然後選擇 Save (儲存)。您可以保留預設檔案名稱，或輸入新名稱。

   此步驟會以 Amazon Ion 格式儲存內容的純文字檔案。檔案的副檔名為，.ion.txt並包含上述對話方塊中列出的所有摘要資訊。以下是摘要檔案內容的範例。欄位的順序可能會因您的瀏覽器而有所不同。

   ```
   {
     "digest": "42zaJOfV8iGutVGNaIuzQWhD5Xb/5B9lScHnvxPXm9E=",
     "digestTipAddress": "{strandId:\"BlFTjlSXze9BIh1KOszcE3\",sequenceNo:73}",
     "ledger": "vehicle-registration",
     "date": "2019-04-17T16:57:26.749Z"
   }
   ```
6. 將此檔案儲存在稍後可存取的位置。在下列步驟中，您可以使用此檔案來驗證文件的修訂版本。

儲存分類帳摘要之後，您可以開始根據該摘要驗證文件修訂的程序。

> **Note**
>
> 在用於驗證的生產用例中，您使用先前保存的摘要，而不是連續執行兩個任務。最佳作法是在稍後要驗證的修訂寫入日誌後，請求並儲存摘要。

核對文件修訂

1. 首先，查詢您要核對`id`和`blockAddress`之文件版次的與分類帳。這些欄位包含在文件的中繼資料中，您可以在已提交的檢視中進行查詢。

   文件`id`是系統指派的唯一 ID 字串。這`blockAddress`是一個 Ion 結構，用於指定提交修訂的塊位置。

   在 QLDB 主控台的導覽窗格中，選擇 PartiQL (專用主控台)。
2. 選擇分`vehicle-registration`類帳。
3. 在查詢編輯器視窗中，輸入下列陳述式，然後選擇 Run (執行)。

   ```
   SELECT r.metadata.id, r.blockAddress
   FROM _ql_committed_VehicleRegistration AS r
   WHERE r.data.VIN = '1N4AL11D75C109151'
   ```
4. 複製並儲存`id`查詢傳回的 AND`blockAddress` 值。請務必省略`id`欄位的雙引號。在 Amazon Ion 中，字串資料類型會以雙引號分隔。
5. 現在您已經選擇了文件修訂版，您可以開始驗證它的過程。

   在導覽窗格中，選擇 Aquery (驗證)。
6. 在 [驗證文件] 表單的 [指定您要驗證的文件] 下，輸入下列輸入參數：

   - 分類帳-選擇`vehicle-registration`。
   - 區塊位址 — 您在步驟 3 中查詢傳回的`blockAddress`值。
   - 文件 ID — 您在步驟 3 中查詢傳回的`id`值。
7. 在「指定用於驗證的摘要」下，選擇「選擇摘要」來選取先前儲存的摘要。如果該文件有效，這會自動填充控制台上的所有摘要字段。或者，您可以直接從摘要文件中手動複製並粘貼以下值：

   - 摘要 — 摘要檔案中的`digest`值。
   - 摘要提示位址 — 摘要檔案中的`digestTipAddress`值。

8. 檢閱文件和摘要輸入參數，然後選擇 [驗證]。

   控制台會為您自動執行兩個步驟：

   a. 向 QLDB 索取您指定文件的證明。
   b. 使用 QLDB 傳回的證明來呼叫用戶端 API，該 API 會根據提供的摘要驗證文件修訂版本。

   主控台會在「驗證結果」卡片中顯示您要求的結果。如需詳細資訊，請參閱驗證結果。
9. 若要測試驗證邏輯，請重複執行「驗證文件修訂」下的步驟 6-8，但變更摘要輸入字串中的單一字元。這會導致您的「驗證」要求失敗，並顯示適當的錯誤訊息。

如果您不再需要使用`vehicle-registration`分類帳，請繼續執行步驟 7 (選擇性)：清除資源。

步驟 7 (選擇性)：清除資源

您可以繼續使用分`vehicle-registration`類帳。但是，如果您不再需要它，則應將其刪除。

如果已啟用對分類帳的刪除保護，您必須先停用它，然後才能使用 QLDB API、AWS Command Line Interface (AWS CLI) 或 QLDB 主控台刪除總帳。

若要刪除分類帳

1. 開啟亞馬遜 QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在導覽窗格中，選擇分類帳。
3. 如果已為此分類帳啟用刪除保護功能，您必須先將其關閉。

   在分類帳清單中，選取`vehicle-registration`，然後選擇「編輯分類帳」。
4. 在 [編輯分類帳] 頁面上，關閉 [刪除保護]，然後選擇 [確認變更]。
5. 在分類帳清單中，`vehicle-registration`再次選取，然後選擇「刪除」。
6. 在提供的欄位**`delete vehicle-registration`**中輸入以確認此操作。

若要進一步了解在 QLDB 中使用總帳，請參閱Amazon QLDB 入門。

Amazon QLDB 入門

如需使用 Amazon QLDB 的詳細資訊，請參閱下列主題：

- 使用亞馬遜 QLDB 中的數據和歷史記錄
- 開始使用 Amazon QLDB 驅動程式
- Amazon QLLDB 並行模型
- 從 Amazon QLDB 匯出日誌資料
- 從 Amazon QLDB 串流日誌資料
- Amazon QLDB 中的數據驗證
- Amazon QLDB 參考

開始使用 Amazon QLDB 驅動程式

本章包含可協助您了解使用 QLDB 驅動程式使用 Amazon QLDB 進行開發套件。此驅動程式建置在 AWS SDK 之上，此 SDK 支援與 QLDB API 互動。

QLDB 工作階段抽象

該驅動程序提供了交易數據 API（QLDB 會話）上方的高級抽象層。它透過管理 SendCommandAPI 呼叫，簡化在分類帳資料上執行 PartiQL 陳述式的程序。這些 API 調用需要驅動程序為您處理的幾個參數，包括會話，事務的管理，並在發生錯誤時重試策略。該驅動程序還具有性能優化，並應用與 QLDB 交互的最佳實踐。

> **Note**
>
> 若要與 Amazon QLDB API 參考中列出的資源管理 API 操作互動，您可以直接使用AWS開發套件而不是驅動程式。管理 API 僅用於管理分類帳資源和非交易資料作業 (例如匯出、串流和資料驗證)。

亞馬遜離子支持

此外，驅動程式使用 Amazon Ion 程式庫提供在執行交易時處理 Ion 資料的支援。這些庫還負責計算離子值的哈希值。QLDB 要求這些離子雜湊來檢查資料交易要求的完整性。

驅動器術語

此工具被稱為驅動程序，因為它與提供開發人員友好界面的其他數據庫驅動程序相媲美。這些驅動程序類似地封裝了邏輯，該邏輯將一組標準的命令和函數轉換為服務的低級 API 所需的特定調用。

此驅動程式是開放原始碼的 GitHub ，可用於下列程式設計語言：

- Java 驅動程式
- 驅動程式
- 移至司機
- Node.js 驅動程式
- Python 驅動程式

如需所有支援之程式設計語言的一般驅動程式資訊以及其他教學課程，請參閱下列主題：

- 驅動程式的工作階段管理
- 驅動程式建議
- 驅動程序重試政策
- 常見錯誤
- 範例應用程式
- 使用 Amazon Amazon Amazon ION
- 取得 PartiQL 陳述式統計

用於 Java 的亞馬遜 QLDB 驅動程序

若要使用總帳中的資料，您可以使用AWS提供的驅動程式，從 Java 應用程式連線到 Amazon QLDB。下列主題說明如何開始使用適用於 Java 的 QLDB 驅動程式。

主題

- 駕駛資源
- 先決條件
- 設定您的預設AWS憑證和區域
- 安裝
- 適用於 Java 的亞馬遜 QLDB 驅動程序 — 快速入門教程
- 適用於 Java 的亞馬遜 QLDB 驅動程序-食譜參考

駕駛資源

如需 Java 驅動程式所支援之功能的詳細資訊，請參閱下列資源：

- 應用程式介面參考：2.x、1.x
- 驅動程式原始程式碼 (GitHub)
- 範例應用程式原始程式碼 (GitHub)
- 分類帳載入程式架構 (GitHub)
- Amazon Ion 程式碼範例

先決條件

開始使用適用於 Java 的 QLDB 驅動程式：

1. 請遵循中的AWS設定指示訪問 Amazon QLDB。這包含下列項目：
    1. 註冊 AWS。
    2. 建立具有適當 QLDB 許可的使用者。
    3. 授予程式設計存取權以進行開發。
2. 透過下載並安裝下列項目來設定 Java 開發環境：
    1. Java SE 開發工具包 8, 如亞馬遜郵輪 8.
    2. （可選）您選擇的 Java 集成開發環境（IDE），例如日蝕或 IntelliJ。
3. 為AWS SDK for Java by 配置您的開發環境設定您的預設AWS憑證和區域。

接下來，您可以下載完整的教學課程範例應用程式 — 或者您可以只在 Java 專案中安裝驅動程式並執行簡短的程式碼範例。

- 若要AWS SDK for Java在現有專案中安裝 QLDB 驅動程式和，請繼續執行安裝。
- 若要設定專案並執行展示分類帳上基本資料交易的簡短程式碼範例，請參閱快速入門教學。
- 若要在完整的教學課程範例應用程式中執行資料和管理 API 作業的更深入範例，請參閱Java 教學.

設定您的預設AWS憑證和區域

QLDB 驅動程式和基礎AWS SDK for Java需要您在執行時間將AWS登入資料提供給您的應用程式。本指南中的程式碼範例假設您正在使用AWS登入資料檔案，如《AWS SDK for Java 2.x開發人員指南》中的設定預設登入資料和 Region 所述。

在這些步驟中，您也應該設定預設值AWS 區域以決定預設 QLDB 端點。程式碼範例會以預設值連線至 QLDBAWS 區域。如需提供 QLDB 的完整區域清單，請參閱中的 Amazon QLDB 端點和配額AWS 一般參考。

以下是名為 `~/.aws/credentials` 的 AWS 登入資料檔案範例，其中波狀符號字元 (~) 代表您的主目錄。

```
[default]
aws_access_key_id = your_access_key_id
```

```
aws_secret_access_key = your_secret_access_key
```

將您自己的AWS認證值替換為##########值。

安裝

QLDB 支援下列 Java 驅動程式版本及其AWS SDK 相依性。

驅動程式版本	AWS SDK	狀態	最新發佈日期
1.	AWS SDK for Java1.	生產發行	2020 年 3 月 20 日
2.x	AWS SDK for Java 2.x	生產發行	2021 年 6 月 4 日

要安裝 QLDB 驅動程序，我們建議使用依賴管理系統，如搖籃或 Maven。例如，將下列成品新增為 Java 專案中的相依性。

2.x

搖籃

在您的build.gradle配置文件中添加此依賴關係。

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

Maven

在您的pom.xml配置文件中添加此依賴關係。

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

```
</dependencies>
```

此成品會自動包含AWS SDK for Java 2.x核心模組、Amazon Ion 程式庫和其他必要的相依性。

1.x

搖籃

在您的build.gradle配置文件中添加此依賴關係。

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

Maven

在您的pom.xml配置文件中添加此依賴關係。

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

此成品會自動包含AWS SDK for Java核心模組、Amazon Ion 程式庫和其他必要的相依性。

> **Important**
>
> Amazon Ion 命名空間 — 在應用程式中匯入 Amazon Ion 類別時，必須使用命名空間下的套件com.amazon.ion。這取AWS SDK for Java決於命名空間下的另一個 Ion 軟件包software.amazon.ion，但這是一個與 QLDB 驅動程序不兼容的舊版軟件包。

如需如何在分類帳上執行基本資料交易的簡短程式碼範例，請參閱食譜參考。

其他可選程式庫

您亦可選擇將下列有用的資源庫新增至專案。這些成品是Java 教學範例應用程式中必要的相依性。

1. aws-java-sdk-qldb— 的 QLDB 模組AWS SDK for Java。QLDB 支援的最低版本為1.11.785。

   在您的應用程式中使用此模組，可直接與中列出的管理 API 作業互動Amazon QLDB API 參考參考。

2. jackson-dataformat-ion— 更快的 XML 的傑克遜數據格式模塊離子. 範例應用程式需要版本2.10.0或更新版本。

Gradle

在build.gradle配置文件中添加這些依賴關係。

```
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

在pom.xml配置文件中添加這些依賴關係。

```
<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>
```

適用於 Java 的亞馬遜 QLDB 驅動程序 — 快速入門教程

在此教學中，您將學習如何使用適用於 Java 的 Amazon QLDB 驅動程式，設定簡單的應用程式。本指南包含安裝驅動程式的步驟，以及基本的建立、讀取、更新和刪除 (CRUD) 操作。如需在完整範例應用程式中示範這些作業的更深入範例，請參閱Java 教學.

主題

先決條件

開始使用之前，請確認執行下列作：開始使用之前，請確認執

1. 完成先決條件 Java 驅動程式 (如果您尚未完成)。這包括註冊AWS、授予程式設計存取以供開發，以及安裝 Java 整合式開發環境 (IDE)。
2. 建立名為的分類帳`quick-start`。

   若要瞭解如何建立分類帳，請參閱Amazon QLDB 分類帳的基本操作或步驟 1：建立新的分類帳在「開始使用主控台」中的。

步驟 1：設定您的 專案

首先，設定您的 Java 專案。我們建議使用 Maven 的依賴管理系統本教程。

> **Note**
>
> 如果您使用的 IDE 具有自動化這些設定步驟的功能，您可以跳到步驟 2：初始化驅動程式初始。

1. 建立您的應用程式資料夾。

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

2. 輸入下列命令以從 Maven 範本初始化您的專案。視需要將*#########*和 *Maven ##*取代為您自己的值。

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

對於*####*，您可以使用基本的 Maven 模板：`maven-archetype-quickstart`

3. 若要將 Java 的 QLDB 驅動程式新增為專案相依性，請瀏覽至新建立的`pom.xml`檔案並新增下列成品。

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

此成品會自動包含AWS SDK for Java 2.x核心模組、Amazon Ion 程式庫和其他必要的相依性。您的`pom.xml`檔案內容看起來如下內容看起來如下內容。

```
<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. 開啟 App.java 檔案。

   然後，在以下步驟中逐步添加代碼示例，以嘗試一些基本的 CRUD 操作。或者，您可以略過 step-by-step 教學課程，而是執行完整的應用程式。

步驟 2：初始化驅動程式初始

初始化連線至名為之分類帳的驅動程式執行個體quick-start。將下列程式碼新增至App.java檔案：將下列程式

```
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();
    }
}
```

步驟 3：建立資料表和索引：建立資料表和

下列程式碼範例示範如何執行CREATE TABLE和CREATE INDEX陳述式。

在main方法中，添加以下代碼，該代碼創建一個名為的表People和該表上的lastName字段的索引。需要索引來最佳化查詢效能，並協助限制最佳化並行控制 (OCC) 衝突例外狀況。

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

步驟 4：插入文件：插入文件

下列程式碼範例示範如何執行INSERT陳述式。QLDB 支援 PartiQL 查詢語言 (SQL 相容) 和亞馬遜離子資料格式 (JSON 的超集)。

添加以下插入文檔到People表中的代碼。

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

此範例使用問號 (?) 做為變數預留位置，將文件資訊傳遞至陳述式。當您使用預留位置時，必須傳遞 type 的值IonValue。

> **Tip**
>
> 若要使用單一INSERT陳述式插入多個文件，您可以將類型的參數 IonList(明確轉換為 aIonValue) 傳遞至陳述式，如下所示。
>
> ```
> // people is an IonList explicitly cast as an IonValue
> txn.execute("INSERT INTO People ?", (IonValue) people);
> ```

傳遞. 時，您不會將變數預留位置 (?) 括在雙角括號 (<<...>>) 中IonList。在手動 PartiQL 陳述式中，雙角括號表示稱為袋子的無序集合。

步驟 5：查詢文件查詢文件查

下列程式碼範例示範如何執行SELECT陳述式。

添加以下代碼查詢從People表中的文檔。

```
// 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
});
```

步驟 6：更新文件更新文件更

下列程式碼範例示範如何執行UPDATE陳述式。

1. 添加以下代碼，通過更新age到更新People表中的文檔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. 再次查詢文件以查看更新的值。

   ```
   // 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. 使用 Maven 或您的 IDE 來編譯和運行該App.java文件。

執行完整的應用程式

下列程式碼範例是應App.java用程式的完整版本。您也可以從頭到尾複製並執行此程式碼範例，而不是個別執行前述步驟。此應用程序演示了名為分類帳的一些基本 CRUD 操作quick-start。

> **Note**
>
> 在執行此程式碼之前，請確定您尚未在quick-start分類帳People中命名為使用中的資料表。
>
> 在第一行中，將####取代為中用於 Maven 命令的groupId值步驟 1：設定您的 專案。

```
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
        });
    }
}
```

使用 Maven 或您的 IDE 來編譯和運行該App.java文件。

適用於 Java 的亞馬遜 QLDB 驅動程序-食譜參考

本參考指南顯示適用於 Java 的 Amazon QLDB 驅動程式的常見使用案例。Aon 程式碼範例會示範如何使用驅動程式來運行基本的建立、讀取、更新和刪除 (CRUD) 操作。它還包括用於處理 Amazon Ion 資料的程式碼範例。此外，本指南還重點介紹了使交易冪等和實施唯一性約束的最佳實踐。

> **Note**
>
> 在適用的情況下，某些使用案例會針對每個支援的 Java QLDB 驅動程式主要版本提供不同的程式碼範例。

內容

匯入驅動程式

下列程式碼範例會匯入驅動程式、QLDB 工作階段用戶端、Amazon Ion 套件及其他相關相依性。

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;
```

實例化驅動程式

下列程式碼範例會建立連線至指定分類帳名稱的驅動程式執行環境，並使用具有自訂重試限制的指定重試邏輯。

> **Note**
>
> 此範例也會實體化 Amazon 離子系統物件 (IonSystem)。在此參考中執行某些資料作業時，您需要此物件來處理 Ion 資料。如需進一步了解，請參閱 使用 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();
```

CRUD 操作

QLDB 運行建立、讀取、更新及刪除 (CRUD) 操作，作為交易的組成部分。

> **Warning**
>
> 根據最佳實務，寫入交易嚴格的冪等。

使交易冪等

我們建議您將寫入事務設為冪等，以避免在重試的情況下出現任何意外的副作用。如果事務可以運行多次並每次產生相同的結果，則事務是冪等的。

例如，假設將文件插入名為的資料表中的交易Person。事務應該首先檢查該文檔是否已經存在於表中。如果沒有此檢查，表格最終可能會出現重複的文件。

假設 QLDB 成功地在服務器端提交事務，但在等待響應的客戶端超時。如果事務不是冪等的，則在重試的情況下，可以多次插入相同的文檔。

使用索引來避免全表掃描

我們也建議您在索引欄位或文件 ID 上使用相等運算子來執行具有述WHERE詞子句的陳述式；例如，WHERE indexedField = 123或WHERE indexedField IN (456, 789)。如果沒有此索引查找，QLDB 需要進行表掃描，這可能會導致交易逾時或樂觀並發控制（OCC）衝突。

如需 OCC 的詳細資訊，請參閱Amazon QLLDB 並行模型。

隱含建立的交易

QldbDriver.execute 方法接受一個 lambda 函數，該函數接收執行程序的實例，您可以使用它來運行語句。Executor執行個體會封裝隱含建立的交易。

您可以使用方法在 lambda 函數中執行陳述Executor.execute式。當 lambda 函數返回時，驅動程序隱式地提交交易。

下列各節說明如何執行基本 CRUD 作業、指定自訂重試邏輯，以及實作唯一性限制。

> **Note**
>
> 在適用的情況下，這些區段提供使用內建 Ion 程式庫和 Jackson Ion 映射程式庫來處理 Amazon Ion 資料的程式碼範例。如需進一步了解，請參閱 使用 Amazon Ion。

內容

- 刪除文件
- 在交易中執行多個陳述式
- Retry 邏輯
- 實現唯一性限制

建立資料表

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

建立索引

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

閱讀文件

```
// 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
});
```

使用查詢參數

下列程式碼範例會使用 Ion 型別查詢參數。

```
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
});
```

下列程式碼範例會使用多個查詢參數。

```
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
});
```

下列程式碼範例使用查詢參數清單。

```
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
});
```

使用傑克遜映射器

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

使用查詢參數

下列程式碼範例會使用 Ion 型別查詢參數。

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

下列程式碼範例會使用多個查詢參數。

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

下列程式碼範例使用查詢參數清單。

```
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**
>
> 當您在沒有索引查閱的情況下執行查詢時，會叫用完整資料表掃描。在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，查詢可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

插入文件

下列程式碼範例會插入 Ion 資料型別。

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

使用傑克遜映射器

下列程式碼範例會插入 Ion 資料型別。

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

此事務將一個文檔插入到Person表中。在插入之前，它會先檢查文件是否已存在於表格中。這個檢查使事務本質上是冪等的。即使您多次運行此事務，也不會造成任何意外的副作用。

> **Note**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

在一個語句中插入多個文檔

若要使用單一INSERT陳述式插入多個文件，您可以將類型的參數 IonList(明確轉換為 aIonValue) 傳遞至陳述式，如下所示。

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

傳遞的時候，您不會將變數預留位置 (?) 括在雙角括號 (<<...>>) 中IonList。在手動 PartiQL 陳述式中，雙角括號表示稱為袋子的無序集合。

為什麼需要明確的演員？

該 TransactionExecutor.execute 方法被重載。它接受可變數量的IonValue參數（可變參數）或單個List<IonValue>參數。在離子 Java 中，IonList被實現為一個List<IonValue>.

當您調用重載方法時，Java 默認為最具體的方法實現。在這種情況下，當您傳遞IonList參數時，它會預設為採用List<IonValue>. 調用時，此方法實現將列表中的IonValue元素作為不同的值傳遞。因此，要調用可變參數方法，您必須明確地將IonList參數轉換為IonValue.

更新文件

```
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);
});
```

使用傑克遜映射器

```
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**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

刪除文件

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

使用傑克遜映射器

```
qldbDriver.execute(txn -> {
    try {
        txn.execute("DELETE FROM Person WHERE GovId = ?",
```

```
                MAPPER.writeValueAsIonValue("TOYENC486FH"));
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

> **Note**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

在交易中執行多個陳述式

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

Retry 邏輯

驅動程式的execute方法具有內建的重試機制，可在發生可重試的例外狀況 (例如逾時或 OCC 衝突) 時重試交易。

2.x

重試試試試試試試試試試試試試試試試試試試試試試試試試試試試試試試試試試試試

預設重試限制為4，且預設輪詢策略為DefaultQldbTransactionBackoffStrategy。您可以使用的執行個體來設定每個驅動程式執行個體以及每個交易的重試組態RetryPolicy。

下列程式碼範例會指定具有自訂重試限制的重試邏輯，以及驅動程式執行個體的自訂輪詢策略。

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

下列程式碼範例會指定具有自訂重試限制的重試邏輯，以及特定交易的自訂輪詢策略。此組態execute會覆寫為驅動程式執行個體設定的重試邏輯。

```
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

重試試試試試試試試試試試試試試試試試 您可以在初始化時設定`retryLimit`屬性來設定重試限制`PooledQldbDriver`。

預設重試限制為4。

實現唯一性限制

QLDB 不支援唯一索引，但您可以在應用程式中實作此行為。

假設您要在`Person`表中的`GovId`字段上實現唯一性約束。要做到這一點，你可以寫入執行下列動作的交易：

1. 斷言該表沒有具有指定的現有文檔`GovId`。
2. 插入文檔，如果斷言通過。

如果競爭交易同時通過斷言，則只有其中一個交易將成功提交。另一個交易將會失敗，並出現 OCC 衝突例外狀況。

下列程式碼範例顯示如何實現此唯一性限制條件限制邏輯。

```
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**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

使用 Amazon Ion

QLDB 中的 Amazon Ion Ion 數據可透過多種方式處理 QLDB Ion 數據。您可以使用 Ion 庫中的內置方法，根據需要靈活地創建和修改文檔。或者，您可以使用 FastterXML 的傑克遜數據格式模塊來將離子文檔映射到普通的舊 Java 對象（POJO）模型。

以下各節提供使用這兩種技術處理離子資料的程式碼範例。

內容

匯入離子套件

將工件離子 Java 添加為 Java 項目中的依賴項。

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>
```

匯入下列 Ion 套件。

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

使用傑克遜映射器

將工件添加jackson-dataformat-ion為 Java 項目中的依賴項。QLDB 需要版本2.10.0或更新版本。

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>
```

匯入下列 Ion 套件。

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

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

Ion 組化 Ion

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

使用傑克遜映射器

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

Ion on 對象

下列程式碼範例會使用IonStruct介面及其內建方法來建立 Ion 物件。

```
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
```

使用傑克遜映射器

假設您有名為 JSON 對應的模型類別Person，如下所示。

```
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;
```

```
    }
}
```

下列程式碼範例會從的執行個體建立IonStruct物件Person。

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

Ion on on 對象

下列程式碼範例會列印執行個ionStruct體的每個欄位。

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

使用傑克遜映射器

下列程式碼範例會讀取IonStruct物件，並將其對應至的執行個體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
```

如需有關使用 Ion 的詳細資訊，請參閱上的 Amazon Ion 文件 GitHub。如需在 QLDB 中使用 Ion 的更多程式碼範例，請參閱在亞馬遜 QLDB 中使用亞馬遜離子數據類型。

用於. NET 的亞馬遜 QLDB 驅動程序

若要使用分類帳中的資料，您可以使用AWS提供的驅動程式，從 Microsoft .NET 應用程式連線到 Amazon QLDB。該驅動程序以 .NET 標準 2.0 為目標。更具體地說，它支持. NET 核心（LTS）2.1 + 和 .NET 框架 4.5.2 +。如需相容性的詳細資訊，請參閱微軟文件網站上的 .NET 標準。

我們強烈建議您使用 Ion 物件對應程式，完全避免在 Amazon Ion 類型和原生 C# 類型之間手動轉換的需求。

下列主題說明如何開始使用 NET 驅動程式 QLDB NET 驅動程式。

主題

驅動資源

如需 NET 驅動程式所支援功能的詳細資訊，請參閱下列資源：

- API 參考
- 驅動程式原始程式碼 (GitHub)
- 範例應用程式原始程式碼 (GitHub)
- 亞馬遜離子食譜
- 離子對象映射器 (GitHub)

先決條件

您必須執行以下作業：業：

1. 按照中的AWS設定指示進行操作訪問 Amazon QLDB。這包含下列項目：
    1. 註冊 AWS。
    2. 建立具有適當 QLDB 許可的使用者。
    3. 授予程式設計存取權的開發存取權
2. 從微軟. NET 下載網站下載並安裝. NET 核心 SDK 2.1 版或更新版本。
3. (選擇性) 安裝您選擇的整合式開發環境 (IDE)，例如視覺工作室、Mac 版視覺工作室或視覺工作室程式碼。您可以從微軟視覺工作室網站下載這些。
4. 為下列項目設定您的開發環境 AWS SDK for .NET：
    1. 設定您的AWS認證。建立共用憑證檔案。

        如需指示，請參閱AWSAWS SDK for .NET開發人員指南中的使用認證檔案設定認證。

2. 設定您的預設值AWS 區域。若要了解如何作業，請參閱AWS 區域選取項

   如需可用區域的完整清單，請參閱中的 Amazon QLDB 端點和配額AWS 一般參考。

接下來，您可以設定基本範例應用程式並執行簡短程式碼範例，或者您也可以在現有的 .NET 專案中安裝驅動程式。

- 若要AWS SDK for .NET在現有專案中安裝 QLDB 驅動程式和，請繼續執行安裝。
- 若要設定專案並執行展示分類帳上基本資料交易的簡短程式碼範例，請參閱Quick 教學課程。

安裝

使用 NuGet 套件管理員安裝 .NET 的 QLDB 驅動程式。我們建議您使用 Visual Studio 或您選擇的 IDE 來新增專案相依性。驅動程式套件名稱是亞馬遜 .qldb.driver。

例如，在 Visual Studio 中，開啟 [工具] 功能表上的 [P NuGet ackage 管理員主控台]。然後，在提示中輸入以下命PM>令。

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

安裝驅動程式也會安裝其相依性，包括AWS SDK for .NET和 Amazon Ion 套件。

安裝離子對象映射器

適用於 .NET 的 QLDB 驅動程式 1.3.0 版引入了接受和傳回原生 C# 資料類型的支援，而不需要使用 Amazon Ion。要使用此功能，請將以下軟件包添加到您的項目中。

- 序列化-一個庫，可以將離子值映射到 C # 普通的舊 CLR 對象（POCO），以及相反的方式。這個 Ion 物件對應程式可讓您的應用程式直接與原生 C# 資料類型互動，而不需要使用 Ion。有關如何使用此庫的簡短指南，請參閱 GitHub 存儲庫中的序列化 .md 文件`awslabs/amazon-qldb-driver-dotnet`。

若要安裝此套件，請輸入以下命令。

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

如需如何在分類帳上執行基本資料交易的簡短程式碼範例，請參閱食譜參考。

適用於. NET 的亞馬遜 QLDB 驅動程序-快速入門教程

在此教學中，您將學習如何使用適用於 .NET 的 Amazon QLDB 驅動程式設定簡單的應用程式。本指南包含安裝驅動程式的步驟和基本的建立、讀取、更新和刪除 (CRUD) 操作的簡短程式碼範例。

主題

先決條件

在開始使用之前，請確認執行下列作業：

1. 如果您尚未這麼做，請完成 .NET 驅動程式。先決條件這包括註冊AWS、授予程式設計存取以供開發，以及安裝 .NET Core SDK。
2. 建立名為的分類帳`quick-start`。

   若要瞭解如何建立分類帳，請參閱Amazon QLDB 分類帳的基本操作或步驟 1：建立新的分類帳在「開始使用主控台」中的。

步驟 1：設定您的 專案

首先，設定您的 .NET 專案。

1. 若要建立並執行範本應用程式，請在終端機 (例如 bash、PowerShell或命令提示字元) 上輸入下列*`dotnet`*命令。

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

此範本會建立名為的資料夾`Amazon.QLDB.QuickStartGuide`。在該文件夾中，它創建一個具有相同名稱的項目和一個名為的文件`Program.cs`。該程序包含顯示輸出的代碼`Hello World!`。

2. 使用 NuGet 套件管理員安裝 .NET 的 QLDB 驅動程式。我們建議您使用 Visual Studio 或您選擇的 IDE，將相依性新增至您的專案。驅動程式套件名稱是亞馬遜 .qldb.driver。

   - 例如，在 Visual Studio 中，開啟 [工具] 功能表上的 [P NuGet ackage 管理員主控台]。在提示中輸入下列命`PM>`令。

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

   - 或者，您可以在終端機上輸入下列命令。

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

   安裝驅動程序還會安裝其依賴項，包括AWS SDK for .NET和 Amazon Ion 庫。

3. 安裝驅動程式的序列化程式庫。

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

4. 開啟 `Program.cs` 檔案。

   然後，在以下步驟中逐步添加代碼示例，以嘗試一些基本的 CRUD 操作。或者，您可以略過 step-by-step 教學課程，而是執行完整的應用程式。

> **Note**
>
> - 在同步和非同步 API 之間進行選擇 — 驅動程式提供同步和非同步 API。對於處理多個要求而不會封鎖的高需求應用程式，我們建議使用非同步 API 來改善效能。該驅動程序提供了同步 API，作為同步編寫的現有代碼庫的更多便利性。
>
>   本教學課程包含同步和非同步程式碼範例。如需 API 的詳細資訊，請參閱 API 文件中的 IQldbDriver 和 IAsyncQldbDriver 介面。
> - 處理 Amazon Ion 資料 — 本教學提供預設使用 Ion 物件對應程式處理 Amazon Ion 資料的程式碼範例。QLDB 在 .NET 驅動程式 1.3.0 版中引入了離子物件對應程式。在適用的情況

下，本教程還提供了使用標準 Ion 庫作為替代方案的代碼示例。如需進一步了解，請參閱 使用 Amazon Ion。

步驟 2：初始化驅動程式

初始化連線至名為之分類帳之驅動程式的執行環境quick-start。將下列程式碼新增至Program.cs檔案：

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();
        }
    }
}
```

使用離子庫

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();
        }
    }
```

```
}
```

步驟 3：建立資料表和索引

對於本教學課程透過步驟 6 的其餘部分，您需要將下列程式碼範例附加到前一個程式碼範例中。

在這個步驟中，下面的代碼演示了如何運行CREATE TABLE和CREATE INDEX語句。它創建一個名為表Person和該表上的firstName字段的索引。需要索引來最佳化查詢效能，並協助限制最佳化並行控制 (OCC) 衝突例外狀況。

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)");
});
```

步驟 4：插入文件

下列程式碼範例示範如何執行INSERT陳述式。QLDB 支援 PartiQL 查詢語言 (與 SQL 相容) 和亞馬遜離子資料格式 (JSON 的超集)。

添加下面的代碼，插入一個文檔到Person表中。

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);
});
```

使用離子庫

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

若要使用單一INSERT陳述式插入多個文件，您可以將 Ion 清單型別參數傳遞至陳述式，如下所示。

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

傳遞離子清單時，您不會將變數預留位置 (?<<...>>) 括在雙尖括號 () 中。在手動 PartiQL 陳述式中，雙角括號表示稱為袋子的無序集合。

步驟 5：查詢文件

下列程式碼範例示範如何執行SELECT陳述式。

添加以下代碼查詢從Person表中的文檔。

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
}
```

使用離子庫

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);
}
```

此範例使用問號 (?) 做為變數預留位置，將文件資訊傳遞至陳述式。當您使用預留位置時，您必須傳遞 type 的值IonValue。

步驟 6：更新文件

下列程式碼範例示範如何執行UPDATE陳述式。

1. 添加以下代碼，通過更新age到 42 更新Person表中的文檔。

   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. 再次查詢文件以查看更新的值。

   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. 若要執行應用程式，請從Amazon.QLDB.QuickStartGuide專案目錄的父目錄中輸入下列指令。

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

使用離子庫

1. 添加以下代碼，通過更新age到 42 更新Person表中的文檔。

   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. 再次查詢文件以查看更新的值。

   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. 若要執行應用程式，請從Amazon.QLDB.QuickStartGuide專案目錄的父目錄中輸入下列指令。

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

執行完整的應用程式

下列程式碼範例是應Program.cs用程式的完整版本。您也可以從頭到尾複製並執行此程式碼範例，而不是個別執行前述步驟。此應用程序演示了名為的分類帳上的一些基本 CRUD 操作quick-start。

> **Note**
>
> 在執行此程式碼之前，請確定您尚未在quick-start分類帳Person中命名為使用中的資料表。

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
            }

        }
    }
}
```

使用離子庫

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);
            }
        }
    }
}
```

若要執行完整的應用程式，請從Amazon.QLDB.QuickStartGuide專案目錄的父目錄中輸入下列指令。

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

適用於. NET 的亞馬遜 QLDB 驅動程序-食譜參考

本參考指南顯示適用於 .NET 的 Amazon QLDB 驅動程式的常見使用案例。C# 程式碼範例會示範如何使用該驅動程式，執行基本的建立、讀取、更新和刪除 (CRUD) 操作。它還包括用於處理 Amazon Ion 資料的程式碼範例。此外，本指南還重點介紹了使交易冪等和實施唯一性約束的最佳實踐。

> **Note**
>
> 本主題提供預設使用 Ion 物件對應程式處理 Amazon Ion 資料的程式碼範例。QLDB 在 .NET 驅動程式 1.3.0 版中引入了離子物件對應程式。在適用的情況下，本主題也會提供使用標準 Ion 程式庫做為替代方案的程式碼範例。如需進一步了解，請參閱 使用 Amazon Ion。

內容

匯入驅動程式

下列程式碼範例會匯入驅動程式。

```
using Amazon.QLDB.Driver;
```

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

使用離子庫

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

實例化驅動程式

下列程式碼範例會建立使用預設設定連線至指定分類帳名稱的驅動程式執行環境。

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();
```

使用離子庫

Async

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

Sync

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

CRUD 操作

QLDB 運行建立、讀取、更新及刪除 (CRUD) 操作。

> **Warning**
>
> 根據最佳實務，使寫入交易嚴格的冪等。

使交易冪等

我們建議您將寫入事務設為冪等，以避免在重試的情況下出現任何意外的副作用。如果事務可以運行多次並每次產生相同的結果，則事務是冪等的。

例如，假設將文件插入名為的資料表中的交易`Person`。事務應該首先檢查該文檔是否已經存在於表中。如果沒有此檢查，表格最終可能會出現重複的文件。

假設 QLDB 成功地在服務器端提交事務，但在等待響應的客戶端超時。如果事務不是冪等的，則在重試的情況下，可以多次插入相同的文檔。

使用索引來避免全表掃描

我們也建議您在索引欄位或文件 ID 上使用相等運算子來執行具有述`WHERE`詞子句的陳述式；例如，`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`。如果沒有此索引查找，QLDB 需要進行表掃描，這可能會導致交易逾時或樂觀並發控制（OCC）衝突。

如需 OCC 的詳細資訊，請參閱Amazon QLLDB 並行模型。

隱含建立的交易

`QldbDriver`執行方法接受接收亞馬遜的實例的 lambda 函數。 `TransactionExecutor`，您可以使用它來執行陳述式。`TransactionExecutor`封裝隱含建立之交易的執行個體。

您可以使用交易執行程`Execute`式的方法，在 lambda 函數中執行陳述式。當 lambda 函數返回時，驅動程序隱式地提交交易。

下列各節說明如何執行基本 CRUD 作業、指定自訂重試邏輯，以及實作唯一性限制。

內容

建立資料表

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 }
}
```

使用離子庫

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"
    // }
}
```

建立索引

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 }
}
```

使用離子庫

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"
    // }
}
```

閱讀文件

```
// 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**
>
> 當您在沒有索引查閱的情況下執行查詢時，會叫用完整資料表掃描。在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，查詢可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

使用查詢參數

下列程式碼範例會使用 C# 型別查詢參數。

```
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.
}
```

下列程式碼範例會使用多個 C# 型別查詢參數。

```
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.
}
```

下列程式碼範例會使用 C# 型別查詢參數的陣列。

```
// 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.
}
```

下列程式碼範例會使用 C# 清單做為值。

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

使用離子庫

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**
>
> 當您在沒有索引查閱的情況下執行查詢時，會叫用完整資料表掃描。在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，查詢可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

下列程式碼範例會使用 Ion 型別查詢參數。

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.
}
```

下列程式碼範例會使用多個查詢參數。

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.
}
```

下列程式碼範例使用查詢參數清單。

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.
}
```

下列程式碼範例會使用 Ion 清單做為值。若要進一步了解使用不同 Ion 類型，請參閱[在亞馬遜 QLDB 中使用亞馬遜離子數據類型](#)。

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"
    //     }
    //     ]
    // }
}
```

插入文件

下列程式碼範例會插入 Ion 資料型別。

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

使用離子庫

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);
});
```

此事務將一個文檔插入到Person表中。在插入之前，它會先檢查文件是否已存在於表格中。這個檢查使事務本質上是冪等的。即使您多次運行此事務，也不會造成任何意外的副作用。

> **Note**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

在一個語句中插入多個文檔

若要使用單一INSERT陳述式插入多個文件，您可以將 C#List 參數傳遞至陳述式，如下所示。

```
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 }
}
```

使用離子庫

要通過使用單個INSERT語句插入多個文檔，可以傳遞離子列表類型的參數到語句如下。

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"
    // }
}
```

傳遞離子清單時，您不會將變數預留位置 (?<<...>>) 括在雙尖括號 () 中。在手動 PartiQL 陳述式中，雙角括號表示稱為袋子的無序集合。

更新文件

```
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 }
}
```

使用離子庫

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**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

刪除文件

```
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 }
}
```

使用離子庫

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**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

在交易中執行多個陳述式

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

使用離子庫

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;
    });
}
```

重試邏輯

如需驅動程式內建重試邏輯的詳細資訊，請參閱瞭解亞馬遜 QLDB 中的驅動程序的重試策略。

實行唯一性限制

QLDB 不支援唯一索引，但您可以在應用程式中實作此行為。

假設您要在Person表中的GovId字段上實現唯一性約束。您可通過下列方式執行此操作：

1. 斷言該表沒有具有指定的現有文檔GovId。
2. 插入文檔，如果斷言通過。

如果競爭交易同時通過斷言，則只有其中一個交易將成功提交。另一個交易將會失敗，並出現 OCC 衝突例外狀況。

下列程式碼範示範如何實現此唯一性限制。

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

使用離子庫

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**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

使用 Amazon Ion

QLDB 中的 Amazon Ion 資料可透過多種方式處理。您可以使用 I on 元件庫來建立和修改 Ion 值。或者，您可以使用 Ion 對象映射器將 C# 普通的舊 CLR 對象（POCO）映射到 Ion 值和離子值。適用於 .NET 的 QLDB 驅動程式 1.3.0 版引入了對離子物件對應程式的支援。

以下各節提供使用這兩種技術處理離子資料的程式碼範例。

內容

- 匯入離子模組
- 建立 Ion 類型
- 獲取離子二進制轉儲
- 獲取離子文本轉儲

匯入離子模組

```
using Amazon.IonObjectMapper;
```

使用離子庫

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

建立 Ion 類型

下列程式碼範示範如何使用 Ion 物件對應程式，從 C# 物件。

```
// 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);
```

使用離子庫

下列程式碼範例會示範使用 Ion 程式庫建立 Ion 值的兩種方式。

使用 **`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);
```

使用 `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);
```

獲取離子二進制轉儲

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

使用離子庫

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

獲取離子文本轉儲

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

使用離子庫

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

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

如需有關使用 Ion 的詳細資訊，請參閱上的 Amazon Ion 文件 GitHub。如需在 QLDB 中使用 Ion 的更多程式碼範例，請參閱在亞馬遜 QLDB 中使用亞馬遜離子數據類型。

適用於 Go 的亞馬遜 QLDB 驅動程序

若要使用總帳中的資料，您可以使用AWS提供的驅動程式，從 Go 應用程式連線到 Amazon QLDB。下主題說明如何開始使用 Go 的 QLDB 驅動程式。

主題

- 駕駛員資源
- 先決條件
- 安裝
- 適用於 Go 的亞馬遜 QLDB 驅動程序 — 快速入門教程
- 適用於 Go 的亞馬遜 QLDB 驅動程序-食譜參考

駕駛員資源

如需 Go 驅動程式支援功能的詳細資訊，請參閱以下資源：

- 應用程式介面參考：3.x 、2.x、1.x
- 驅動程式原始程式碼 (GitHub)
- 亞馬遜離子食譜

先決條件

開始使用 Go 的 QLDB 驅動程式：

1. 按照中的AWS設定指示進行操作訪問 Amazon QLDB。這包含下列項目：
    1. 註冊 AWS。
    2. 建立具有適當 QLDB 權限的使用者。
    3. 授予程式設計存取權以開發作業。

2. (選用) 安裝您選擇的整合式開發環境 (IDE)。如需 Go 常用 IDE 的清單，請參閱 Go 網站上的編輯器外掛程式和 IDE。
3. 從 Go 下載網站下載並安裝下列其中一個 Go 版本：
   - 1.15 或更高版本 — 適用於 Go v3 的 QLDB 驅動程式
   - 1.14 — 適用於轉到 V1 或 V2 的 QLDB 驅動程式
4. 為下列項目設定您的開發環境 AWS SDK for Go：
   1. 設定您的AWS認證。建議建立共用憑證檔案。

      如需指示，請參閱AWS SDK for Go開發人員指南中的指定認證。
   2. 設定您的預設值AWS 區域。若要瞭解如何操作，請參閱指定AWS 區域.

      如需可用區域的完整清單，請參閱中的 Amazon QLDB 端點和配額AWS 一般參考。

接下來，您可以設定基本範例應用程式並執行簡短程式碼範例，或者您也可以在現有的 Go 專案中安裝驅動程式。

- 若要AWS SDK for Go在現有專案中安裝 QLDB 驅動程式和，請繼續執行安裝。
- 若要設定專案並執行展示分類帳上基本資料交易的簡短程式碼範例，請參閱快速入門教學。

安裝

Go 的 QLDB 驅動程式在 GitHub 儲存庫 awslab/ 中是開放原始碼的amazon-qldb-driver-go。QLDB 支援下列驅動程式版本及其 Go 相依性。

驅動程式版本	圍棋版	狀態	最新發佈日期
1.	1.14 或更新版本	生產發行	2021 年 6 月 16 日
2.x	1.14 或更新版本	生產發行	2021 年 7 月 21 日
3.x	1.15 或更新版本	生產發行	2022 年 11 月 10 日

安裝驅動程式

1. 確保您的項目正在使用 Go 模塊來安裝項目依賴項。

2. 在您的項目目錄中，輸入以下go get命令。

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
```

安裝驅動程式也會安裝其相依性，包括AWS SDK for Go或 AWS SDK for Gov2 和 Amazon Ion 套件。

如需如何在分類帳上執行基本資料交易的簡短程式碼範例，請參閱食譜參考書參考。

適用於 Go 的亞馬遜 QLDB 驅動程序 — 快速入門教程

在此教學中，您將學習如何使用適用於 Go 的 Amazon QLDB 驅動程式，設定簡單的應用程式。該指南包含安裝驅動程式的步驟和基本的建立、讀取、更新和刪除 (CRUD) 操作。

主題

先決條件

在開始使用之前，請確認執行下列作業：

1. 完成先決條件 Go 驅動程式 (如果您尚未這麼做) 這包括註冊AWS，授予用於開發的程序化訪問權限以及安裝 Go。
2. 建立名為的分類帳quick-start。

   若要瞭解如何建立分類帳，請參閱Amazon QLDB 分類帳的基本操作或步驟 1：建立新的分類帳在「開始使用主控台」中的。

步驟 1：安裝驅動程式

確保您的項目正在使用 Go 模塊來安裝項目依賴項。

在您的項目目錄中，輸入以下go get命令。

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

安裝驅動程式也會安裝其相依性，包括 AWS SDK for Gov2 和 Amazon Ion 套件。

步驟 2：匯入套件

匯入下列AWS套件。

```
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"
)
```

步驟 3：初始化驅動程式

初始化連線至名為之分類帳的驅動程式執行個體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**
>
> 在此程式碼範例中，將 *us-east-1* 取代為AWS 區域您建立分類帳的位置。

步驟 4：建立資料表和索引

下列程式碼範例示範如何執行CREATE TABLE和CREATE INDEX陳述式。

```
_, 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)
}
```

此程式碼會建立名為的資料表People，以及該資料表上firstName和age欄位的索引。需要索引來最佳化查詢效能，並協助限制最佳化並行控制 (OCC) 衝突例外狀況。

步驟 5：插入文件

下列程式碼範例示範如何執行INSERT陳述式。QLDB 支援 PartiQL 查詢語言 (SQL 相容) 和亞馬遜離子資料格式 (JSON 的超集)。

使用文字 PartiQL

下列程式碼會使用字串常值 PartiQL 陳述式，將文件插入People資料表中。

```
_, 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)
}
```

使用離子資料類型

與 Go 的內置 JSON 包類似，您可以將 Go 數據類型編組和解組在 Ion 之間。

1. 假設您已命名為以下 Go 結構Person。

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

2. 建立 Person 的執行個體。

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

驅動程式會為您封送離子編碼person的文字表示。

> **Important**
>
> 為了使元帥和解組正常工作，必須導出 Go 數據結構的字段名稱（首字母大寫）。

3. 將person實例傳遞給交易的Execute方法。

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

此範例使用問號 (?) 做為變數預留位置，將文件資訊傳遞至陳述式。使用預留位置時，必須傳遞離子編碼的文字值。

> **Tip**
>
> 要通過使用單個INSERT語句插入多個文檔，你可以通過類型列表的參數到語句如下。
>
> ```
> // people is a list
> txn.Execute("INSERT INTO People ?", people)
> ```
>
> 傳遞清單時，您不會將變數預留位置 (?<<...>>) 括在雙尖括號 () 中。在手動 PartiQL 陳述式中，雙角括號表示稱為袋子的無序集合。

步驟 6：查詢文件

下列程式碼範例示範如何執行SELECT陳述式。

```
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")
}
```

這個範例會從People資料表查詢您的文件，假設結果集不是空的，並從結果傳回您的文件。

步驟 7：更新文件

下列程式碼範例示範如何執行UPDATE陳述式。

```
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)
}
```

步驟 8：查詢更新的文件

下列程式碼範例會依據查詢People資料表，firstName並傳回結果集中的所有文件。

```
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")
}
```

步驟 9：放下資料

下列程式碼範例示範如何執行DROP TABLE陳述式。

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

執行完整的應用程式

下列程式碼範例是應用程式的完整版本。您也可以從頭到尾複製並執行此程式碼範例，而不是個別執行前述步驟。此應用程序演示了名為分類帳的一些基本 CRUD 操作quick-start。

> **Note**
>
> 在執行此程式碼之前，請確定您尚未在quick-start分類帳People中命名為使用中的資料表。

```
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)
    }
}
```

適用於 Go 的亞馬遜 QLDB 驅動程序-食譜參考

本參考指南顯示適用於 Go 的 Amazon QLDB 驅動程式的常見使用案例。Go 程式碼範例會示範如何使用驅動程式碼執行基本的建立、讀取、更新和刪除 (CRUD) 操作。它還包括用於處理 Amazon Ion 資料的程式碼範例。此外，本指南還重點介紹了使交易冪等和實施唯一性約束的最佳實踐。

> **Note**
>
> 在適用的情況下，某些使用案例針對 Go QLDB 驅動程式的每個支援主要版本都有不同的程式碼範例。

內容

匯入驅動程式驅

下列程式碼範例會匯入驅動程式和其他必要的AWS套件。

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**
>
> 此範例也會匯入 Amazon Ion 套件 (amzn/ion-go/ion)。在此參考中執行某些資料作業時，您需要此套件來處理 Ion 資料。如需進一步了解，請參閱 使用 Amazon Ion Ion Ion。

實例化驅動程式

下列程式碼範例會建立連線至指定分類帳名稱中之指定分類帳名稱的驅動程式執行環境AWS 區域。

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)
}
```

CRUD 操作

QLDB 運行建立、讀取、更新及刪除 (CRUD) 操作。

> **Warning**
>
> 根據最佳實務，寫入交易必須嚴格使用冪等的寫入交易。

使交易冪等

我們建議您將寫入事務設為冪等，以避免在重試的情況下出現任何意外的副作用。如果事務可以運行多次並每次產生相同的結果，則事務是冪等的。

例如，假設將文件插入名為的資料表中的交易Person。事務應該首先檢查文檔是否已經存在於表中。如果沒有此檢查，表格最終可能會出現重複的文件。

假設 QLDB 成功地在服務器端提交事務，但在等待響應的客戶端超時。如果事務不是冪等的，則在重試的情況下，可以多次插入相同的文檔。

使用索引來避免全表掃描

我們也建議您在索引欄位或文件 ID 上使用相等運算子來執行具有述WHERE詞子句的陳述式；例如，WHERE indexedField = 123或WHERE indexedField IN (456, 789)。如果沒有此索引查找，QLDB 需要進行表掃描，這可能會導致交易逾時或樂觀並發控制（OCC）衝突。

如需的詳細資訊，請參閱Amazon QLLDB 並行模型。

隱含建立的交易

該 QLDB 驅動程序。執行函數接受接收事務的實例，您可以使用它來運行語句的 lambda 函數。Transaction封裝隱含建立之交易的執行個體。

您可以使用函數在 lambda 函數中執行陳述式。Transaction.Execute當 lambda 函數返回時，驅動程序隱式地提交交易。

下列各節說明如何執行基本 CRUD 作業、指定自訂重試邏輯，以及實作唯一性限制。

內容

建立資料表

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

建立索引的

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

閱讀文件

```
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)
}
```

使用查詢參數參

下列程式碼範例會使用原生型別查詢參數。

```
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)
}
```

下列程式碼範例會使用多個查詢參數。

```
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)
}
```

下列程式碼範例使用查詢參數清單。

```
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**
>
> 當您在沒有索引查閱的情況下執行查詢時，會叫用完整資料表掃描。在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，查詢可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

插入文件檔

下列程式碼範例會插入原生資料類型。

```
_, 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
})
```

此事務將一個文檔插入到Person表中。在插入之前，它會先檢查文件是否已存在於表格中。這個檢查使事務本質上是冪等的。即使您多次運行此事務，也不會造成任何意外的副作用。

> **Note**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

在一個語句中插入多個文檔

要通過使用單個INSERT語句插入多個文檔，你可以通過類型列表的參數到語句如下。

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

傳遞清單時，您不會將變數預留位置 (?<<...>>) 括在雙尖括號 () 中。在手動 PartiQL 陳述式中，雙角括號表示稱為袋子的無序集合。

更新文件

下列程式碼範例會使用原生資料類型。

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

> **Note**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

刪除文件

下列程式碼範例會使用原生資料類型。

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

> **Note**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

在交易中執行多個陳述式

```
// 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
}
```

重試次數的

驅動程式的Execute函數具有內建的重試機制，可在發生可重試的例外狀況 (例如逾時或 OCC 衝突) 時重試交易。重試次數的上限上限的重試次數的上限的上限的上限的上限的上限

預設重試限制為4，且預設的輪詢策略ExponentialBackoffStrategy以10毫秒為基礎。您可以使用的執行個體來設定每個驅動程式執行個體以及每個交易的重試原則RetryPolicy。

下列程式碼範例會指定具有自訂重試限制的重試邏輯，以及驅動程式執行個體的自訂輪詢策略。

```
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)
  }
}
```

下列程式碼範例會指定具有自訂重試限制的重試邏輯，以及特定匿名函式的自訂輪詢策略。此SetRetryPolicy函數會覆寫為驅動程式執行個體設定的重試原則。

```
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")
  })
}
```

實作唯一性限制

QLDB 不支援唯一索引，但您可以在應用程式中實作此行為。

假設您要在`Person`表中的`GovId`字段上實現唯一性約束。要做到這一點，您可以寫入執行下列動作的交易：

1. 斷言該表沒有具有指定的現有文檔`GovId`。
2. 插入文檔，如果斷言通過。

如果競爭交易同時通過斷言，則只有其中一個交易將成功提交。另一個交易將會失敗，並出現 OCC 衝突例外狀況。

下列程式碼範例示範如何實作此唯一性限制條件限制條件限制

```
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**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

使用 Amazon Ion Ion Ion

下列幾節說明如何使用 Amazon Ion 模組來處理 Ion 資料的方式。

內容

- 建立 Ion 類型型
- Ion 子二進位檔
- Ion 文字文字檔

匯入離子模組

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

建立 Ion 類型型

Go 的 Ion 庫目前不支持文檔對象模型（DOM），因此您無法創建 Ion 數據類型。但是在使用 QLDB 時，您可以在 Go 本地類型和離子二進製文件之間進行編組和解組。

Ion 子二進位檔

```
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]
```

Ion 文字文字檔

```
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"}
```

如需 Ion 的詳細資訊，請參閱上的 Amazon Ion 文件 GitHub。如需在 QLDB 中使用 Ion 的更多程式碼範例，請參閱在亞馬遜 QLDB 中使用亞馬遜離子數據類型。

Node.js 的 Amazon QLDB 驅動程序

若要使用總帳中的資料，您可以使用AWS提供的驅動程式，從 Node.js 應用程式連線到 Amazon QLDB。下列主題說明如何開始使用適用於 Node.js 的 QLDB 驅動程式。

主題

- 司機資源
- 必要條件
- 安裝
- 設定建議
- Node.js 的亞馬遜 QLDB 驅動程序-快速入門教程
- Node.js 的亞馬遜 QLDB 驅動程序-食譜參考

司機資源

如需 Node.js 驅動程式支援之功能的詳細資訊，請參閱下列資源：

- 應用程式介面參考：3.x、2.x、1.x
- 驅動程式原始程式碼 (GitHub)
- 應用程式原始程式碼範例 (GitHub)
- Amazon Ion 程式碼範例
- 使AWS Lambda用（博客）在 QLDB 上構建一個簡單的 CRUD 操作和數據流 AWS

必要條件

在您開始使用 Node.js 的 QLDB 驅動程式之前，您必須執行下列動作：

1. 請遵循中的AWS設定指示訪問 Amazon QLDB。這包含下列項目：
    1. 註冊 AWS。

2. 建立具有適當 QLDB 權限的使用者。
3. 授予程序化訪問以進行開發。

2. 從 Node.js 下載網站安裝 Node.js 版本 14.x 或更新版本。(先前版本的驅動程式支援 Node.js 10.x 版或更新版本。)
3. JavaScript 在 Node.js 中為開發AWS套件設定您的開發環境：

   1. 設定您的AWS認證。建議您建立共用認證檔案。

      如需指示，請參閱AWS SDK for JavaScript開發人員指南中的從共用認證檔案載入 Node.js 中的認證。

   2. 設定您的預設值AWS 區域。若要瞭解如何操作，請參閱設定 AWS 區域.

      如需可用區域的完整清單，請參閱中的 Amazon QLDB 端點和配額。AWS 一般參考

接下來，您可以下載完整的教學課程範例應用程式，或者您可以只在 Node.js 專案中安裝驅動程式並執行簡短的程式碼範例。

- 若要在現有專案中安裝 Node.js JavaScript 中的 QLDB 驅動程式和 AWS SDK，請繼續執行。安裝
- 若要設定專案並執行展示分類帳基本資料交易的簡短程式碼範例，請參閱快速入門教學。
- 若要在完整的教學課程範例應用程式中執行資料和管理 API 作業的更深入範例，請參閱Node.js 教學課程.

安裝

QLDB 支援下列驅動程式版本及其 Node.js 相依性。

驅動程式版本	Node.js 版本	Status	最新發布日期
1.	10.x 或更新版本	生產發行	2020 年 6 月 5 日
2.x	10.x 或更新版本	生產發行	2021 年 5 月 6 日
3.x	14.x 或更新版本	生產發行	2023 年 11 月 10 日

要使用 npm（Node.js 軟件包管理器）安裝 QLDB 驅動程序，請從項目根目錄中輸入以下命令。

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
```

此驅動程式對下列套件具有對等相依性。您還必須在項目中將這些軟件包作為依賴項安裝。

3.x

模組化聚合式 QLDB 用戶端 (管理 API)

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

模組化彙總的 QLDB 工作階段用戶端 (資料 API)

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

Amazon 離子數據格式

```
npm install ion-js
```

純粹的 JavaScript 實施 `BigInt`

```
npm install jsbi
```

2.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Amazon 離子數據格式

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

純粹的 JavaScript 實施 BigInt

```
npm install jsbi@3.1.1
```

1.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Amazon 離子數據格式

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

純粹的 JavaScript 實施 BigInt

```
npm install jsbi@3.1.1
```

使用驅動程序連接到分類帳

然後，您可以導入驅動程序並使用它來連接到分類帳。下列程式 TypeScript 碼範例顯示如何針對指定的總帳名稱和建立驅動程式執行個體AWS 區域。

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);
});
```

如需如何在分類帳上執行基本資料交易的簡短程式碼範例，請參閱食譜參考。

設定建議

以保持活動狀態重複使用連線

QLDB Node.js 驅動程式 v3

預設 Node.js HTTP/HTTPS 代理程式會為每個新的請求建立新的 TCP 連線。為了避免建立新連線的成本，AWS SDK for JavaScriptv3 預設會重複使用 TCP 連線。如需詳細資訊並瞭解如何停用重複使用連線，請參閱開發人員指南中的 Node.js 中以保持活動狀態重複使用連線。AWS SDK for JavaScript

我們建議您使用預設設定，重複使用 Node.js 的 QLDB 驅動程式中的連線。在驅動程式初始化期間，請`maxSockets`將低階用戶端 HTTP 選項設定為`maxConcurrentTransactions`與您設定的相同值。

例如，請參閱下列 JavaScript 或 TypeScript 程式碼。

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 驅動程式 V2

預設 Node.js HTTP/HTTPS 代理程式會為每個新的請求建立新的 TCP 連線。若要避免建立新連線的成本，建議您重複使用現有連線。

若要重複使用 Node.js 的 QLDB 驅動程式中的連線，請使用下列其中一個選項：

- 在驅動程式初始化期間，請設定下列低階用戶端 HTTP 選項：
  - `keepAlive` – `true`
  - `maxSockets`— 與您設定的相同值 `maxConcurrentTransactions`

  例如，請參閱下列 JavaScript 或 TypeScript 程式碼。

  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);
```

- 或者，您可以將AWS_NODEJS_CONNECTION_REUSE_ENABLED環境變數設定為1。如需詳細資訊，請參閱開發人員指AWS SDK for JavaScript南中的 Node.js 中以保持活動狀態重複使用連線。

> **Note**
>
> 如果您設定此環境變數，它會影響所有使用AWS SDK for JavaScript. AWS 服務

Node.js 的亞馬遜 QLDB 驅動程序-快速入門教程

在此教學中，您將學習如何使用 Node.js 的 Amazon QLDB 驅動程式來設定簡單的應用程式。本指南包含安裝驅動程式的步驟，以 JavaScript 及 TypeScript 基本的建立、讀取、更新和刪除 (CRUD) 操作。如需在完整範例應用程式中示範這些作業的更深入範例，請參閱Node.js 教學課程.

> **Note**
>
> 在適用的情況下，某些步驟會針對 Node.js 的 QLDB 驅動程式的每個支援主要版本提供不同的程式碼範例。

主題

先決條件

在開始使用之前，請確認執行下列作業：

1. 如果您尚未完成，請完成 Node.js 驅動程式。必要條件這包括註冊AWS、授予程式設計存取以供開發，以及安裝 Node.js。
2. 建立名為的分類帳`quick-start`。

若要瞭解如何建立分類帳，請參閱Amazon QLDB 分類帳的基本操作或步驟 1：建立新的分類帳在「開始使用主控台」中的。

如果您正在使用 TypeScript，則還必須執行以下設置步驟。

使用 TypeScript

若要安裝 TypeScript

1. 安裝 TypeScript 套裝。QLDB 驅動程式在 TypeScript 3.8.x 上執行。

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

2. 安裝套裝，請執行下列命令，確認安裝了 TypeScript 編譯器。

```
$ tsc --version
```

若要在下列步驟中執行程式碼，請注意，您必須先將 TypeScript 檔案轉譯為可執行 JavaScript 程式碼，如下所示。

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

步驟 1：設定您的 專案

首先，設定您的 Node.js 專案。

1. 建立您的應用程式資料夾。

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

2. 若要初始化您的專案，請輸入下列npm指令並回答設定期間所詢問的問題。您可以對大多數問題使用預設值。

```
$ npm init
```

3. 安裝適用於 Node.js 的亞馬遜 QLDB 驅動程式。
   - 使用第 3.x 版

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

- 使用版本 2.x

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

- 使用版本 1.x

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

4. 安裝驅動程序的對等依賴關係。
   - 使用第 3.x 版

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

   - 使用版本 2.x 或 1.x 版

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

5. 建立一個名`app.js`為 JavaScript、或的新檔`app.ts`案 TypeScript。

   然後，在以下步驟中逐步添加代碼示例，以嘗試一些基本的 CRUD 操作。或者，您可以略過 step-by-step 教學課程，而是執行完整的應用程式。

步驟 2：初始化驅動程式

初始化連線至名為之分類帳之驅動程式的執行環境`quick-start`。將以下代碼添加到您的`app.js`或`app.ts`文件中。

使用第 3.x 版

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

- 在此程式碼範例中，將 *us-east-1* 取代為AWS 區域您建立分類帳的位置。
- 為了簡化起見，本指南中其餘的程式碼範例會使用具有預設設定的驅動程式，如下列 1.x 版範例中所指定的驅動程式。您也可以RetryConfig改用您自己的驅動程式實例。

使用版本 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**
>
> - 在此程式碼範例中，將 *us-east-1* 取代為AWS 區域您建立分類帳的位置。
> - 版本 2.x 引入了用`RetryConfig`於初始化的新的可選參數`QldbDriver`。
> - 為了簡化起見，本指南中其餘的程式碼範例會使用具有預設設定的驅動程式，如下列 1.x 版範例中所指定的驅動程式。您也可以`RetryConfig`改用您自己的驅動程式實例。
> - 此程式碼範例會透過設定 keep-alive 選項，初始化重複使用現有連線的驅動程式。如需進一步了解，請設定建議參閱 Node.js 驅動程式。

使用版本 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**
>
> 您可以設定AWS_REGION環境變數來指定「區域」。如需詳細資訊，請參閱AWS SDK for JavaScript開發人員指南AWS 區域中的設定。

步驟 3：建立資料表和索引

下列程CREATE INDEX式碼示範例示範例示範例示範例示範例示範例CREATE TABLE

1. 添加以下函數，創建一個名為的表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. 新增下列函數，為People資料表上的firstName欄位建立索引。需要索引來最佳化查詢效能，並協助限制最佳化並行控制 (OCC) 衝突例外狀況。

   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. 在main函數中，你先調用createTable，然後調用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. 運行代碼來創建表和索引。

JavaScript

```
$ node app.js
```

TypeScript

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

步驟 4：插入文件

下列程式碼範例示範例示範例示範例示範例。INSERTQLDB 支援 PartiQL 查詢語言 (SQL 相容) 和亞馬遜離子資料格式 (JSON 的超集)。

1. 添加以下將文檔插入People表中的函數。

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);
}
```

此範例使用問號 (?) 做為變數預留位置，將文件資訊傳遞至陳述式。此方`execute`法支援亞馬遜離子類型和 Node.js 原生類型中的值。

> **Tip**
>
> 要通過使用單個INSERT語句插入多個文檔，你可以通過類型列表的參數到語句如下。
>
> ```
> // people is a list
> txn.execute("INSERT INTO People ?", people);
> ```
>
> 傳遞清單時，您不會將變數預留位置 (?<<...>>) 括在雙尖括號 () 中。在手動 PartiQL 陳述式中，雙角括號表示稱為袋子的無序集合。

2. 在`main`函數中，移除`createTable`和`createIndex`呼叫，然後將呼叫新增至`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();
}
```

步驟 5：查詢文件

下列程式碼範例示範例示範例示範例示範例示範例。SELECT

1. 添加以下功能，從People表中查詢文檔。

   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. 在main函數中，在呼叫fetchDocuments後將下列呼叫新增至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();
```

```
}
```

步驟 6：更新文件

下列程式碼範例示範例示範例示範例示範例。UPDATE

1. 新增下列函數，透過將變更為來更新People表格中的文lastName件"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. 在main函數中，在呼叫updateDocuments後將下列呼叫新增至fetchDocuments。然後，fetchDocuments再次打電話以查看更新的結果。

   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. 執行程式碼以插入、查詢和更新文件。

JavaScript

```
$ node app.js
```

TypeScript

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

執行完整的應用程式

下列程式碼範例是app.js和的完整版本app.ts。您也可以從頭到尾運行此代碼，而不是單獨執行前面的步驟。此應用程序演示了名為的分類帳上的一些基本 CRUD 操作quick-start。

> **Note**
>
> 在執行此程式碼之前，請確定您尚未在quick-start分類帳People中命名為使用中的資料表。

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();
}
```

如需執行完整的應用程式，請輸入下列命令。

JavaScript

```
$ node app.js
```

TypeScript

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

Node.js 的亞馬遜 QLDB 驅動程序-食譜參考

本參考指南顯示適用於 Node.js 的亞馬遜 QLDB 驅動程式的常見使用案例。它提供 JavaScript 的程式 TypeScript 碼範例會示範如何使用驅動程式來執行基本的建立、讀取、更新及刪除 (CRUD) 操作。它還包括用於處理 Amazon Ion 資料的程式碼範例。此外，本指南還重點介紹了使交易冪等和實施唯一性約束的最佳實踐。

內容

匯入驅動程式

下列程式碼範例會匯入驅動程式。

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**
>
> 此範例也會匯入 Amazon Ion 套件 (`ion-js`)。在此參考中執行某些資料作業時，您需要此套件來處理 Ion 資料。如需進一步了解，請參閱 使用 Amazon Ion。

實例化驅動程式

下列程式碼範例會建立使用預設設定連線至指定分類帳名稱的驅動程式執行環境。

JavaScript

```
const qldbDriver = new qldb.QldbDriver("vehicle-registration");
```

TypeScript

```
const qldbDriver: QldbDriver = new QldbDriver("vehicle-registration");
```

CRUD 操作

QLDB 運行建立、讀取、更新及刪除 (CRUD) 操作。

> **Warning**
>
> 根據最佳實務，寫入交易必須使用嚴格的冪等。

使交易冪等

我們建議您將寫入事務設為冪等，以避免在重試的情況下出現任何意外的副作用。如果事務可以運行多次並每次產生相同的結果，則事務是冪等的。

例如，假設將文件插入名為的資料表中的交易Person。事務應該首先檢查該文檔是否已經存在於表中。如果沒有此檢查，表格最終可能會出現重複的文件。

假設 QLDB 成功地在服務器端提交事務，但在等待響應的客戶端超時。如果事務不是冪等的，則在重試的情況下，可以多次插入相同的文檔。

使用索引來避免完整表格掃描

我們也建議您在索引欄位或文件 ID 上使用相等運算子來執行具有述WHERE詞子句的陳述式；例如，`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`。如果沒有此索引查找，QLDB 需要進行表掃描，這可能會導致交易逾時或樂觀並發控制（OCC）衝突。

如需的詳細資訊，請參閱Amazon QLLDB 並行模型。

隱含建立的交易

QldbDriver.executeLambda 方法接受接收的TransactionExecutor執行個體 (可用來執行陳述式) 的 Lambda 函數。TransactionExecutor封裝隱含建立之交易的執行個體。

您可以使用交易執行程式的執行方法，在 lambda 函數中執行陳述式。當 lambda 函數返回時，驅動程序隱式地提交交易。

> **Note**
>
> 此方execute法同時支援亞馬遜離子類型和 Node.js 原生類型。如果您將 Node.js 原生類型作為引數傳遞給execute，驅動程式會使用ion-js套件將其轉換為 Ion 類型 (前提是支援指定 Node.js 資料類型的轉換)。有關支持的數據類型和轉換規則，請參閱 Ion JavaScript DOM 自述文件。

下列各節說明如何執行基本 CRUD 作業、指定自訂重試邏輯，以及實作唯一性限制。

內容

建立資料表

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

建立索引

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

閱讀文件

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']
        }
    });
}());
```

使用查詢參數

下列程式碼範例會使用原生型別查詢參數。

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']
        }
    });
}());
```

下列程式碼範例會使用 Ion 型別查詢參數。

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']
        }
    });
}());
```

下列程式碼範例會使用多個查詢參數。

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']
        }
    });
}());
```

下列程式碼範例使用查詢參數清單。

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**
>
> 當您在沒有索引查閱的情況下執行查詢時，會叫用完整資料表掃描。在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，查詢可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

插入文件

下列程式碼範例會插入原生資料類型。

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

下列程式碼範例會插入 Ion 資料型別。

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

此事務將一個文檔插入到Person表中。在插入之前，它會先檢查文件是否已存在於表格中。這個檢查使事務本質上是冪等的。即使您多次運行此事務，也不會造成任何意外的副作用。

> **Note**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

在一個語句中插入多個文檔

要通過使用單個INSERT語句插入多個文檔，你可以通過類型列表的參數到語句如下。

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

傳遞清單時，您不會將變數預留位置 (?<<...>>) 括在雙尖括號 () 中。在手動 PartiQL 陳述式中，雙角括號表示稱為袋子的無序集合。

更新文件

下列程式碼範例會使用原生資料類型。

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

下列程式碼範例會使用 Ion 資料型別。

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**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

刪除文件

下列程式碼範例會使用原生資料類型。

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

下列程式碼範例會使用 Ion 資料型別。

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**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

在交易中執行多個陳述式

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;
    });
};
```

重試邏輯

驅動程式的`executeLambda`方法具有內建的重試機制，可在發生可重試的例外狀況 (例如逾時或 OCC 衝突) 時重試交易。重試嘗試嘗試次數的上限和輪詢策略是可設定的。

預設重試限制為`4`，且預設的輪詢策略defaultBackoffFunction以`10`毫秒為基礎。您可以使用的執行個體來設定每個驅動程式執行個體以及每個交易的重試組態RetryConfig。

下列程式碼範例會指定具有自訂重試限制的重試邏輯，以及驅動程式執行個體的自訂輪詢策略。

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);
```

下列程式碼範例會指定具有自訂重試限制的重試邏輯，以及針對特定 lambda 執行的自訂輪詢策略。此組態`executeLambda`會覆寫為驅動程式執行個體設定的重試邏輯。

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);
}());
```

實施唯一性限制

QLDB 不支援唯一索引，但您可以在應用程式中實作此行為。

假設您要在Person表中的GovId字段上實現唯一性約束。要執行此操作，您可以編寫執行下列操作的交易：

1. 斷言該表沒有具有指定的現有文檔GovId。
2. 插入文檔，如果斷言通過。

如果競爭交易同時通過斷言，則只有其中一個交易將成功提交。另一個交易將會失敗，並出現 OCC 衝突例外狀況。

下列程式碼範例示範例示範如何實現此唯一性限制邏輯。

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**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

使用 Amazon Ion

下列幾節說明如何使用 Amazon Ion 模組來處理 Ion 資料。

內容

匯入離子模組

JavaScript

```
var ionjs = require('ion-js');
```

TypeScript

```
import { dom, dumpBinary, dumpText, load } from "ion-js";
```

建立 Ion 類型

下列程式碼範例會從 Ion 文字建立 Ion 物件。

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']
```

下列程式碼範例會使用 Node.js 字典建立 Ion 物件。

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']
```

獲取離子二進制轉儲

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
```

獲取離子文本轉儲

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"}
```

如需 Ion 的詳細資訊，請參閱上的 Amazon Ion 文件 GitHub。如需在 QLDB 中使用 Ion 的更多程式碼範例，請參閱在亞馬遜 QLDB 中使用亞馬遜離子數據類型。

亞馬遜 QLDB 驅動程序的 Python

若要使用分類帳中的資料，您可以使用AWS提供的驅動程式，從 Python 應用程式連線到 Amazon QLDB。下列主題說明如何開始使用適用於 Python 的 QLDB 驅動程式。

主題

- 駕駛員資源
- 先決條件
- 安裝
- 適用於 Python 的亞馬遜 QLDB 驅動程序 — 快速入門教程
- 亞馬遜 QLDB 驅動程序的 Python-食譜參考

駕駛員資源

如需 Python 驅動程式所支援功能的詳細資訊，請參閱下列資源：

- API 參考資料：3.x、2.x
- 驅動程式原始程式碼 (GitHub)
- 應用程式原始程式碼範例 (GitHub)
- Amazon Ion 程式碼範例

先決條件

您必須執行以下作：

1. 請遵循中的AWS設定指示訪問 Amazon QLDB。這包含下列項目：
    1. 註冊 AWS。
    2. 建立具有適當 QLDB 許可的使用者。
    3. 授予程式設計存取權，以供開發。
2. 從 Python 下載網站安裝下列其中一個版本：
    - 3.6 或更高版本-適用於蟒蛇 V3 的 QLDB 驅動程序
    - 3.4 或更新版本 — 適用於 Python V2 的 QLDB 驅動程式
3. 設定您的AWS憑證和預設值AWS 區域。如需指示，請參閱AWS SDK for Python (Boto3)文件中的快速入門。

    如需可用區域的完整清單，請參閱中的 Amazon QLDB 端點和配額AWS 一般參考。

接下來，您可以下載完整的教學課程範例應用程式，或者您可以只在 Python 專案中安裝驅動程式並執行簡短的程式碼範例。

- 若要AWS SDK for Python (Boto3)在現有專案中安裝 QLDB 驅動程式和，請繼續執行安裝。
- 若要設定專案並執行展示分類帳基本資料交易的簡短程式碼範例，請參閱速入門教學課程。
- 若要在完整的教學課程範例應用程式中執行資料和管理 API 作業的更深入範例，請參閱Python 教學課程.

安裝

QLDB 支援下列驅動程式版本及其 Python 相依性。

驅動程式版本	Python 版本	狀態	最新發佈日期
2.x	3.4 或更高版本	生產發行	2020 年 5 月 7 日
3.x	3.6 或更高版本	生產發行	2021 年 10 月 28 日

要使用`pip`（Python 的軟件包管理器）從 PyPI 安裝 QLDB 驅動程序，請在命令行中輸入以下內容。

3.x

```
pip install pyqldb
```

2.x

```
pip install pyqldb==2.0.2
```

安裝驅動程式也會安裝其相依性，包括AWS SDK for Python (Boto3)和 Amazon Ion 套件。

使用驅動程序連接到分類帳

然後，您可以導入驅動程序並使用它來連接到分類帳。下列 Python 程式碼範例示範如何建立指定分類帳名稱的工作階段。

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)
```

如需如何在分類帳上執行基本資料交易的簡短程式碼範例，請參閱食譜參考。

適用於 Python 的亞馬遜 QLDB 驅動程序 — 快速入門教程

在此教學中，您將學習如何使用適用於 Python 的最新版 Amazon QLDB 驅動程式，設定簡單的應用程式。本指南包含安裝驅動程式的步驟和基本的建立、讀取、更新和刪除 (CRUD) 操作的簡短程式碼範例。如需在完整範例應用程式中示範這些作業的更深入範例，請參閱Python 教學課程.

主題

先決條件

在開始使用之前，請確認執行下列作業：

1. 完成先決條件 Python 驅動程式，如果您尚未這麼做。這包括註冊AWS、授予程式設計存取以供開發，以及安裝 Python 3.6 版或更新版本。
2. 建立名為的分類帳`quick-start`。

   若要瞭解如何建立分類帳，請參閱Amazon QLDB 分類帳的基本操作或步驟 1：建立新的分類帳在「開始使用主控台」中的。

步驟 1：設定您的 專案

首先，設定您的 Python 專案。

> **Note**
>
> 如果您使用的 IDE 具有自動執行這些設定步驟的功能，您可以跳到步驟 2：初始化驅動程式。

1. 建立您的應用程式資料夾。

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

2. 若要從 PyPI 安裝適用於 Python 的 QLDB 驅動程式，請輸入以下`pip`指令。

```
$ pip install pyqldb
```

安裝驅動程式也會安裝其相依性，包括AWS SDK for Python (Boto3)和 Amazon Ion 套件。

3. 建立名為 app.py 的新檔案。

然後，在以下步驟中逐步添加代碼示例，以嘗試一些基本的 CRUD 操作。或者，您可以略過 step-by-step 教學課程，而是執行完整的應用程式。

步驟 2：初始化驅動程式

初始化連線至名為之分類帳之驅動程式的執行環境quick-start。將下列程式碼新增至您的app.py檔案：

```
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)
```

步驟 3：建立資料表和索引

下列程式碼範例示範例示範如何執行CREATE TABLE和CREATE INDEX陳述式。

添加以下代碼，該代碼創建一個名為的表People和該表上的lastName字段的索引。需要索引來最佳化查詢效能，並協助限制最佳化並行控制 (OCC) 衝突例外狀況。

```
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))
```

步驟 4：插入文件

下列程式碼範例示範如何執行INSERT陳述式。QLDB 支援 PartiQL 查詢語言 (與 SQL 相容) 和亞馬遜離子資料格式 (JSON 的超集)。

添加下面的代碼，插入一個文檔到People表中。

```
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))
```

此範例使用問號 (?) 做為變數預留位置，將文件資訊傳遞至陳述式。該execute_statement方法支持亞馬遜離子類型和 Python 本機類型中的值。

> **Tip**
>
> 要通過使用單個INSERT語句插入多個文檔，你可以通過類型列表的參數到語句如下。
>
> ```
> # people is a list
> transaction_executor.execute_statement("INSERT INTO Person ?", people)
> ```
>
> 傳遞清單時，您不會將變數預留位置 (?<<...>>) 括在雙尖括號 () 中。在手動 PartiQL 陳述式中，雙角括號表示稱為袋子的無序集合。

步驟 5：查詢文件

下列程式碼範例示範如何執行SELECT陳述式。

添加以下代碼查詢從People表中的文檔。

```
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))
```

步驟 6：更新文件

下列程式碼範例示範例示範如何執行UPDATE陳述式。

1. 添加以下代碼，通過更新age到更新People表中的文檔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. 再次查詢資料表以查看更新的值。

   ```
   # Query the updated document
   qldb_driver.execute_lambda(lambda executor: read_documents(executor))
   ```

3. 若要執行應用程式，請從您的專案目錄輸入下列命令。

   ```
   $ python app.py
   ```

執行完整的應用程式

下列程式碼範例是應app.py用程式的完整版本。您也可以從頭到尾複製並執行此程式碼範例，而不是個別執行前述步驟。此應用程序演示了名為的分類帳上的一些基本 CRUD 操作quick-start。

> **Note**
>
> 在執行此程式碼之前，請確定您尚未在quick-start分類帳People中命名為使用中的資料表。

```
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))
```

若要執行完整的應用程式，請從您的專案目錄輸入下列命令。

```
$ python app.py
```

亞馬遜 QLDB 驅動程序的 Python-食譜參考

本參考指南顯示適用於 Python 的亞馬遜 QLDB 驅動程式的常見使用案例。該範例示範如何使用驅動程式碼範例會示範如何使用該驅動程式碼範例會示範如何使用驅動程式碼範例會示範如何使用驅動程式碼

範例 它還包括用於處理 Amazon Ion 資料的程式碼範例。此外，本指南還重點介紹了使交易冪等和實施唯一性約束的最佳實踐。

> **Note**
>
> 在適用的情況下，某些使用案例會針對每個支援的 Python QLDB 驅動程式主要版本提供不同的程式碼範例。

內容

匯入驅動程式

下列程式碼範例會匯入驅動程式。

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**
>
> 此範例也會匯入 Amazon Ion 套件 (`amazon.ion.simpleion`)。在此參考中執行某些資料作業時，您需要此套件來處理 Ion 資料。如需進一步了解，請參閱 使用 Amazon Ion。

實例化驅動程式

下列程式碼範例會建立使用預設設定連線至指定分類帳名稱的驅動程式執行環境。

3.x

```
qldb_driver = QldbDriver(ledger_name='vehicle-registration')
```

2.x

```
qldb_driver = PooledQldbDriver(ledger_name='vehicle-registration')
```

CRUD 操作

QLDB 運行建立、讀取、更新和刪除 (CRUD) 操作。

> **Warning**
>
> 根據最佳實務，寫入交易必須使寫入交易嚴格的冪等。

使交易冪等

我們建議您將寫入事務設為冪等，以避免在重試的情況下出現任何意外的副作用。如果事務可以運行多次並每次產生相同的結果，則事務是冪等的。

例如，假設將文件插入名為的資料表中的交易Person。事務應該首先檢查該文檔是否已經存在於表中。如果沒有此檢查，表格最終可能會出現重複的文件。

假設 QLDB 成功地在服務器端提交事務，但在等待響應的客戶端超時。如果事務不是冪等的，則在重試的情況下，可以多次插入相同的文檔。

使用索引來避免完整表格掃描

我們也建議您在索引欄位或文件 ID 上使用相等運算子來執行具有述WHERE詞子句的陳述式；例如，WHERE indexedField = 123或WHERE indexedField IN (456, 789)。如果沒有此索引查找，QLDB 需要進行表掃描，這可能會導致交易逾時或樂觀並發控制（OCC）衝突。

如需 OCC 的詳細資訊，請參閱「Amazon QLLDB 並行模型」。

隱含建立的交易

執行程序 .execute_lambda 方法接受一個 lambda 函數，該函數接受一個 lambda 函數，該函數接收執行程序。執行程序，您可以使用它來運行語句的實例。Executor封裝隱含建立之交易的執行個體。

您可以使用交易執行程式的 execute_陳述式方法，在 Lambda 函數中執行陳述式。當 lambda 函數返回時，驅動程序隱式地提交交易。

> **Note**
>
> 該execute_statement方法同時支持亞馬遜離子類型和 Python 本機類型。如果您將 Python 原生類型作為引數傳遞給execute_statement，則驅動程序會使用amazon.ion.simpleion模塊將其轉換為 Ion 類型（前提是支持給定 Python 數據類型的轉換）。有關支持的數據類型和轉換規則，請參閱簡單源代碼。

下列各節說明如何執行基本 CRUD 作業、指定自訂重試邏輯，以及實作唯一性限制。

內容

- 建立資料表
- 建立索引
- 閱讀文件

建立資料表

```
def create_table(transaction_executor):
    transaction_executor.execute_statement("CREATE TABLE Person")

qldb_driver.execute_lambda(lambda executor: create_table(executor))
```

建立索引

```
def create_index(transaction_executor):
    transaction_executor.execute_statement("CREATE INDEX ON Person(GovId)")

qldb_driver.execute_lambda(lambda executor: create_index(executor))
```

閱讀文件

```
# 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))
```

使用查詢參數

下列程式碼範例會使用原生型別查詢參數。

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId = ?",
 'TOYENC486FH')
```

下列程式碼範例會使用 Ion 型別查詢參數。

```
name = ion.loads('Brent')
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE FirstName
 = ?", name)
```

下列程式碼範例會使用多個查詢參數。

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId = ?
 AND FirstName = ?", 'TOYENC486FH', "Brent")
```

下列程式碼範例使用查詢參數清單。

```
gov_ids = ['TOYENC486FH','ROEE1','YH844']
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId IN
 (?,?,?)", *gov_ids)
```

> **Note**
>
> 當您在沒有索引查閱的情況下執行查詢時，會叫用完整資料表掃描。在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，查詢可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

插入文件

下列程式碼範例會插入原生資料類型。

```
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))
```

下列程式碼範例會插入 Ion 資料型別。

```
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))
```

此事務將一個文檔插入到Person表中。在插入之前，它會先檢查文件是否已存在於表格中。這個檢查使事務本質上是冪等的。即使您多次運行此事務，也不會造成任何意外的副作用。

> **Note**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

在一個語句中插入多個文檔

要通過使用單個INSERT語句插入多個文檔，你可以通過類型列表的參數到語句如下。

```
# people is a list
transaction_executor.execute_statement("INSERT INTO Person ?", people)
```

傳遞清單時，您不會將變數預留位置 (?<<...>>) 括在雙尖括號 () 中。在手動 PartiQL 陳述式中，雙角括號表示稱為袋子的無序集合。

更新文件

下列程式碼範例會使用原生資料類型。

```
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))
```

下列程式碼範例會使用 Ion 資料型別。

```
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**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

刪除文件

下列程式碼範例會使用原生資料類型。

```
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))
```

下列程式碼範例會使用 Ion 資料型別。

```
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**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

在交易中執行多個陳述式

```
# 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))
```

重試邏輯

驅動程式的`execute_lambda`方法具有內建的重試機制，可在發生可重試的例外狀況 (例如逾時或 OCC 衝突) 時重試交易。

3.x

重試試試試次數上限的重數上限的重試次數上限的重試次數上限，

預設重試限制為`4`，預設輪詢策略為指數輪詢和抖動，以`10`毫秒為基礎。您可以通過使用 pyqldb.config.retry_config 的實例來設置每個驅動程序實例和每個事務的重試配置。 RetryConfig。

下列程式碼範例會指定具有自訂重試限制的重試邏輯，以及驅動程式執行個體的自訂輪詢策略。

```
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)
```

下列程式碼範例會指定具有自訂重試限制的重試邏輯，以及針對特定 lambda 執行的自訂輪詢策略。此組態execute_lambda會覆寫為驅動程式執行個體設定的重試邏輯。

```
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

重試試試試次數上限的重試試次數上限的重 您可以在初始化時設定retry_limit屬性來設定重試限制PooledQldbDriver。

預設重試限制為4。

實行唯一性限制

QLDB 不支援唯一索引，但您可以在應用程式中實作此行為。

假設您要在Person表中的GovId字段上實現唯一性約束。您可通過下列方式執行此操作：

1. 斷言該表沒有具有指定的現有文檔GovId。
2. 插入文檔，如果斷言通過。

如果競爭交易同時通過斷言，則只有其中一個交易將成功提交。另一個交易將會失敗，並出現 OCC 衝突例外狀況。

下列程式碼範例示範如何實作此唯一性限制邏輯。

```
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**
>
> 在此範例中，我們建議在GovId欄位上建立索引以最佳化效能。如果沒有開啟索引GovId，陳述式可能會有更多延遲，也可能導致 OCC 衝突例外狀況或交易逾時。

使用 Amazon Ion

下列幾節示範如何使用 Amazon Ion 模組來處理 Ion 資料。

內容

匯入離子模組

```
import amazon.ion.simpleion as simpleion
```

建立 Ion 類型

下列程式碼範例會從 Ion 文字建立 Ion 物件。

```
ion_text = '{GovId: "TOYENC486FH", FirstName: "Brent"}'
ion_obj = simpleion.loads(ion_text)

print(ion_obj['GovId']) # prints TOYENC486FH
print(ion_obj['Name']) # prints Brent
```

下列程式碼範例會從 Python 建立離子物件`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
```

獲取離子二進制轉儲

```
# 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'
```

獲取離子文本轉儲

```
# ion_obj is an Ion struct
print(simpleion.dumps(ion_obj, binary=False)) # prints $ion_1_0
 {GovId:'TOYENC486FH',FirstName:"Brent"}
```

如需有關使用 Ion 的詳細資訊，請參閱上的 Amazon Ion 文件 GitHub。如需在 QLDB 中使用 Ion 的更多程式碼範例，請參閱在亞馬遜 QLDB 中使用亞馬遜離子數據類型。

了解 Amazon QLDB 中的驅動程式的工作階段管理

如果您有使用關聯式資料庫管理系統 (RDBMS) 的經驗，您可能熟悉並行連線。QLDB 不具有傳統的 RDBMS 連接的相同概念，因為事務與 HTTP 請求和響應消息運行。

在 QLDB 中，類似的概念是一個活動的會話。工作階段在概念上與使用者登入類似，它會管理您對總帳之資料交易請求的相關資訊。使用中的工作階段是主動執行交易的工作階段。它也可以是最近完成交易的工作階段，服務預期會立即啟動另一個交易。QLDB 支援每個工作階段一個主動執行的交易。

在中定義每個分類帳的並行作用中階段作業限制亞馬遜 QLDB 中的配額和限制。達到此限制之後，任何嘗試啟動交易的工作階段都會導致 error (`LimitExceededException`)。

如需使用 QLDB 驅動程式在應用程式中設定工作階段集區的最佳實務，請參閱 Amazon QLDB 驅動程式建議設定 QldbDriver 對象中的。

內容

工作階段

QLDB 工作階段 API 作業的下列順序代表 QLDB 工作階段的典型生命週期：

1. `StartSession`
2. `StartTransaction`
3. `ExecuteStatement`
4. `CommitTransaction`
5. 重複步驟 2—4 以啟動更多交易 (一次一個交易)。
6. `EndSession`

工作階段過期

無論是否主動執行交易，QLDB 都會在總存留期間 13 到 17 分鐘後過期並捨棄工作階段。工作階段可能會因為多種原因而遺失或受損，例如硬體故障、網路故障或應用程式重新啟動。QLDB 會強制執行工作階段的最大存留期，以確保用戶端應用程式能夠回復工作階段失敗。

QLDB 驅動程式中的工作階段處理

雖然您可以使用AWS SDK 直接與 QLDB 工作階段 API 互動，但這會增加複雜性，並要求您計算提交摘要。QLDB 使用此提交摘要來確保交易的完整性。我們建議您使用 QLDB 驅動程式，而不是直接與此 API 互動。

驅動程式提供交易資料 API 上方的高階抽象層。它透過管理 SendCommandAPI 呼叫，簡化在分類帳資料上執行 PartiQL 陳述式的程序。這些 API 調用需要驅動程序為您處理的幾個參數，包括會話，事務的管理，並在發生錯誤時重試策略。

主題

階段集區概觀

在舊版的 QLDB 驅動程式 (例如 Java v1.1.0) 中，我們提供了驅動程式物件的兩種實作：標準、非集區`QldbDriver`和`PooledQldbDriver`. 顧名思義，`PooledQldbDriver`維護跨交易重複使用的工作階段集區。

根據用戶反饋，開發人員更喜歡默認情況下獲得池功能及其優點，而不是使用標準驅動程序。因此，我們移除`PooledQldbDriver`並將工作階段集區功能移至`QldbDriver`。此變更包含在每個驅動程式的最新版本中 (例如，Java v2.0.0)。

驅動程式提供三個層級的抽象：

- 驅動程序（池驅動程序實現）-頂級抽象。驅動程式會維護和管理工作階段集區。當您要求驅動程序運行事務時，驅動程序從池中選擇一個會話並使用它來運行事務。如果交易因為工作階段錯誤 (`InvalidSessionException`) 而失敗，則驅動程式會使用另一個工作階段重試該交易。基本上，驅動程式提供完全受控的工作階段體驗。
- 工作階段 — 低於驅動程式抽象的一個層級。會話包含在池中，驅動程序管理會話的生命週期。如果交易失敗，驅動程式會在使用相同工作階段嘗試指定次數的情況下重試。但是，如果會話遇到 error (`InvalidSessionException`)，QLDB 會在內部丟棄它。然後，驅動程序負責從池中獲取另一個會話以重試該事務。
- 交易 — 抽象的最低層級。事務包含在會話中，並且會話管理事務的生命週期。工作階段負責在發生錯誤時重試交易。工作階段也可確保它不會洩漏尚未提交或取消的開啟交易。

在每個驅動程序的最新版本中，您只能在抽象的驅動程序級別執行操作。您無法直接控制個別工作階段和交易 (也就是說，沒有 API 作業可以手動啟動新的工作階段或交易)。

階段作業集區和交易邏輯

每個驅動程式的最新版本不再提供驅動程式物件的非集區實作。依預設，`QldbDriver`物件會管理工作階段集區。當您撥打電話來執行交易時，驅動程式會執行下列步驟：

1. 驅動程式會檢查是否已達到工作階段集區限制。如果是這樣，驅動程序立即拋出`NoSessionAvailable`異常。否則，會繼續至下一個步驟。
2. 驅動程式會檢查集區是否有可用的工作階段。
   - 如果池中有可用的會話，則驅動程序將使用它來運行事務。
   - 如果池中沒有可用的會話，驅動程序將創建一個新的會話並使用它來運行事務。
3. 驅動程序在步驟 2 中獲得會話後，驅動程序調用會話實例上的`execute`操作。
4. 在會話的`execute`操作中，驅動程序嘗試通過調用啟動事務`startTransaction`。
   - 如果工作階段無效，`startTransaction`呼叫會失敗，且驅動程式會返回到步驟 1。
   - 如果通`startTransaction`話成功，驅動程式會繼續至下一個步驟。
5. 驅動程序運行您的 lambda 表達式。這個 lambda 表達式可以包含一個或多個運行 PartiQL 語句的調用。lambda 表達式完成運行後沒有任何錯誤，驅動程序繼續提交交易。
6. 交易的提交可以有以下兩種結果之一：
   - 提交成功，驅動程序將控制權返回到您的應用程序代碼。
   - 由於樂觀的並發控制（OCC）衝突，提交失敗。在此情況下，驅動程式會使用相同的工作階段重試步驟 4-6。重試嘗試次數的上限，預設限制為4。

> **Note**
>
> 如果`InvalidSessionException`在步驟 4 到 6 期間擲回，驅動程式會將工作階段標示為已關閉，並返回步驟 1 以重試交易。
> 如果在步驟 4-6 期間擲回任何其他例外狀況，驅動程式會檢查是否可以重試例外狀況。如果是這樣，它會重試交易，直到指定的重試嘗試次數。如果沒有，它會將異常傳播（冒泡並拋出）到您的應用程序代碼中。

將工作階段傳回集區

如果InvalidSessionException在使用中交易過程中的任何時間發生，驅動程式不會將工作階段傳回集區。QLDB 會捨棄工作階段，而驅動程式會從集區取得另一個工作階段。在其他情況下，驅動程式會將工作階段傳回至儲存區。

Amazon QLDB 驅動程式推薦

本節介紹為任何支持的語言配置和使用 Amazon QLDB 驅動程序的最佳實踐。提供的代碼示例專門用於 Java。

這些建議適用於大多數典型的使用案例，但一種尺寸並不適合所有情況。根據您認為適合您的應用程序，請使用以下建議。

主題

設定 QldbDriver 對象

所以此QldbDriver對象通過維護會議，它們可以跨事務重複使用。一個會議表示與分類帳的單個連接。QLDB 支持每個會話一個主動運行的事務。

> **Important**
>
> 對於較舊版本的驅動程序，會話池功能仍然在PooledQldbDriver物件而不是QldbDriver。如果您使用以下版本之一，請將QldbDriver取代為PooledQldbDriver瞭解本主題的其餘部分。

驅動程式	版本
Java	1.1.0或舊版
.NET	0.1.0-beta

驅動程式	版本
Node.js	1.0.0-rc.1 或舊版
Python	2.0.2或舊版

所以此PooledQldbDriver物件在最新版本的驅動程式中被棄用。建議您升級至最新版本，並將PooledQldbDriver至QldbDriver。

設定 QldbDriver 作為全局對象

要優化驅動程序和會話的使用，請確保您的應用程序實例中只有一個驅動程序的全局實例。例如，在 Java 中，您可以使用依賴注入框架，例如春天、Google Guice, 或匕首。以下程式碼範例會示範如何將QldbDriver作為一個單身人士。

```
@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();
}
```

配置重試嘗試

當常見瞬態異常（例如SocketTimeoutException或者NoHttpResponseException）發生。要設定重試次數上限，您可以使用maxRetries參數的transactionRetryPolicy配置對象時創建QldbDriver。（對於上一節中列出的較舊版本的驅動程序，請使用retryLimit參數PooledQldbDriver。）

maxRetries 的預設值為 4。

客戶端錯誤，例如InvalidParameterException無法重試。當它們發生時，事務將中止，會話返回到池，並將異常拋出給驅動程序的客戶端。

設定並行會話和事務的數目上限

實例使用的最大分類帳會話數QldbDriver來運行事務的定義是由其maxConcurrentTransactions參數。（對於上一節中列出的較舊的驅動程序版本，這是由poolLimit參數PooledQldbDriver。）

此限制必須大於零，且小於或等於會話客戶端允許的打開 HTTP 連接上限，如特定AWS開發套件。例如，在 Java 中，最大連接數在ClientConfiguration物件。

預設值為maxConcurrentTransactions是您的AWS開發套件。

當您配置QldbDriver，請考慮以下縮放注意事項：

- 池的會話數應始終至少與您計劃擁有的並行運行事務的數量相同。
- 在主管線程委派給工作線程的多線程模型中，驅動程序應至少具有與工作線程數相同的會話數。否則，在峯值負載時，線程將排隊等待可用會話。
- 每個分類帳的併發活動會話的服務限制在亞馬遜 QLDB 中的配額和限制。確保配置的併發會話不超過此限制，以用於所有客戶端的單個分類帳。

在異常情況下重試

在重試 QLDB 中發生的異常時，請考慮以下建議。

在 OCC 衝突時重試

樂觀並行控制(OCC) 衝突異常發生，當事務正在訪問的數據自事務開始以來發生更改。QLDB 在嘗試提交事務時拋出此異常。驅動程序重試事務多達maxRetries已配置。

有關 OCC 和使用索引限制 OCC 衝突的最佳實踐的詳細信息，請參閱Amazon QLLDB 並行模型。

重試 QLDB 驅動程序之外的其他異常

若要在運行時引發自定義的應用程序定義異常時在驅動程序之外重試事務，您必須包裝該事務。例如，在 Java 中，以下程式碼會示範如何使用雷斯利恩塞 4J庫來重試 QLDB 中的事務。

```
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**
>
> 在 QLDB 驅動程序之外重試事務會產生乘數效應。例如，如果`QldbDriver`配置為重試三次，並且自定義重試邏輯也重試三次，同一事務最多可以重試九次。

使交易冪等

作為最佳做法，使寫入事務具有冪等性，以避免在重試情況下出現任何意外的副作用。交易為冪等性如果它可以運行多次並且每次產生相同的結果。

如需進一步了解，請參閱 Amazon QLLDB 並行模型。

最佳化效能

要在使用驅動程序運行事務時優化性能，請注意以下事項：

- 所以此execute操作始終使至少三個SendCommandAPI 調用 QLDB，包括以下命令：

  1. StartTransaction
  2. ExecuteStatement

     此命令被調用為您在execute區塊。

  3. CommitTransaction

  計算應用程序的整體工作負載時，請考慮所進行的 API 調用總數。

- 通常，我們建議從單線程編寫器開始，並通過批處理單個事務中的多個語句來優化事務。最大限度地提高事務處理大小、文檔大小和每個事務處理的文檔數量的配額，如亞馬遜 QLDB 中的配額和限制。
- 如果批處理不足以滿足大型事務加載，您可以通過添加其他寫入程序來嘗試多線程。但是，您應該仔細考慮您的應用程序對文檔和事務排序的要求以及這帶來的額外複雜性。

每個事務運行多個語句

如上一節，您可以為每個事務運行多個語句以優化應用程序的性能。在下面的代碼示例中，查詢表，然後在事務中更新該表中的文檔。您可以通過將 lambda 表達式傳遞給executeoperation.

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))
```

驅動程式`execute`操作隱式啟動會話和該會話中的事務。在 lambda 表達式中運行的每個語句都包裝在事務中。所有語句運行後，驅動程序會自動提交事務。如果任何語句在用盡自動重試限制後失敗，則事務將中止。

傳播事務中的異常

當每個事務運行多個語句時，我們通常不建議您 catch 和吞嚥事務中的異常。

例如，在 Java 中，以下程序捕獲`RuntimeException`，記錄錯誤，然後繼續。此代碼示例被認為是不好的做法，因為即使`UPDATE`語句失敗。因此，客户端可能會假設更新成功，當它沒有成功。

> **Warning**
>
> 不要使用此代碼示例。它是為了顯示一個被認為是不好的做法的反模式示例。

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

傳播（冒泡）異常。如果事務的任何部分失敗，請讓`execute`操作中止事務，以便客户端可以相應地處理異常。

瞭解亞馬遜 QLDB 中的驅動程序的重試策略

Amazon QLDB 驅動程序使用重試策略通過透明地重試失敗的事務處理暫時異常。這些例外情況，例如`CapacityExceededException`和`RateExceededException`，通常經過一段期間後便會自行糾正。如果在適當的延遲後重試失敗但出現異常的事務，則很可能會成功。這有助於提高使用 QLDB 的應用程序的穩定性。

主題

- 可重試錯誤的類型
- 預設重試政策

可重試錯誤的類型

當且僅當事務處理中的操作期間發生以下任何異常時，驅動程序將自動重試事務：

- 容量超出欺騙— 當請求超過分類帳的處理能力時返回。
- 無效異常— 當會話不再有效或會話不存在時返回。
- LimitExceededException— 如果超出資源限制（如活動會話數），則返回。
- 八方衝突— 當交易由於在驗證階段失敗而無法寫入日誌時返回樂觀並行(OCC).
- 超越欺騙率— 當請求率超過允許的輸送。

預設重試政策

重試策略由重試條件和退避策略組成。重試條件定義應該重試事務的時間，而退避策略則定義在重試事務之前等待的時間。

創建驅動程序的實例時，默認重試策略指定最多重試四次，並使用指數退避策略。指數退避策略使用 10 毫秒的最小延遲和 5000 毫秒的最大延遲，抖動相等。如果無法在重試策略中成功提交事務，我們建議您在其他時間嘗試該事務。

指數退避的概念是，對於連續錯誤回應，讓重試之間的等待時間漸進拉長。如需詳細資訊，請參閱 。AWS部落格文章指數退避和抖動。

亞馬遜 QLDB 驅動程序的常見錯誤

本節說明與 QLDB 工作階段 API 互動時，Amazon QLDB 驅動程式可能會擲回的執行階段錯誤。

以下是驅動程式所傳回的常用例外狀況清單。每個例外都包含特定的錯誤訊息，後面接著簡短的描述和可能解決方案的建議。

CapacityExceededException

訊息：容量超過

Amazon QLDB 拒絕了該請求，因為它超出了分類帳的處理能力。QLDB 會針對每個分類帳強制執行內部調整規模限制，以維持服務的健康狀態和效能。此限制會根據每個個別要求的工作負載大小而有所不同。例如，如果要求執行效率低下的資料交易 (例如，非索引限定查詢所產生的資料表掃描)，工作負載可能會增加。

我們建議您等待，然後再重試請求。如果您的應用模組持續遇到此例外，請最佳化對帳單，並降低傳送至分類帳之請求的匯率與數量。陳述式最佳化的範例包括每個交易執行較少的陳述式以及調整資料表索引。若要瞭解如何最佳化陳述式並避免表格掃描，請參閱最佳化查詢效能。

我們亦建議您使用最新版本的 QLDB 驅動程式。驅動程式具有預設的重試原則，該原則會使用指數輪詢和抖動來自動重試例外狀況 (例如此類)。指數輪詢的概念是在連續錯誤響應的重試之間使用逐漸更長的等待時間。

InvalidSessionException

訊息：交易*## ID* 已過期

交易超過其最大生命週期。在認可之前，交易最多可以執行 30 秒。在此逾時限制之後，任何在交易上完成的工作都會遭到拒絕，而 QLDB 會捨棄工作階段。此限制可透過啟動交易而不提交或取消工作階段來保護用戶端不會洩漏工作階段。

如果這是應用程式中常見的例外狀況，則可能是交易執行時間太長。如果交易執行階段的時間超過 30 秒，請最佳化陳述式以加速交易。陳述式最佳化的範例包括每個交易執行較少的陳述式以及調整資料表索引。如需詳細資訊，請參閱最佳化查詢效能。

InvalidSessionException

訊息：*#### ID* 已過期

QLDB 捨棄工作階段，因為它超過其最大總存留期。QLDB 會在 13—17 分鐘後捨棄工作階段，不論作用中的交易為何。工作階段可能會因為多種原因而遺失或受損，例如硬體故障、網路故障或應用程式重新啟動。因此，QLDB 會強制執行工作階段的最大生命週期，以確保用戶端軟體能夠抵禦工作階段失敗。

如果您遇到此例外狀況，建議您取得新的階段作業，然後重試該交易。我們也建議您使用最新版本的 QLDB 驅動程式，該驅動程式會代表應用程式管理工作階段集區及其健全狀況。

InvalidSessionException

訊息：沒有此類工作階段

用戶端嘗試使用不存在的工作階段與 QLDB 進行交易。假設用戶端使用先前存在的工作階段，工作階段可能因為下列其中一種情況而不再存在：

- 如果工作階段涉及內部伺服器故障 (也就是 HTTP 回應碼為 500 的錯誤)，QLDB 可能會選擇完全捨棄工作階段，而不是允許客戶在不確定狀態的工作階段進行交易。然後，該會話上的任何重試嘗試都會失敗並顯示此錯誤。
- 過期的工作階段最終會被 QLDB 遺忘。然後，任何嘗試繼續使用會話都會導致此錯誤，而不是初始錯誤`InvalidSessionException`。

如果您遇到此例外狀況，建議您取得新的階段作業，然後重試該交易。我們也建議您使用最新版本的 QLDB 驅動程式，該驅動程式會代表應用程式管理工作階段集區及其健全狀況。

RateExceededException

訊息：超過比率

QLDB 限制了基於呼叫者的身份的客戶端。QLDB 會使用權杖儲存貯體節流演算法，針對每個區域、每個帳戶強制執行節流。QLDB 這樣做是為了協助服務效能，並確保所有 QLDB 客戶的公平使用。例如，嘗試使用該作業取得大量並行工`StartSessionRequest`作階段可能會導致節流。

若要維持應用程式健康狀態並減輕進一步的節流，您可以使用指數輪詢和抖動在此例外狀況上重試。指數輪詢的概念是在連續錯誤響應的重試之間使用逐漸更長的等待時間。我們建議您使用最新版本的 QLDB 驅動程式。驅動程式具有預設的重試原則，該原則會使用指數輪詢和抖動來自動重試例外狀況 (例如此類)。

如果您的應用程式持續受到 QLDB 限制`StartSessionRequest`呼叫，最新版本的 QLDB 驅動程式也會有所幫助。驅動程式會維護跨交易重複使用的工作階段集區，這有助於減少應用程式進行的`StartSessionRequest`呼叫次數。若要請求提高 API 調節限制，請與中AWS Support心聯絡。

LimitExceededException

訊息：超過工作階段限制

分類帳超過其作用中階段作業數目的配額 (也稱為限制)。此配額定義於中亞馬遜 QLDB 中的配額和限制。分類帳的作用中階段作業計數最終是一致的，且持續在配額附近執行的分類帳可能會定期看到此例外。

為了維持應用程式的健康狀態，我們建議您重試此例外狀況。若要避免此例外狀況，請確定您尚未針對所有用戶端的單一分類帳設定超過 1,500 個並行階段作業。例如，您可以使用適用於 Java 的 Amazon QLDB 驅動程式maxConcurrentTransactions方法來設定驅動程式執行個體中可用工作階段的數目上限。

QldbClientException

訊息：只有在上階交易開啟時，串流的結果才有效

交易已關閉，無法用於從 QLDB 擷取結果。交易在提交或取消時關閉。

當用戶端直接使用`Transaction`物件，並嘗試提交或取消交易之後從 QLDB 擷取結果時，就會發生此例外狀況。為了緩解此問題，客戶端必須在關閉交易之前讀取數據。

使用範例應用程式教學課程開始使用 Amazon QLDB

在本教學中，您將使用 Amazon QLDB 驅動程式搭配AWS開發套件來建立 QLDB 分類帳，並使用範例資料填入該分類帳。驅動程式可讓您的應用程式使用交易資料 API 與 QLDB 互動。AWSSDK 支援與 QLDB 資源管理 API 的互動。

您在此案例中建立的範例總帳是汽車部門 (DMV) 資料庫，可追蹤車輛註冊的完整歷史資訊。下列主題說明如何新增車輛註冊、修改以及檢視這些註冊的變更歷史記錄。本指南還向您展示如何以密碼方式驗證註冊文件，並透過清理資源並刪除範例分類帳來結束註冊文件。

範本應用程式教程可用於以下程式設計語言。

主題

- Amazon QLDB Java 教學
- Amazon QLDB Node.js 教程
- Amazon QLDB Python 教學

Amazon QLDB Java 教學

在此教學課程範例應用程式的實作中，您可以搭配使用 Amazon QLDB 驅動程式AWS SDK for Java來建立 QLDB 分類帳，並使用範例資料填入該分類帳。

在學習此教學課程時，您可以AWS SDK for Java參考《API 參考》。對於交易數據操作，您可以參考 Java API 參考的 QLDB 驅動程序。

> **Note**
>
> 在適用的情況下，某些教學課程步驟會針對每個支援的 Java QLDB 驅動程式主要版本提供不同的指令或程式碼範例。

主題

- 安裝亞馬遜 QLDB Java 範例應用程式
- 步驟 1：建立新分類帳
- 步驟 2：測試分類帳的連線
- 步驟 3：建立資料表、索引和範例資料

- 步驟 4：查詢分類帳中的表格
- 步驟 5：修改分類帳中的文件
- 步驟 6：檢視文件的修訂歷程記錄
- 步驟 7：驗證分類帳中的文件
- 步驟 8：匯出並驗證分類帳中的分錄資料
- 步驟 9 (選用)：清除資源

安裝亞馬遜 QLDB Java 範例應用程式

本節說明如何安裝和執行針對 step-by-step Java 教學課程提供的 Amazon QLDB 範例應用程式。此示例應用程序的用例是一個機動車輛部門（DMV）數據庫，用於跟踪有關車輛註冊的完整歷史信息。

適用於 Java 的 DMV 示例應用程序在 GitHub 存儲庫中是開源的 aws-Sample/amazon-qldb-dmv-sample-java。

先決條件

在開始之前，請確定您已完成適用於 Java 的 QLDB 驅動程式先決條件。這包含下列項目：

1. 註冊 AWS。
2. 建立具有適當 QLDB 權限的使用者。若要完成本教學課程中的所有步驟，您需要透過 QLDB API 存取分類帳資源的完整管理存取權。
3. 如果您使用的是 IDE 以外的其他 IDEAWS Cloud9，請安裝 Java 並授予程式設計存取權以進行開發。

安裝

下列步驟說明如何使用本機開發環境下載及設定範例應用程式。或者，您可以將範例應用AWS Cloud9程式設定自動化，方法是將您的 IDE 和AWS CloudFormation範本用於佈建您的開發資源。

本機開發環境

這些指示說明如何使用您自己的資源和開發環境下載和安裝 QLDB Java 範例應用程式。

下載並執行範例應用程式

1. 輸入下列命令以從中複製範例應用程式 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
```

此套件包含 Gradle 組態和來自Java 教學.

2. 載入並執行提供的應用程式。
   - 如果您使用的是日食：
     a. 啟動 Eclipse，並在 Eclipse 菜單上，選擇文件，導入，然後現有搖籃項目。
     b. 在專案根目錄中，瀏覽並選取包含該`build.gradle`檔案的應用程式目錄。然後，選擇「完成」以使用導入的默認 Gradle 設置。
     c. 你可以嘗試運行程`ListLedgers`序作為一個例子。開啟檔案的內容 (按一下右鍵) 開啟`ListLedgers.java`檔案的內容 (按一下右鍵) 開啟檔案的內容選單
   - 如果您使用的是 Intellij：
     a. 啟動 IntelliJ，然後在 In telliJ 功能表上選擇檔案，然後選擇開啟。
     b. 在專案根目錄中，瀏覽並選取包含該`build.gradle`檔案的應用程式目錄。然後，選擇「確定」。保留預設設定，然後再次選擇「確定」。
     c. 你可以嘗試運行程`ListLedgers`序作為一個例子。開啟檔案的內容 (按一下右鍵) 開啟`ListLedgers.java`檔案的內容 (按一下右鍵) 開啟檔案ListLedgers的內容
3. 繼續進行步驟 1：建立新分類帳以啟動自學課程並建立分類帳。

AWS Cloud9

這些指示說明如何將 Amazon QLDB 車輛註冊範例應用程式設定為 Java，並將其用AWS Cloud9作您的 IDE 自動化。在本指南中，您會使用AWS CloudFormation範本來佈建您的開發資源。

如需有關 AWS Cloud9 的詳細資訊，請參閱《使用者指南》AWS Cloud9。若要進一步了解 AWS CloudFormation，請參閱 AWS CloudFormation 使用者指南。

主題

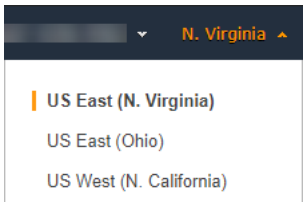
第 1 部分：佈建資源

在第一個步驟中，您可AWS CloudFormation以使用 Amazon QLDB 範例應用程式佈建設定開發環境所需的資源。

開啟AWS CloudFormation主控台並載入 QLDB 範例應用程式範本

1. 請登入 AWS Management Console，開啟位於 https://console.aws.amazon.com/cloudformation 的 AWS CloudFormation 主控台。

   切換至支援 QLDB 的地區。如需完整清單，請參閱中的 Amazon QLDB 端點和配額AWS 一般參考。下列的螢幕擷取得下列螢幕擷取得下列螢幕擷取得AWS Management Console了下列螢幕擷取得的畫AWS 區域面

2. 在AWS CloudFormation主控台上，選擇 [建立堆疊]，然後選擇 [使用新資源 (標準)]。
3. 在「建立堆疊」頁面的「指定範本」下，選擇 Amazon S3 URL。
4. 輸入下列 URL，然後選擇 [下一步]。

   ```
   https://amazon-qldb-assets.s3.amazonaws.com/templates/QLDB-DMV-SampleApp.yml
   ```

5. 輸入堆疊名稱 (例如**`qldb-sample-app`**)，然後選擇「下一步」。
6. 您可以視需要新增任何標籤，並保留預設選項。然後選擇 Next (下一步)。
7. 檢閱堆疊設定，然後選擇 [建立堆疊]。指AWS CloudFormation令碼可能需要幾分鐘時間才能完成完成設定。

   在本教程中，此指令碼使用相關的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體佈建您的AWS Cloud9環境，可供您用來執行 QLDB 範例應用程式。它還將 aws-Samples/amazon-qldb-dmv-sample-java 存儲庫從克隆 GitHub 到您的AWS Cloud9開發環境中。

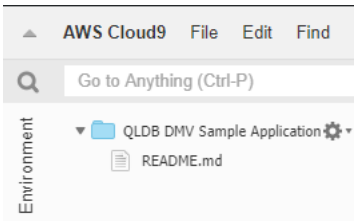
第 2 部分：設定您的 IDE

在此步驟中，您將完成設定您的開發環境。您可以下載並執行提供的 shell 指令碼，以使用範例應用程式的相依性來設定AWS Cloud9 IDE。

設定您的AWS Cloud9環境。

1. 請在以下位置開啟AWS Cloud9主控台。 https://console.aws.amazon.com/cloud9/
2. 在 [您的環境] 下，找出名為 QLDB DMV 範例應用程式之環境的介面卡，然後選擇 [開啟 IDE]。基礎 EC2 執行個體啟動時，您的環境可能需要一分鐘的時間才能載入。

   您的AWS Cloud9環境已預先設定好執行教學課程所需的系統相依性。在主控台的 [環境] 導覽窗格中，確認您看到名為的資料夾`QLDB DMV Sample Application`。AWS Cloud9主控台的下列螢幕擷取畫面顯示 QLDB DMV 範例應用程式環境資料夾窗格。

   如果您沒有看到導覽窗格，請切換主機左側的 [環境] 索引標籤。如果您在窗格中沒有看到任何資料夾，請使用設定圖示啟用「顯示環境根目錄」( )。
3. 在控制台的底部窗格中，您應該會看到一個打開的bash終端窗口。如果沒有看到此選項，請從主機頂端的「視窗」 選單中選擇「新增終端機」。
4. 接下來，下載並執行安裝程式指令碼以安裝 OpenJDK 8，如果適用的話，請從 Git 儲存庫中檢出適當的分支。在上一個步驟中建立的AWS Cloud9終端機中，執行下列兩個命令，按順序執行下列兩個命令：

   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
```

完成設定後，您應可看到在終端中打印下列訊息：

```
** DMV Sample App setup completed , enjoy!! **
```

5. 請花點時間瀏覽中的範例應用程式程式碼AWS Cloud9，特別是在下列目錄路徑中：`src/main/java/software/amazon/qldb/tutorial`。

第 3 部分：執行 QLDB DMV 範例應用程式

在此步驟中，您將學習如何使用執行 Amazon QLDB DMV 範例應用程式任務AWS Cloud9。要運行示例代碼，請返回AWS Cloud9終端機或創建一個新的終端機窗口，就像在第 2 部分：設置 IDE 中所做的那樣。

執行範例應用程式

1. 在終端機中執行下列命令以切換到專案根目錄：

```
cd ~/environment/amazon-qldb-dmv-sample-java
```

請確定您正在執行下列目錄路徑中的範例。

```
/home/ec2-user/environment/amazon-qldb-dmv-sample-java/
```

2. 下面的命令顯示了搖籃語法來運行每個任務。

```
./gradlew run -Dtutorial=Task
```

例如，執行下列命令以列出您AWS 帳戶的所有分類帳。

```
./gradlew run -Dtutorial=ListLedgers
```

3. 繼續進行步驟 1：建立新分類帳以啟動自學課程並建立分類帳。
4. (選擇性) 完成教學課程後，如果不再需要資AWS CloudFormation源，請清理您的資源。

   a. 在 https://console.aws.amazon.com/cloudformation 開啟AWS CloudFormation主控台，然後刪除您在第 1 部分：佈建資源中建立的堆疊。

   b. 同時刪除AWS CloudFormation範本為您建立的AWS Cloud9堆疊。

步驟 1：建立新分類帳

在此步驟中，您會建立名為的新 Amazon QLDB 分類帳`vehicle-registration`。

建立新分類帳

1. 檢閱下列 file (`Constants.java`)，其中包含本自學課程中所有其他程式所使用的常數值。

   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**
>
> 對於 Amazon Ion 套件，您必須在應用程式`com.amazon.ion`中使用命名空間。這取 AWS SDK for Java決於命名空間下的另一個 Ion 軟件包`software.amazon.ion`，但這是一個與 QLDB 驅動程序不兼容的舊版軟件包。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> 此Constants類別包含開放原始碼傑克遜IonValueMapper類別的執行個體。在進行讀取和寫入交易時，您可以使用此映射器來處理 Amazon Ion 數據。

該CreateLedger.java文件還具有以下程序（DescribeLedger.java），該程序描述了總帳的當前狀態的依賴關係。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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. 編譯並執行`CreateLedger.java`程式，以建立名為的分類帳`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**
>
> - 在`createLedger`通話中，您必須指定分類帳名稱和權限模式。建議您使用`STANDARD`許可模式來最大化分類帳的安全性。
> - 建立分類帳時，預設會啟用刪除保護。它是 QLDB 中的功能，可防止任何使用者刪除總帳。您可以選擇使用 QLDB API 或AWS Command Line Interface (AWS CLI) 停用分類帳建立時的刪除保護。
> - 您也可以選擇指定要附加至分類帳的盤點。

若要驗證您與新分類帳的連線，請繼續執行步驟 2：測試分類帳的連線。

步驟 2：測試分類帳的連線

在此步驟中，您確認可以使用交易資料 API 端點連接到 Amazon QLDB 中的`vehicle-registration`總帳。

測試分類帳的連線

1. 複查下列程式 (ConnectToLedger.java)，此程式會建立與vehicle-registration分類帳的資料階段作業連線。

   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**
>
> - 若要在分類帳上執行資料作業，您必須建立`QldbDriver`類別的執行環境，以連線至特定分類帳。這與您在上一個步驟中用來建立分類帳的用`AmazonQLDB`戶端不同的用戶端物件。先前的用戶端僅用於執行中列出的管理 API 作業Amazon QLDB API 參考參考。
> - 首先，創建一個`QldbDriver`對象。建立此動因時，您必須指定分類帳名稱。
>
>   然後，您可以使用此驅動程式的`execute`方法來執行 PartiQL 陳述式。
> - 您可以選擇性地指定異動例外的重試次數上限。此方`execute`法會自動重試樂觀並行控制 (OCC) 衝突及其他常見暫態例外狀況，達到此可設定的限制。預設值為 4。
>
>   如果達到限制後交易仍然失敗，則驅動程序拋出異常。如需進一步了解，請參閱 瞭解亞馬遜 QLDB 中的驅動程序的重試策略。

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

- 若要在分類帳上執行資料作業，您必須建立PooledQldbDriver或QldbDriver類別的執行環境，以連線至特定分類帳。這與您在上一個步驟中用來建立分類帳的用AmazonQLDB戶端不同的用戶端物件。先前的用戶端僅用於執行中列出的管理 API 作業Amazon QLDB API 參考參考。

  我們建議您使用，PooledQldbDriver除非您需要使用QldbDriver. 的預設集區大小PooledQldbDriver是工作階段用戶端允許的開啟 HTTP 連線數目上限。
- 首先，創建一個PooledQldbDriver對象。建立此動因時，您必須指定分類帳名稱。

  然後，您可以使用此驅動程式的execute方法來執行 PartiQL 陳述式。或者，您可以從此池驅動程序對象手動創建會話，並使用會話的execute方法。會話表示與分類帳的單個連接。
- 您可以選擇性地指定異動例外的重試次數上限。此方execute法會自動重試樂觀並行控制 (OCC) 衝突及其他常見暫態例外狀況，達到此可設定的限制。預設值為 4。

  如果達到限制後交易仍然失敗，則驅動程序拋出異常。如需進一步了解，請參閱 瞭解亞馬遜 QLDB 中的驅動程序的重試策略。

2. 編譯並執行ConnectToLedger.java程式，以測試資料階段作業與vehicle-registration分類帳的連線。

覆寫AWS 區域

範例應用程式會以預設值連線至 QLDBAWS 區域，您可以按照先決條件步驟中的說明進行設定設定您的預設AWS憑證和區域。您也可以透過修改 QLDB 工作階段用戶端產生器屬性來變更區域。

2.x

下列程式碼範例會實體化新QldbSessionClientBuilder物件。

```
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);
```

您可以使用此region方法，對 QLDB 執行程式碼。如需完整清單，請參閱中的 Amazon QLDB 端點和配額AWS 一般參考。

1.x

下列程式碼範例會實體化新AmazonQLDBSessionClientBuilder物件。

```
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);
```

您可以使用此withRegion方法，對 QLDB 執行程式碼。如需完整清單，請參閱中的 Amazon QLDB 端點和配額AWS 一般參考。

若要在vehicle-registration分類帳中建立表格，請繼續執行步驟 3：建立資料表、索引和範例資料。

步驟 3：建立資料表、索引和範例資料

當 Amazon QLDB 分類帳處於作用中狀態並接受連線時，您可以開始建立有關車輛、其擁有者及其註冊資訊的資料表。創建表和索引後，您可以使用數據加載它們。

在此步驟中，可在vehicle-registration分類帳中建立四個資料表：

- VehicleRegistration
- Vehicle
- Person
- DriversLicense

您也可以建立下列索引。

資料表名稱	欄位
VehicleRegistration	VIN
VehicleRegistration	LicensePlateNumber
Vehicle	VIN
Person	GovId
DriversLicense	LicenseNumber
DriversLicense	PersonId

插入範例資料時，首先將文件插入Person表格中。然後，您可以使用id從每個Person文件指派的系統來填入適當VehicleRegistration和DriversLicense文件中的對應欄位。

> **Tip**
>
> 最佳作法是使用文件的系統指派id為外部索引鍵。雖然您可以定義要做為唯一識別碼的欄位 (例如，車輛的 VIN)，但文件的真正唯一識別碼就是其id。此欄位包含在文件的中繼資料中，您可以在認可的檢視 (資料表的系統定義檢視) 中進行查詢。
> 如需 QLDB 中視圖的詳細資訊，請參閱核心概念。若要進一步了解中繼資料，請參閱查詢文件元資料。

設定範例資料

1. 檢閱下列.java檔案。這些模型類別代表您儲存在vehicle-registration表格中的文件。它們可序列化，並從亞馬遜離子格式。

> **Note**
>
> Amazon QLDB 文件以 Ion 格式儲存，亦即 JSON 的超集。所以，你可以使用更快的 XML 傑克遜庫模型 JSON 中的數據。

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. 檢閱下列檔案 (SampleData.java)，它代表您插入到vehicle-registration表格中的範例資料。

   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**
>
> 對於 Amazon Ion 套件，您必須在應用程式`com.amazon.ion`中使用命名空間。這取 AWS SDK for Java決於命名空間下的另一個 Ion 軟件包`software.amazon.ion`，但這是一個與 QLDB 驅動程序不兼容的舊版軟件包。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
```

```
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> - 這個類使用 Ion 庫來提供幫助方法，將數據轉換為離子格式。
> - 該`getDocumentId`方法在具有前綴的表上運行查詢`_ql_committed_`。這是一個保留的前綴，表示您要查詢表的提交視圖。在此檢視中，您的資料以巢狀方式嵌套在`data`欄位中，而中繼資料則以巢狀方式嵌套在`metadata`欄位中。

3. 編譯並運行下面的程序（`CreateTable.java`）來創建前面提到的表。

   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**
>
> 該程序演示瞭如何將TransactionExecutor lambda 傳遞給該execute方法。在此範例中，您可以使用 Lambda 運算式，在單一交易中執行多個CREATE TABLE PartiQL 陳述式。
> 該execute方法隱含地啟動一個事務，運行 lambda 中的所有語句，然後自動提交交易。

4. 如先前所述，編譯並執行下列程式 (CreateIndex.java) 以在資料表上建立索引。

   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. 編譯並執行下列程式 (InsertDocument.java)，將範例資料插入資料表中。

   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**
>
> - 該程序演示如何調用具有參數化值的`execute`方法。除了要執行的 PartiQL 陳述`IonValue`式之外，您還可以傳遞類型的資料參數。在陳述式字串中使用問號 (?) 做為變數預留位置。
> - 如果`INSERT`陳述式成功，則會傳回每`id`個插入文件的。

接下來，您可以使用`SELECT`語句從`vehicle-registration`分類帳中的表中讀取數據。繼續執行「步驟 4：查詢分類帳中的表格」。

步驟 4：查詢分類帳中的表格

在 Amazon QLDB 分類帳中建立表格並將其載入資料後，您可以執行查詢以檢閱剛插入的車輛註冊資料。QLDB 使用 PartiQL 作為其查詢語言，而亞馬遜離子作為其文件導向的資料模型。

PartiQL 是一種開放原始碼、SQL 相容的查詢語言，已經擴充為與 Ion 搭配使用。使用 PartiQL，您可以使用熟悉的 SQL 運算子插入、查詢和管理資料。亞馬遜離子是 JSON 的超集合。Ion 是一種開放原始碼、以文件為基礎的資料格式，可讓您靈活地儲存和處理結構化、半結構化和巢狀資料。

在此步驟中，您可以使用`SELECT`陳述式從`vehicle-registration`分類帳中的表格讀取資料。

> **Warning**
>
> 當您在沒有索引查閱的情況下在 QLDB 中執行查詢時，它會叫用完整資料表掃描。PartiQL 支持這樣的查詢，因為它是 SQL 兼容。不過，請勿在 QLDB 中針對生產使用案例執行資料表掃描。資料表掃描可能會造成大型資料表的效能問題，包括並行衝突和交易逾時。
> 若要避免資料表掃描，您必須在索引欄位或文件 ID 上使用相等運算子來執行具有述`WHERE`詞子句的陳述式；例如，`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`。如需詳細資訊，請參閱最佳化查詢效能。

若要查詢資料表

- 編譯並運行以下程序（`FindVehicles.java`）以查詢分類帳中某個人下註冊的所有車輛。

  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**
>
> 首先，這個程Person序查詢與GovId LEWISR261LL獲取其id元數據字段的文檔表。然後，它使用此文檔id作為外鍵來查詢VehicleRegistration表PrimaryOwner.PersonId。它也會VehicleRegistration與VIN欄位上的Vehicle表格連接在一起。

若要瞭解如何在vehicle-registration分類帳中修改表格中的文件，請參閱步驟 5：修改分類帳中的文件。

步驟 5：修改分類帳中的文件

既然您有資料需要使用，就可以開始變更 Amazon QLDB 中vehicle-registration分類帳中的文件。在此步驟中，下列程式碼範例會示範如何執行資料操作語言 (DML) 陳述式。這些聲明更新了一輛車的主要車主，並將次要車主添加到另一輛車。

若要修改文件

1. 編譯並執行下列程式 (TransferVehicleOwnership.java)，以分類帳1N4AL11D75C109151中的 VIN 更新車輛的主要擁有者。

   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. 編譯並執行下列程式 (AddSecondaryOwner.java)，以KM8SRDHF6EU074761在分類帳中使用 VIN 將次要擁有者新增至車輛。

   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.");
    }
}
```

若要複查vehicle-registration分類帳中的這些變更，請參閱步驟 6：檢視文件的修訂歷程記錄。

步驟 6：檢視文件的修訂歷程記錄

在上一個步驟中修改車輛的註冊資料後，您可以查詢其所有註冊車主的歷史記錄以及任何其他更新的欄位。在此步驟中，您會查詢vehicle-registration分類帳VehicleRegistration表格中文件的修訂歷史記錄。

若要檢視修訂歷史記錄

1. 檢閱下列程式 (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

- 您可以使用下列語法查詢內建的文件，以檢視文件[歷史功能](#)的修訂歷程記錄。

```
SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
```

```
[ WHERE h.metadata.id = 'id' ]
```

- 開始時間和結束時間都是可選的。它們是亞馬遜離子字面值，可以用反引號 (`...`) 表示。如需進一步了解，請參閱 在亞馬遜 QLDB 中使用 PartiQL 查詢離子。
- 最佳作法是使用日期範圍 (開始時間和結束時間) 和文件 ID (metadata.id) 來限定歷史查詢。QLDB 處理交易中的SELECT查詢，這些交易受到交易逾時限制。

  QLDB 歷程記錄會依文件 ID 編製索引，而且您目前無法建立其他歷程記錄索引。包含開始時間和結束時間的歷史查詢可獲得日期範圍限定的好處。

2. 編譯並運行該QueryHistory.java程序以使用 VIN 查詢VehicleRegistration文檔的修訂歷史記錄1N4AL11D75C109151。

若要以密碼編譯方式驗證vehicle-registration分類帳中的文件版次，請繼續執行步驟 7：驗證分類帳中的文件。

步驟 7：驗證分類帳中的文件

使用 Amazon QLDB，您可以使用搭配 SHA-256 的加密雜湊，有效地驗證分類帳日誌中文件的完整性。若要深入瞭解驗證和加密雜湊如何在 QLDB 中運作，請參閱Amazon QLDB 中的數據驗證。

在此步驟中，您會核對vehicle-registration分類帳中VehicleRegistration表格中的文件修訂。首先，您要求摘要，該摘要會以輸出檔案的形式傳回，並做為分類帳整個變更歷史記錄的簽章。然後，您要求相對於該摘要的修訂版本的證明。使用此證明，如果所有驗證檢查都通過，則會驗證修訂的完整性。

核對文件修訂

1. 檢閱下列.java檔案，這些檔案代表驗證所需的 QLDB 物件，以及 Ion 和字串值的輔助方法公用程式類別。

   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. 使用兩個.java檔案 (GetDigest.java和GetRevision.java) 來執行下列步驟：

   - 請求分vehicle-registration類帳中的新摘要。
   - 請求表格中文件的每個修訂版本的證VehicleRegistration明。
   - 透過重新計算摘要，使用傳回的摘要和校樣來驗證修訂。

   該GetDigest.java程序包含以下代碼。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> 使用該getDigest方法請求摘要，其中涵蓋分類帳中日誌的當前提示。日誌的提示是指 QLDB 收到您的請求時最新的已提交區塊。

該GetRevision.java程序包含以下代碼。

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**
>
> 在方`getRevision`法傳回指定文件修訂的證明之後，此程式會使用用戶端 API 來驗證該修訂。如需此 API 所使用演算法的概觀，請參閱使用證明重新計算摘要。

3. 編譯並運行該`GetRevision.java`程序以密碼編譯驗證與 VIN 的`VehicleRegistration`文檔`1N4AL11D75C109151`。

若要匯出並驗證分`vehicle-registration`類帳中的分錄資料，請繼續執行步驟 8：匯出並驗證分類帳中的分錄資料。

步驟 8：匯出並驗證分類帳中的分錄資料

在 Amazon QLDB 中，您可以存取分類帳中的日誌內容，用於各種目的，例如資料保留、分析和稽核。如需詳細資訊，請參閱從 Amazon QLDB 匯出日誌資料。

在此步驟中，您將日誌區塊從分`vehicle-registration`類帳匯出到 Amazon S3 儲存貯體。然後，您可以使用匯出的資料來驗證日誌區塊與每個區塊中個別雜湊元件之間的雜湊鏈。

您使用的AWS Identity and Access Management (IAM) 主體實體必須具有足夠的 IAM 許可 Amazon S3 能在您的AWS 帳戶. 如需詳細資訊，請參閱 Amazon S3 使用者指南中的 Amazon S3 的原則和許可。您還必須擁有許可才能建立具有連接許可政策的 IAM 角色，該政策允許 QLDB 將物件寫入 Amazon S3 儲存貯體。若要進一步了解，請參閱 IAM 使用者指南中的存取 IAM 資源所需的許可。

匯出與驗證分錄資料的步驟

1. 檢閱下列檔案 (JournalBlock.java)，它代表日誌區塊及其資料內容。它包括一個名為 演verifyBlockHash()示如何計算塊散列的每個單獨組件的方法。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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. 編譯並運行以下程序（ValidateQldbHashChain.java）來執行以下步驟：

   1. 將日誌區塊從分vehicle-registration類帳匯出到名為的 Amazon S3 儲存貯體**qldb-tutorial-journal-export-*111122223333***(以您的AWS 帳戶編號取代)。
   2. 通過調用驗證每個塊中的單個哈希組件verifyBlockHash()。
   3. 驗證日誌區塊之間的雜湊鏈。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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;
        }
    }

}
```

如果您不再需要使用vehicle-registration分類帳，請繼續執行步驟 9 (選用)：清除資源。

步驟 9 (選用)：清除資源

您可以繼續使用分vehicle-registration類帳。但是，如果您不再需要它，則應將其刪除。

若要刪除分類帳

1. 編譯並執行下列程式 (DeleteLedger.java)，以刪除vehicle-registration分類帳及其所有內容。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
```

```
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> 如果已啟用分類帳的刪除保護，您必須先停用它，然後才能使用 QLDB API 刪除總帳。

2. 如果您在上一步中匯出了日誌資料，但不再需要它，請使用 Amazon S3 主控台刪除 S3 儲存貯體。

   請在 https://console.aws.amazon.com/s3/ 開啟 Amazon Simple Storage Service (Amazon S3) 主控台。

Amazon QLDB Node.js 教程

在此教學課程範例應用程式的實作中，您可以在 Node.js JavaScript 中使用 Amazon QLDB 驅動程式搭配AWS開發套件來建立 QLDB 分類帳，並使用範例資料填入該分類帳。

在學習本教學課程時，您可以參考管理 AWS SDK for JavaScript API 操作的 API 參考。對於交易數據操作，您可以參考 Node.js API 參考的 QLDB 驅動程序。

> **Note**
>
> 在適用的情況下，某些教學課程步驟會針對每個支援的 Node.js QLDB 驅動程式主要版本提供不同的命令或程式碼範例。

主題

安裝 Amazon QLDB Node.js 示例應用程序

本節說明如何安裝和執行針對 step-by-step Node.js 教學課程提供的 Amazon QLDB 範例應用程式。此示例應用程序的用例是一個機動車輛部門（DMV）數據庫，用於跟踪有關車輛註冊的完整歷史信息。

Node.js 的 DMV 範例應用程式在 GitHub 儲存庫中是開放原始碼的 AWS 範例/-amazon-qldb-dmv-sample nodejs。

必要條件

在開始之前，請確定您已完成 Node.js 的 QLDB 驅動程式。必要條件這包括安裝 Node.js 並執行以下操作：

1. 註冊 AWS。
2. 建立具有適當 QLDB 權限的使用者。
3. 授予程序化訪問以進行開發。

若要完成本教學課程中的所有步驟，您需要透過 QLDB API 存取分類帳資源的完整管理存取權。

安裝

若要安裝範例應用程式

1. 輸入下列命令以從中複製範例應用程式 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
   ```

   範例應用程式會封裝本教學課程中的完整原始程式碼及其相依性，包括 Node.js 驅動程式和 Node.js JavaScript 中的 AWS SDK。這個應用程序是寫在 TypeScript.

2. 切換至要複製`amazon-qldb-dmv-sample-nodejs`套件的目錄。

   ```
   cd amazon-qldb-dmv-sample-nodejs
   ```

3. 做一個乾淨的依賴關係安裝。

   ```
   npm ci
   ```

4. 移植包。

   ```
   npm run build
   ```

   轉譯的 JavaScript 文件被寫入`./dist`目錄中。

5. 繼續進行步驟 1：建立新分類帳以啟動自學課程並建立分類帳。

步驟 1：建立新分類帳

在此步驟中，您會建立名為的新 Amazon QLDB 分類帳。`vehicle-registration`

若要建立新的分類帳

1. 檢閱下列 file (Constants.ts)，其中包含本自學課程中所有其他程式所使用的常數值。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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. 使用下列程式 (CreateLedger.ts) 建立名為的分類帳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**
>
> - 在createLedger通話中，您必須指定分類帳名稱和權限模式。我們建議您使用STANDARD權限模式來最大化分類帳資料的安全性。
> - 建立分類帳時，依預設會啟用刪除保護。這是 QLDB 中的一項功能，可防止任何使用者刪除分類帳。您可以選擇使用 QLDB API 或 AWS Command Line Interface (AWS CLI) 停用分類帳建立時的刪除保護。
> - 您也可以選擇性地指定要附加至分類帳的盤點單。

3. 要運行轉譯程序，請輸入以下命令。

```
node dist/CreateLedger.js
```

若要驗證您與新分類帳的連線，請繼續執行步驟 2：測試與分類帳的連線。

步驟 2：測試與分類帳的連線

在此步驟中，您確認可以使用交易資料 API 端點連接到 Amazon QLDB 中的`vehicle-registration`總帳。

若要測試與分類帳的連線

1. 使用下列程式 (`ConnectToLedger.ts`) 建立與`vehicle-registration`分類帳的資料階段作業連線。

   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**
>
> 若要在分類帳上執行資料交易，您必須建立 QLDB 驅動程式物件以連線至指定的分類帳。這與您在上一個步驟中用來建立分類帳的`qldbClient`物件不同的用戶端物件。先前的用戶端僅用於執行中列出的管理 API 作業Amazon QLDB API 參考參考。

2. 要運行轉譯程序，請輸入以下命令。

```
node dist/ConnectToLedger.js
```

若要在`vehicle-registration`分類帳中建立表格，請繼續執行步驟 3：建立資料表、索引和範例資料。

步驟 3：建立資料表、索引和範例資料

當 Amazon QLDB 分類帳處於作用中狀態並接受連線時，您可以開始建立有關車輛、其擁有者及其註冊資訊的資料表。創建表和索引後，您可以使用數據加載它們。

在此步驟中，您可以在`vehicle-registration`分類帳中建立四個表格：

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

您也可以建立下列索引。

資料表名稱	欄位
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`

資料表名稱	欄位
DriversLicense	LicenseNumber
DriversLicense	PersonId

插入範例資料時，首先將文件插入Person表格中。然後，您可以使用id從每個Person文件指派的系統來填入適當VehicleRegistration和DriversLicense文件中的對應欄位。

> **Tip**
>
> 最佳作法是使用文件的系統指派id為外部索引鍵。雖然您可以定義要做為唯一識別碼的欄位 (例如，車輛的 VIN)，但文件的真正唯一識別碼就是其id。此欄位包含在文件的中繼資料中，您可以在認可的檢視 (資料表的系統定義檢視) 中進行查詢。
> 如需 QLDB 中視圖的詳細資訊，請參閱。核心概念若要進一步瞭解中繼資料，請參閱查詢文件元資料。

若要建立表格和索引

1. 使用下面的程序（CreateTable.ts）來創建前面提到的表。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
```

```
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> 該程序演示如何在 QLDB 驅動程序實例中使用該executeLambda函數。在此範例中，您會使用單一 Lambda 運算式執行多個 CREATE TABLE PartiQL 陳述式。
> 這個執行函數隱含地啟動一個事務，運行 lambda 中的所有語句，然後自動提交交易。

2. 要運行轉譯程序，請輸入以下命令。

```
node dist/CreateTable.js
```

3. 如前所述，使用下面的程序（CreateIndex.ts）在表上創建索引。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
```

```
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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. 要運行轉譯程序，請輸入以下命令。

```
node dist/CreateIndex.js
```

若要將資料載入表格

1. 檢閱下列.ts檔案。

    1. SampleData.ts— 包含您插入到表格中的範例資vehicle-registration料。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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`— 從`ion-js`套件匯入的公用程式模組，可提供轉換、剖析和列印 Amazon Ion 資料的輔助函數。

```
/*
```

```
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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函數會執行查詢，該查詢會從資料表傳回系統指派的文件 ID。如需進一步了解，請參閱 使用 BY 子句來查詢文件 ID。

2. 使用下面的程序（InsertDocument.ts）插入樣本數據到您的表。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> - 該程序演示如何調用具有參數化值的`execute`函數。除了要執行的 PartiQL 陳述式之外，您還可以傳遞資料參數。在陳述式字串中使用問號 (?) 做為變數預留位置。
> - 如果`INSERT`陳述式成功，則會傳回每個插入文件`id`的。

3. 要運行轉譯程序，請輸入以下命令。

```
node dist/InsertDocument.js
```

接下來，您可以使用`SELECT`語句從`vehicle-registration`分類帳中的表中讀取數據。繼續執行「步驟 4：查詢分類帳中的表格」。

步驟 4：查詢分類帳中的表格

在 Amazon QLDB 分類帳中建立表格並將其載入資料後，您可以執行查詢以檢閱剛插入的車輛註冊資料。QLDB 使用 PartiQL 作為其查詢語言，而 Amazon 離子作為其文件導向的資料模型。

PartiQL 是一種開放原始碼、SQL 相容的查詢語言，已經擴充為與 Ion 搭配使用。使用 PartiQL，您可以使用熟悉的 SQL 運算子插入、查詢和管理資料。Amazon 離子是 JSON 的超集合。Ion 是一種開放原始碼、以文件為基礎的資料格式，可讓您靈活地儲存和處理結構化、半結構化和巢狀資料。

在此步驟中，您可以使用`SELECT`陳述式從`vehicle-registration`分類帳中的表格讀取資料。

> **Warning**
>
> 當您在沒有索引查閱的情況下在 QLDB 中執行查詢時，它會叫用完整資料表掃描。PartiQL 支持這樣的查詢，因為它是 SQL 兼容。不過，請勿在 QLDB 中針對生產使用案例執行資料表掃描。資料表掃描可能會造成大型資料表的效能問題，包括並行衝突和交易逾時。
> 若要避免資料表掃描，您必須在索引欄位或文件 ID 上使用相等運算子來執行具有述`WHERE`詞子句的陳述式；例如，`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`。如需詳細資訊，請參閱最佳化查詢效能。

若要查詢資料表

1. 使用以下程序（`FindVehicles.ts`）查詢在分類帳中以人員註冊的所有車輛。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
```

```
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> 首先，這個程Person序查詢與GovId LEWISR261LL獲取其id元數據字段的文檔表。然後，它使用此文檔id作為外鍵來查詢VehicleRegistration表PrimaryOwner.PersonId。它還VehicleRegistration與VIN字段上的Vehicle表格連接。

2. 要運行轉譯程序，請輸入以下命令。

```
node dist/FindVehicles.js
```

若要瞭解如何在vehicle-registration分類帳中修改表格中的文件，請參閱步驟 5：修改分類帳中的文件。

步驟 5：修改分類帳中的文件

既然您有資料需要使用，就可以開始變更 Amazon QLDB 中vehicle-registration分類帳中的文件。在此步驟中，下列程式碼範例會示範如何執行資料操作語言 (DML) 陳述式。這些聲明更新了一輛車的主要車主，並將次要車主添加到另一輛車。

若要修改文件

1. 使用下列程式 (TransferVehicleOwnership.ts)，以分類帳1N4AL11D75C109151中的 VIN 更新車輛的主要擁有者。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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. 要運行轉譯程序，請輸入以下命令。

```
node dist/TransferVehicleOwnership.js
```

3. 使用下列程式 (AddSecondaryOwner.ts) 將次要擁有者新增至分類帳中含有 V KM8SRDHF6EU074761 IN 的車輛。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
```

```
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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. 要運行轉譯程序，請輸入以下命令。

```
node dist/AddSecondaryOwner.js
```

若要複查vehicle-registration分類帳中的這些變更，請參閱步驟 6：檢視文件的修訂歷程記錄。

步驟 6：檢視文件的修訂歷程記錄

在上一個步驟中修改車輛的註冊資料後，您可以查詢其所有註冊車主的歷史記錄以及任何其他更新的欄位。在此步驟中，您會查詢vehicle-registration分類帳VehicleRegistration表格中文件的修訂歷史記錄。

若要檢視修訂歷史記錄

1. 使用下面的程序（QueryHistory.ts）用 VIN 查詢VehicleRegistration文檔的修訂歷史記錄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

- 您可以使用下列語法查詢內建的文件，以檢視文件歷史功能的修訂歷程記錄。

  ```
  SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
  [ WHERE h.metadata.id = 'id' ]
  ```

- 開始時間和結束時間都是可選的。它們是 Amazon 離子字面值，可以用反引號 ( ) `...`表示。如需詳細資訊，請參閱在亞馬遜 QLDB 中使用 PartiQL 查詢離子。
- 最佳作法是使用日期範圍 (開始時間和結束時間) 和文件 ID (`metadata.id`) 來限定歷史查詢。QLDB 處理交易中的SELECT查詢，這些交易受到交易逾時限制。

> QLDB 歷程記錄會依文件 ID 編製索引，而且您目前無法建立其他歷程記錄索引。包含開始時間和結束時間的歷史查詢可獲得日期範圍限定的好處。

2. 要運行轉譯程序，請輸入以下命令。

```
node dist/QueryHistory.js
```

若要以密碼編譯方式驗證vehicle-registration分類帳中的文件版次，請繼續執行步驟 7：驗證分類帳中的文件。

步驟 7：驗證分類帳中的文件

使用 Amazon QLDB，您可以使用搭配 SHA-256 的加密雜湊，有效地驗證分類帳日誌中文件的完整性。若要深入瞭解驗證和加密雜湊如何在 QLDB 中運作，請參閱。Amazon QLDB 中的數據驗證

在此步驟中，您會核對vehicle-registration分類帳VehicleRegistration表格中的文件修訂。首先，請求摘要，該摘要作為輸出文件返回，並充當分類帳的整個變更歷史記錄的簽名。然後，您要求相對於該摘要的修訂版本的證明。使用此證明，如果所有驗證檢查都通過，則會驗證修訂的完整性。

核對文件修訂

1. 檢閱下列.ts檔案，其中包含驗證所需的 QLDB 物件。

    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. 使用兩個.ts程式 (GetDigest.ts和GetRevision.ts) 來執行下列步驟：

   - 請求分vehicle-registration類帳中的新摘要。
   - 請1N4AL11D75C109151從表格中使用 VIN 為文檔的每個修訂版提供證VehicleRegistration明。
   - 透過重新計算摘要，使用傳回的摘要和校樣來驗證修訂。

   該GetDigest.ts程序包含以下代碼。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
```

```
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> 使用此getDigest函數可要求摘要，其中涵蓋分類帳中分類帳目前提示的摘要。日誌的提示是指 QLDB 收到您的請求時最新的已提交區塊。

該GetRevision.ts程序包含以下代碼。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> 在getRevision函數傳回指定文件修訂的證明之後，此程式會使用用戶端 API 來驗證該修訂版本。

3. 要運行轉譯程序，請輸入以下命令。

```
node dist/GetRevision.js
```

如果您不再需要使用vehicle-registration分類帳，請繼續執行步驟 8 (選擇性)：清理資源。

步驟 8 (選擇性)：清理資源

您可以繼續使用分vehicle-registration類帳。但是，如果不再需要它，則應將其刪除。

若要刪除分類帳

1. 使用以下程序（DeleteLedger.ts）刪除vehicle-registration分類帳及其所有內容。

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> 如果已啟用分類帳的刪除保護，您必須先停用它，然後才能使用 QLDB API 刪除分類帳。

2. 要運行轉譯程序，請輸入以下命令。

```
node dist/DeleteLedger.js
```

Amazon QLDB Python 教學

在此教學課程範例應用程式的實作中，您可以搭配使用 Amazon QLDB 驅動程式AWS SDK for Python (Boto3)來建立 QLDB 分類帳，並使用範例資料填入該分類帳。

跟著本教學進行操作的同時，可以參 QLDBAWS SDK to 3 API 參考 API 操作的同時，可以參考 SDK to 3 API 參考。對於事務性數據操作，您可以參考 QLDB 驅動程序的 Python API 參考。

> **Note**
>
> 在適用的情況下，某些教學課程步驟會針對每個受支援的 Python QLDB 驅動程式主要版本提供不同的指令或程式碼範例。

主題

安裝亞馬遜 QLDB Python 示例應用程序

本節說明如何安裝和執行針對 step-by-step Python 教學課程所提供的 Amazon QLDB 範例應用程式。此示例應用程序的用例是一個機動車輛部門（DMV）數據庫，用於跟踪有關車輛註冊的完整歷史信息。

適用於 Python 的 DMV 示例應用程序是 GitHub 存儲庫中的開源軟件庫 aws-樣本/amazon-qldb-dmv-sample-python。

先決條件

跟著 Python 進行操作的同時，請確定您已完成 QLDB 驅動程式先決條件。這包括安裝 Python 並執行以下操作：

1. 註冊 AWS。
2. 建立具有適當 QLDB 權限的使用者。
3. 授予程序化訪問以進行開發。

若要完成本教學課程中的所有步驟，您需要透過 QLDB API 存取分類帳資源的完整管理存取權。

安裝

安裝範例應用程式

1. 輸入下列pip命令，以從中複製並安裝範例應用程式 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
   ```

   範例應用程式會封裝本教學課程中的完整原始程式碼及其相依性，包括 Python 驅動程式和 AWS SDK for Python (Boto3).

2. 在命令列開始執行程式碼之前，請先將目前的工作目錄切換到安裝pyqldbsamples套件的位置。輸入以下命令。

   ```
   cd $(python -c "import pyqldbsamples; print(pyqldbsamples.__path__[0])")
   ```

3. 繼續進行步驟 1：建立新的分類帳以啟動自學課程並建立分類帳。

步驟 1：建立新的分類帳

在此步驟中，您會建立名為的新 Amazon QLDB 分類帳vehicle-registration。

建立新分類帳

1. 檢閱下列 file (constants.py)，其中包含本自學課程中所有其他程式所使用的常數值。

   ```
   # Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: MIT-0
   #
   # Permission is hereby granted, free of charge, to any person obtaining a copy of
    this
   ```

```
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# 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. 使用下列程式 (`create_ledger.py`) 建立名為的分類帳`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**
>
> - 在create_ledger通話中，您必須指定分類帳名稱和權限模式。建議您使用STANDARD許可模式來最大化分類帳資料的安全性。
> - 建立分類帳的同時，預設會啟用刪除保護功能。QLDB 中的一項功能可防止任何使用者刪除總帳。您可以選擇使用 QLDB API 或AWS Command Line Interface (AWS CLI) 停用分類帳建立時的刪除保護。
> - 您也可以選擇指定要附加至分類帳的盤點單。

3. 若要執行程式，請輸入下列命令。

```
python create_ledger.py
```

若要驗證您與新分類帳的連線，請繼續執行步驟 2：測試與分類帳的連線。

步驟 2：測試與分類帳的連線

在此步驟中，您確認可以使用交易資料 API 端點連接到 Amazon QLDB 中的vehicle-registration總帳。

測試與分類帳的連線

1. 使用下列程式 (connect_to_ledger.py) 建立與vehicle-registration分類帳的資料階段作業連線。

   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**
>
> - 若要在分類帳上執行資料交易，您必須建立 QLDB 驅動程式物件以連線至特定分類帳。這與您在上一個步驟中用來建立分類帳的`qldb_client`物件不同的用戶端物件。先前的用戶端僅用於執行中列出的管理 API 作業Amazon QLDB API 參考參考。
> - 建立此動因物件時，您必須指定分類帳名稱。

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**
>
> - 若要在分類帳上執行資料交易，您必須建立 QLDB 驅動程式物件以連線至特定分類帳。這與您在上一個步驟中用來建立分類帳的`qldb_client`物件不同的用戶端物件。先前的用戶端僅用於執行中列出的管理 API 作業Amazon QLDB API 參考參考。
> - 首先，建立集區的 QLDB 驅動程式物件。建立此動因時，您必須指定分類帳名稱。
> - 然後，您可以從這個集區的驅動程式物件建立工作階段。

2. 若要執行程式，請輸入下列命令。

```
python connect_to_ledger.py
```

若要在`vehicle-registration`分類帳中建立表格，請繼續執行步驟 3：建立資料表、索引和範例資料。

步驟 3：建立資料表、索引和範例資料

當 Amazon QLDB 分類帳處於作用中狀態並接受連線時，您可以開始建立有關車輛、其擁有者及其註冊資訊的資料表。創建表和索引後，您可以使用數據加載它們。

在此步驟中，可在`vehicle-registration`分類帳中建立四個表格：

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

您也可以建立下列索引。

資料表名稱	欄位
`VehicleRegistration`	`VIN`

資料表名稱	欄位
VehicleRegistration	LicensePlateNumber
Vehicle	VIN
Person	GovId
DriversLicense	LicenseNumber
DriversLicense	PersonId

插入範例資料時，首先將文件插入Person表格中。然後，您可以使用id從每個Person文件指派的系統來填入適當VehicleRegistration和DriversLicense文件中的對應欄位。

> **Tip**
>
> 最佳作法是使用文件的系統指派id為外部索引鍵。雖然您可以定義要做為唯一識別碼的欄位 (例如，車輛的 VIN)，但文件的真正唯一識別碼就是其id。此欄位包含在文件的中繼資料中，您可以在認可的檢視 (資料表的系統定義檢視) 中進行查詢。
> 如需 QLDB 中視圖的詳細資訊，請參閱核心概念。若要進一步了解中繼資料，請參閱查詢文件元資料。

建立資料表及索引

1. 使用下面的程序（`create_table.py`）來創建前面提到的表。

   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**
>
> 本程式會示範execute_lambda函數的使用方式。在此範例中，您會使用單一 Lambda 運算式執行多個CREATE TABLE PartiQL 陳述式。
> 這個執行函數隱含地啟動一個事務，運行 lambda 中的所有語句，然後自動提交交易。

2. 若要執行程式，請輸入下列命令。

```
python create_table.py
```

3. 如前所述，使用下面的程序（create_index.py）在表上創建索引。

   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. 若要執行程式，請輸入下列命令。

```
python create_index.py
```

將資料載入資料表

1. 檢閱下列檔案 (sample_data.py)，它代表您插入到vehicle-registration表格中的範例資料。此檔案也會從amazon.ion套件匯入，以提供轉換、剖析和列印 Amazon Ion 資料的輔助函數。

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# 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函數會執行查詢，該查詢會從資料表傳回系統指派的文件 ID。如需進一步了解，請參閱 使用 BY 子句來查詢文件 ID。

2. 使用下面的程序（insert_document.py）插入樣本數據到您的表。

   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**
>
> - 該程序演示如何調用具有參數化值的`execute_statement`函數。除了要執行的 PartiQL 陳述式之外，您還可以傳遞資料參數。在陳述式字串中使用問號 (?) 做為變數預留位置。
> - 如果`INSERT`陳述式成功，則會傳回每`id`個插入文件的。

3. 若要執行程式，請輸入下列命令。

```
python insert_document.py
```

接下來，您可以使用`SELECT`語句從`vehicle-registration`分類帳中的表中讀取數據。繼續執行「步驟 4：查詢分類帳中的表格」。

步驟 4：查詢分類帳中的表格

在 Amazon QLDB 分類帳中建立表格並將其載入資料後，您可以執行查詢以檢閱剛插入的車輛註冊資料。QLDB 使用 PartiQL 作為其查詢語言，而亞馬遜離子作為其文件導向的資料模型。

PartiQL 是一種開放原始碼、SQL 相容的查詢語言，已經擴充為與 Ion 搭配使用。使用 PartiQL，您可以使用熟悉的 SQL 運算子插入、查詢和管理資料。亞馬遜離子是 JSON 的超集合。Ion 是一種開放原始碼、以文件為基礎的資料格式，可讓您靈活地儲存和處理結構化、半結構化和巢狀資料。

在此步驟中，您可以使用`SELECT`陳述式從`vehicle-registration`分類帳中的表格讀取資料。

> **Warning**
>
> 當您在沒有索引查閱的情況下在 QLDB 中執行查詢時，它會叫用完整資料表掃描。PartiQL 支持這樣的查詢，因為它是 SQL 兼容。不過，請勿在 QLDB 中針對生產使用案例執行資料表掃描。資料表掃描可能會造成大型資料表的效能問題，包括並行衝突和交易逾時。
> 若要避免資料表掃描，您必須在索引欄位或文件 ID 上使用相等運算子來執行具有述`WHERE`詞子句的陳述式；例如，`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`。如需詳細資訊，請參閱最佳化查詢效能。

若要查詢資料表

1. 使用以下程序（find_vehicles.py）查詢在分類帳中以人員註冊的所有車輛。

   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**
>
> 首先，這個程Person序查詢與GovId LEWISR261LL獲取其id元數據字段的文檔表。然後，它使用此文檔id作為外鍵來查詢VehicleRegistration表PrimaryOwner.PersonId。它也會VehicleRegistration與VIN欄位上的Vehicle表格連接在一起。

2. 若要執行程式，請輸入下列命令。

```
python find_vehicles.py
```

若要瞭解如何在vehicle-registration分類帳中修改表格中的文件，請參閱步驟 5：修改分類帳中的文件。

步驟 5：修改分類帳中的文件

既然您有資料需要使用，就可以開始變更 Amazon QLDB 中vehicle-registration分類帳中的文件。在此步驟中，下列程式碼範例會示範如何執行資料操作語言 (DML) 陳述式。這些聲明更新了一輛車的主要車主，並將次要車主添加到另一輛車。

若要修改文件

1. 使用下列程式 (transfer_vehicle_ownership.py)，以分類帳1N4AL11D75C109151中的 VIN 更新車輛的主要擁有者。

   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. 若要執行程式，請輸入下列命令。

```
python transfer_vehicle_ownership.py
```

3. 使用下列程式 (add_secondary_owner.py) 將次要擁有者新增至分類帳中含有 VKM8SRDHF6EU074761 IN 的車輛。

   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. 若要執行程式，請輸入下列命令。

```
python add_secondary_owner.py
```

若要複查vehicle-registration分類帳中的這些變更，請參閱步驟 6：檢視文件的修訂歷程記錄。

步驟 6：檢視文件的修訂歷程記錄

在上一個步驟中修改車輛的註冊資料後，您可以查詢其所有註冊車主的歷史記錄以及任何其他更新的欄位。在此步驟中，您會查詢vehicle-registration分類帳VehicleRegistration表格中文件的修訂歷史記錄。

若要檢視修訂歷史記錄

1. 使用下面的程序（query_history.py）用 VIN 查詢VehicleRegistration文檔的修訂歷史記錄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**
>
> - 您可以使用下列語法查詢內建的文件，以檢視文件歷史功能的修訂歷程記錄。
>
>   ```
>   SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
>   [ WHERE h.metadata.id = 'id' ]
>   ```
>
> - 開始時間和結束時間都是可選的。它們是亞馬遜離子字面值，可以用反引號 (`...`) 表示。如需進一步了解，請參閱 在亞馬遜 QLDB 中使用 PartiQL 查詢離子。
> - 最佳作法是使用日期範圍 (開始時間和結束時間) 和文件 ID (`metadata.id`) 來限定歷史查詢。QLDB 處理交易中的`SELECT`查詢，這些交易受到交易逾時限制。
>
>   QLDB 歷程記錄會依文件 ID 編製索引，而且您目前無法建立其他歷程記錄索引。包含開始時間和結束時間的歷史記錄查詢可獲得日期範圍限定的好處。

2. 若要執行程式，請輸入下列命令。

```
python query_history.py
```

若要以密碼編譯方式驗證`vehicle-registration`分類帳中的文件版次，請繼續執行步驟 7：驗證分類帳中的文件。

步驟 7：驗證分類帳中的文件

使用 Amazon QLDB，您可以使用搭配 SHA-256 的加密雜湊，有效地驗證分類帳日誌中文件的完整性。若要深入瞭解驗證和加密雜湊如何在 QLDB 中運作，請參閱Amazon QLDB 中的數據驗證。

在此步驟中，您會核對`vehicle-registration`分類帳中`VehicleRegistration`表格中的文件修訂。首先，您要求摘要，該摘要會以輸出檔案的形式傳回，並做為分類帳整個變更歷史記錄的簽章。然後，您要求相對於該摘要的修訂版本的證明。使用此證明，如果所有驗證檢查都通過，則會驗證修訂的完整性。

核對文件修訂

1. 檢閱下列`.py`檔案，這些檔案代表需要驗證的 QLDB 物件，以及具有協助程式函數的公用程式模組，將 QLDB 回應類型轉換為字串。

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. 使用兩個.py程式 (get_digest.py和get_revision.py) 來執行下列步驟：

   - 請求分vehicle-registration類帳中的新摘要。
   - 請1N4AL11D75C109151從表格中使用 VIN 為文檔的每個修訂版提供證VehicleRegistration明。
   - 透過重新計算摘要，使用傳回的摘要和校樣來驗證修訂。

   該get_digest.py程序包含以下代碼。

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# 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**
>
> 使用此get_digest_result函數可要求摘要，其中涵蓋分類帳中分類帳目前提示的摘要。日誌的提示是指 QLDB 收到您的請求時最新的已提交區塊。

該get_revision.py程序包含以下代碼。

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**
>
> 在get_revision函數傳回指定文件修訂的證明之後，此程式會使用用戶端 API 來驗證該修訂版本。

3. 若要執行程式，請輸入下列命令。

```
python get_revision.py
```

如果您不再需要使用vehicle-registration分類帳，請繼續執行步驟 8 (選擇性)：清除資源。

步驟 8 (選擇性)：清除資源

您可以繼續使用分vehicle-registration類帳。但是，如果您不再需要它，應將其刪除。

若要刪除分類帳

1. 使用以下程序（delete_ledger.py）刪除vehicle-registration分類帳及其所有內容。

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
```

```
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# 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**
>
> 如果已對分類帳已啟用刪除保護，您必須先停用它，然後才能使用 QLDB API 刪除總帳。

該delete_ledger.py文件還具有對下面的程序（describe_ledger.py）的依賴。

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# 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. 若要執行程式，請輸入下列命令。

```
python delete_ledger.py
```

在亞馬遜 QLDB 中使用亞馬遜離子數據類型

亞馬遜 QLDB 以亞馬遜離子格式存儲其數據。若要使用 QLDB 中的資料，您必須使用 Ion 程式庫做為支援程式設計語言的相依性。

在本節中，了解如何將數據從本地類型轉換為其 Ion 等效物，以及相反的方法。本參考指南顯示使用 QLDB 驅動程式來處理 QLDB 分類帳中的離子資料的程式碼範例。它包括 Java、.NET (C#) Go、Node.js (TypeScript) 和 Python。

主題

先決條件

下列程式碼範例假設您有一個 QLDB 驅動程式執行個體，該執行個體連線至具有名為的表格的作用中分類帳`ExampleTable`。此表格包含具有下列八個欄位的單一現有文件：

- ExampleBool
- ExampleInt
- ExampleFloat

- ExampleDecimal
- ExampleTimestamp
- ExampleString
- ExampleBlob
- ExampleList

> **Note**
>
> 基於此參考的目的，假設每個欄位中儲存的類型與其名稱相符。實際上，QLDB 不會強制執行文件欄位的結構描述或資料類型定義。

Bool

下列程式碼範例示範如何處理 Ion Boolean 類型。

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**
>
> 若要轉換為同步程式碼，請移除await和async關鍵字，然後將IAsyncResult類型變更為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

下列程式碼範例示範如何處理 Ion Inon 整數類型。

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**
>
> 若要轉換為同步程式碼，請移除await和async關鍵字，然後將IAsyncResult類型變更為IResult。

Go

```
exampleInt, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aInt := 256

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleInt = ?", aInt)
 if err != nil {
  return nil, err
 }

 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleInt FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult int
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleInt is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonInt(driver: QldbDriver): Promise<number> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleInt = ?", 256);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleInt FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Transforming Ion to a TypeScript Number
            const intValue: number = ionValue.numberValue();
            return intValue;
        })
    );
}
```

Python

```
def update_and_query_ion_int(txn):
    # QLDB can take in a Python int
    a_int = 256

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleInt = ?", a_int)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleInt FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python int is a child class of Python int
        a_int = ion_value

    # example_int is now the value fetched from QLDB
    return a_int


example_int = driver.execute_lambda(lambda txn: update_and_query_ion_int(txn))
```

Float

下列程式碼範例示範如何處理 Ion float 類型。

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

使用 C# 浮動

```
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**
>
> 若要轉換為同步程式碼，請移除await和async關鍵字，然後將IAsyncResult類型變更為IResult。

使用 C# 雙

```
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**
>
> 若要轉換為同步程式碼，請移除await和async關鍵字，然後將IAsyncResult類型變更為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 (小數)

下列程式碼範例示範如何處理 Ion (Ion) 小數位數。

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**
>
> 若要轉換為同步程式碼，請移除await和async關鍵字，然後將IAsyncResult類型變更為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**
>
> 您也可以使ionValue.numberValue()用 Ion 十進位數轉換為 JavaScript 數字。使用ionValue.numberValue()具有更好的性能，但不太精確。

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))
```

時間戳記

下列程式碼範例示範如何處理 Ion Ion 時間戳記。

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**
>
> 若要轉換為同步程式碼，請移除await和async關鍵字，然後將IAsyncResult類型變更為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))
```

字串

下列程式碼範例示範如何處理 Ion (Ion) 字串類型。

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**
>
> 若要轉換為同步程式碼，請移除await和async關鍵字，然後將IAsyncResult類型變更為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

下列程式碼範例示範如何處理 Ion Blob。

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

byte[] exampleBytes = driver.execute((txn) -> {
    // Transforming a Java byte array to Ion
    // Transform any arbitrary data to a byte array to store in QLDB
    String aString = "Hello world!";
    byte[] aByteArray = aString.getBytes();
    IonValue ionBlob = ionSystem.newBlob(aByteArray);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleBlob = ?", ionBlob);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleBlob from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java byte array
        // Cast IonValue to IonBlob first
        aByteArray = ((IonBlob)ionValue).getBytes();
    }

    // exampleBytes is now the value fetched from QLDB
    return aByteArray;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
using System.Text;
...

IValueFactory valueFactory = new ValueFactory();

// Transform any arbitrary data to a byte array to store in QLDB.
string nativeString = "Hello world!";
```

```
byte[] nativeByteArray = Encoding.UTF8.GetBytes(nativeString);
// Transforming a C# byte array to Ion.
IIonValue ionBlob = valueFactory.NewBlob(nativeByteArray);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleBlob = ?", ionBlob);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleBlob from ExampleTable");
});

byte[] retrievedByteArray = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# byte array.
    retrievedByteArray = ionValue.Bytes().ToArray();
}
```

> **Note**
>
> 若要轉換為同步程式碼，請移除`await`和`async`關鍵字，然後將`IAsyncResult`類型變更為`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))
```

列出

下列程式碼範例示範如何處理 Ion List 類型。

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**
>
> 若要轉換為同步程式碼，請移除await和async關鍵字，然後將IAsyncResult類型變更為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

在 QLDB 中，`struct`數據類型與其他離子類型特別獨特。插入到表格中的頂層文件必須是該`struct`類型。文檔字段也可以存儲嵌套`struct`。

為了簡化起見，下列範例會定義僅`ExampleInt`包含`ExampleString`和欄位的文件。

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;
});
```

或者，您可以使用傑克遜庫將數據類型映射到 Ion 和從 Ion 映射。該庫支持其他 Ion 數據類型，但這個例子側重於類`struct`型。

```
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

使用亞馬遜 .qldb.driver. 序列化庫將本機 C # 數據類型映射到離子和離子。該庫支持其他 Ion 數據類型，但這個例子側重於類`struct`型。

```
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**
>
> 若要轉換為同步程式碼，請移除await和async關鍵字，然後將IAsyncResult類型變更為IResult。

或者，您可以使用 Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon IonDotnet. 用於處理離子數據類型的構建器庫。

```
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))
```

空值和動態類型

QLDB 支援開放式內容，不會強制執行文件欄位的結構描述或資料類型定義。您也可以將離子空值存儲在 QLDB 文檔中。所有先前的範例都假設每個傳回的資料類型都是已知且不為 null。下列範例示範如何配合使用 Ion，當資料類型不知道或可能為 null 時，如何配合使用 Ion。

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

編組限制

在 Go 中，當您編組然後解組它們時，nil值不會保留它們的類型。一個nil值編組為離子空，但離子空解組為零值而不是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!
```

向下轉換為 JSON

如果您的應用程式需要 JSON 相容性，您可以將 Amazon 離子資料下載轉換為 JSON。但是，在某些情況下，您的數據使用 JSON 中不存在的豐富 Ion 類型的情況下，將 Ion 轉換為 JSON 是有損的。

有關離子到 JSON 轉換規則的詳細信息，請參閱亞馬遜離子食譜中的向下轉換為 JSON。

使用亞馬遜 QLDB 中的數據和歷史記錄

下列主題提供建立、更新及刪除 (CRUD) 陳述式的基本範例。您可以使用 QLDB 主控台或 QLDB 殼層上的 PartiQL 編輯器來手動執行這些陳述式。本指南也會引導您完成 QLDB 如何在分類帳中進行變更時處理資料的程序。

QLDB 支援的 PartiQL 語言。

如需示範如何使用 QLDB 驅動程式以程式設計方式執行類似陳述式的程式碼範例，請參閱開始使用驅動程式.

> **Tip**
>
> 以下是在 QLDB 中使用 PartiQL 的秘訣和最佳作法的簡短摘要：
>
> - 瞭解並行與交易限制 — 包括SELECT查詢在內的所有陳述式都受到樂觀的並行控制 (OCC) 衝突和交易限制，包括 30 秒的交易逾時。
> - 使用索引 — 使用高基數索引並執行目標查詢，以最佳化陳述式並避免完整表格掃描。如需進一步了解，請參閱 最佳化查詢效能。
> - 使用相等述詞 — 索引查詢需要相等運算子 (=或IN)。不等式運算子 (<、>LIKE、BETWEEN) 不符合索引查詢的資格，因此會產生完整的資料表掃描。
> - 僅使用內部聯結 — QLDB 僅支援內部聯結。最佳做法是聯結您要加入的每個資料表索引的欄位。為聯結準則和相等述詞選擇高基數索引。

主題

- 取得 PartiQL 陳述式統計
- 查詢系統目錄
- 管理資料表
- 管理索引
- 亞馬遜 QLDB 中的唯一 ID

使用索引建立表格並插入文件

建立 Amazon QLDB 分類帳之後，您的第一個步驟是建立包含基本CREATE TABLE陳述式的資料表。表包括QLDB 文件，這是亞馬遜離子struct格式的數據集。

主題

- 建立資料表及索引
- 插入文件

建立資料表及索引

表具有簡單，區分大小寫的名稱，沒有命名空間。QLDB 支援開放式內容，不會強制執行結構定義，因此您不會在建立資料表時定義屬性或資料類型。

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle
```

CREATE TABLE陳述式會傳回新資料表的系統指派 ID。QLDB 中所有系統指派的 ID 都是通用唯一識別碼 (UUID)，每個識別碼都以 Base62 編碼的字串表示。

> **Note**
>
> 或者，您可以在創建表格時定義表格資源的標籤。如要瞭解如何作業，請參閱 建立時標記資料表。

您也可以在資料表上建立索引來最佳化查詢效能。

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

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

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

> **Important**
>
> QLDB 需要一個索引來有效地查找文檔。如果沒有索引，QLDB 需要在讀取文檔時進行全表掃描。這可能會造成大型資料表的效能問題，包括並行衝突和交易逾時。
> 若要避免資料表掃描，您必須在索引欄位或文件 ID 上使用相等運算子 (=或IN) 執行具有述WHERE詞子句的陳述式。如需詳細資訊，請參閱最佳化查詢效能。

建立索引時，請注意下列條件約束：

- 索引只能在單一頂層欄位上建立。不支援複合、巢狀、唯一和以函數為基礎的索引。
- 您可以在任何 Ion 資料類型上建立索引，包括`list`和`struct`。但是，無論 Ion 類型如何，您都只能通過整個 Ion 值的相等性來進行索引查找。例如，當使用`list`類型作為索引時，您不能通過列表中的一個項目進行索引查找。
- 只有在使用相等述詞時才會改善查詢效能；例如，`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`。

  QLDB 不尊重查詢謂詞中的不等式。因此，範圍篩選的掃描不會實作。
- 索引欄位的名稱區分大小寫，且最大長度可為 128 個字元。
- QLDB 中的索引建立是非同步的。完成建立非空白資料表索引所需的時間量因資料表大小而異。如需詳細資訊，請參閱管理索引。

插入文件

然後，您可以將文檔插入到表格中。QLDB 文件以亞馬遜離子格式存儲。下列 PartiQLINSERT 陳述式包含中使用的車輛登記範例資料的子集Amazon QLDB 主控台。

```
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'
} >>
```

PartiQL 和語義

- 欄位名稱會以單引號括住 (`'...'`)。
- 字串值也會以單引號 (`'...'`) 括住。
- 時間戳記包含在反引號 (`` `...` ``) 中。反引號可用於表示任何離子文字。
- 整數和小數是不需要被表示的文字值。

如需 PartiQL 語法和語意的詳細資訊，請參閱在亞馬遜 QLDB 中使用 PartiQL 查詢離子。

`INSERT`陳述式會建立版本編號為零的文件的初始版本修訂。為了唯一標識每個文檔，QLDB 分配一個文檔 ID 作為元數據的一部分。插入陳述式會傳回每個插入文件的識別碼。

> **Important**
>
> 由於 QLDB 不會強制執行結構定義，因此您可以多次將相同的文件插入資料表中。每個 insert 陳述式都會提交個別的文件項目給分錄，而 QLDB 會為每個文件指派唯一的識別碼。

若要瞭解如何查詢已插入表格的文件，請繼續執行查詢資料。

查詢資料

用戶視圖僅返回用戶數據的最新未刪除版本。這是亞馬遜 QLDB 中的默認視圖。這表示當您只想查詢資料時，不需要特殊限定元。

如需下列查詢範例語法和參數的詳細資訊，請參閱 Amazon QLDB PartiQL 參考資料`SELECT`中的。

主題

- 基本查詢
- 投影和濾波器
- 聯結
- 巢狀資料

基本查詢

基本`SELECT`查詢會傳回您插入到表格中的文件。

> **Warning**
>
> 當您在沒有索引查閱的情況下在 QLDB 中執行查詢時，它會叫用完整資料表掃描。PartiQL 支持這樣的查詢，因為它是 SQL 兼容。不過，請勿在 QLDB 中針對生產使用案例執行資料表掃描。資料表掃描可能會造成大型資料表的效能問題，包括並行衝突和交易逾時。
>
> 若要避免資料表掃描，您必須在索引欄位或文件 ID 上使用相等運算子來執行具有述WHERE詞子句的陳述式；例如，`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`。如需詳細資訊，請參閱最佳化查詢效能。

下列查詢會顯示您先前插入的車輛登記文件的結果使用索引建立表格並插入文件。結果的順序不是特定的，並且每個SELECT查詢可能會有所不同。您不應該依賴 QLDB 中任何查詢的結果順序。

```
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

在 PartiQL 中，您可以使用單引號來表示資料處理語言 (DML) 或查詢陳述式中的字串。但是 QLDB 主控台和 QLDB 殼層會以 Amazon Ion 文字格式傳回查詢結果，因此您會看到以雙引號括住的字串。

此語法可讓 PartiQL 查詢語言維護 SQL 相容性，而 Amazon Ion 文字格式則可維持 JSON 相容性。

投影和濾波器

您可以進行預測（目標SELECT）和其他標準過濾器（WHERE子句）。下面的查詢返回從VehicleRegistration表中的文檔字段的子集。它會針對具有下列條件的車輛進行篩選：

- 字符串過濾器-它在西雅圖註冊。
- 十進制過濾器 — 它的待處罰單金額小於100.0。
- 日期篩選器 — 註冊日期在 2019 年 9 月 4 日或之後有效。

```
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" }]
    }
}
```

聯結

您也可以編寫內部聯接查詢。以下示例顯示了一個隱含的內部聯接查詢，該查詢返回所有註冊文件以及已註冊車輛的屬性。

```
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"
}
```

或者，您可以在顯式語法中編寫相同的內部連接查詢，如下所示。

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

巢狀資料

您可以在 QLDB 中使用 PartiQL 來查詢文件中的巢狀資料。下列範例顯示扁平化巢狀資料的相關子查詢。在這裡，@字符在技術上是可選的。但它明確表明你想要結Owners構VehicleRegistration，而不是名為的不同集合Owners（如果存在的話）。

```
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: []
}
```

以下顯示SELECT清單中的子查詢，該子查詢專案巢狀資料 (除了內部關連外)。

```
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"]
}
```

下Owners.SecondaryOwners列查詢會傳回VehicleRegistration文件清單中每個人員的PersonId與索引 (序數) 編號。

```
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
}
```

若要瞭解如何查詢文件中繼資料，請繼續執行查詢文件元資料。

查詢文件元資料

INSERT陳述式會建立版本編號為零的文件的初始版本修訂。為了唯一識別每個文件，Amazon QLDB 會指派一個文件 ID 做為中繼資料的一部分。

除了文件 ID 和版本號碼之外，QLDB 會將每個文件的其他系統產生的中繼資料儲存在表格中。此中繼資料包括交易資訊、日誌屬性和文件的雜湊值。

所有系統指派的 ID 都是通用唯一識別碼 (UUID)，每個識別碼都以 Base62 編碼的字串表示。如需詳細資訊，請參閱亞馬遜 QLDB 中的唯一 ID。

主題

- 已提交檢視
- 加入提交和用戶視圖

已提交檢視

您可以透過查詢已提交的檢視來存取文件中繼資料。此檢視會從系統定義的資料表傳回直接對應至您的使用者資料表的文件。它包括資料和系統產生的中繼資料的最新提交、未刪除的修訂版本。若要查詢此檢視，請在查詢中將前置字元新增_ql_committed_至資料表名稱。（系統對象的前綴_ql_保留在 QLDB 中。）

```
SELECT * FROM _ql_committed_VehicleRegistration AS r
WHERE r.data.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

使用先前插入的資料使用索引建立表格並插入文件，此查詢的輸出會顯示每個未刪除文件的最新修訂版本的系統內容。系統文件的中繼資料巢狀於metadata欄位中，而您的使用者資料則以巢狀方式嵌套在data欄位中。

```
{
    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"
    }
```

```
}
```

已提交檢視欄位

- blockAddress— 分類帳期刊中已確認文件修訂的區塊位置。可用於密碼編譯驗證的位址具有以下兩個欄位。
  - strandId— 包含區塊之日誌鏈的唯一 ID。
  - sequenceNo— 指定圖塊在鏈中的位置的索引號碼。

> **Note**
>
> 此示例中的兩個文檔具有blockAddress相同的相同sequenceNo。因為這些文件是在單一交易中插入 (在這種情況下，在單一陳述式中)，所以它們是在同一個區塊中提交。

- hash— 唯一代表文件修訂版本的 SHA-256 離子雜湊值。雜湊涵蓋修訂版本data和metadata欄位，可用於密碼編譯驗證。
- data— 文件的使用者資料屬性。

  如果您編輯修訂版本，則此data結構會由dataHash欄位取代，其值為已移除data結構的 Ion 雜湊值。
- metadata— 文件的中繼資料屬性。
  - id— 系統指派的文件唯一 ID。
  - version— 文件的版本號碼。這是一個從零開始的整數，會隨著每個文件修訂版遞增。
  - txTime— 將文件修訂送至日誌時的時間戳記。
  - txId— 認可文件修訂之交易的唯一 ID。

加入提交和用戶視圖

您可以撰寫查詢，將已提交檢視中的資料表與使用者檢視中的資料表聯結在一起。例如，您可能想要將一個資料表id的文件與另一個資料表的使用者定義欄位聯結。

下列查詢會分別連接兩個名為DriversLicense ANDPerson 的資料表PersonId和文件id欄位，並使用後者的認可檢視表。

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS p
ON d.PersonId = p.metadata.id
```

```
WHERE p.metadata.id = '1CWScY2qHYI9G88C2SjvtH'
```

若要瞭解如何在預設使用者檢視中查詢文件 ID 欄位，請繼續執行使用 BY 子句來查詢文件 ID。

使用 BY 子句來查詢文件 ID

雖然您可以定義要做為唯一識別碼的欄位 (例如，車輛的 VIN)，但文件的真正唯一識別碼就是中id繼資料欄位，如中所述插入文件。因此，您可以使用id欄位來建立資料表之間的關聯性。

文件id欄位只能在已提交的檢視中直接存取，但您也可以使用BY子句在預設使用者檢視中進行投影。如需範例，請參閱下列查詢及其結果。

```
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" }]
    }
}
```

在此查詢中，r_id是使用BY關鍵字在FROM子句中宣告的使用者定義別名。此r_id別名會繫結至查詢結果集中每個文件的中id繼資料欄位。您可以在SELECT子句中使用此別名，也可以在使用者檢視中的查詢WHERE子句中使用此別名。

但是，若要存取其他中繼資料屬性，您必須查詢已提交的檢視表。

在文件 ID 上加入

假設您使用一個表id的文檔作為另一個表的用戶定義字段中的外鍵。您可以使用BY子句為這些欄位上的兩個資料表撰寫內部聯結查詢 (類似加入提交和用戶視圖前一個主題)。

下列範例會使用後者的BY子句，分別連接兩個名DriversLicense為 AND 的表格PersonId和文件id欄位。Person

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
ON d.PersonId = pid
WHERE pid = '1CWScY2qHYI9G88C2SjvtH'
```

若要瞭解如何變更表格中的文件，請繼續執行更新和刪除文件。

更新和刪除文件

在 Amazon QLDB 中，文件修訂是 Amazon Ion 結構，代表以唯一文件 ID 識別的一系列文件的單一版本。每個修訂版都包含文件的完整資料集，包括使用者資料和系統產生的中繼資料。每個版本修訂都是由文件 ID 和從零開始的版本編號的組合來唯一識別。

當您更新文件時，QLDB 會建立具有相同文件 ID 和遞增版本號碼的新修訂版本。當您從表格中刪除文件時，文件的生命週期就會結束。這表示無法再次建立具有相同文件 ID 的文件修訂版本。

修訂文件

例如，下列陳述式會插入新的車輛登記、更新註冊城市，然後刪除註冊。這會導致文件的三個修訂版本。

```
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**
>
> Insert 陳述式和其他 DML 陳述式會傳回每個受影響文件的識別碼。繼續之前，請先儲存此 ID，因為下一個主題中的歷程記錄功能需要此 ID。您也可以使用尋找文件 ID。

```
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'
```

如需有關這些 DML 陳述式語法的詳細範例UPDATE和資訊，請參閱 Amazon QLDB PartiQL 參考資料DELETE中的和。

若要插入和移除文件中的特定元素，您可以使用UPDATE陳述式或其他以FROM關鍵字開頭的 DML 陳述式。如需詳細資訊和範例，請參閱參從 (插入、移除或設定)考資料。

刪除文件後，您將無法再在已提交或使用者檢視中查詢該文件。若要瞭解如何使用內建的歷程記錄功能查詢此文件的修訂歷程記錄，請繼續執行查詢修訂記錄。

查詢修訂記錄

Amazon QLDB 會將每個文件的完整歷史記錄儲存在表格中。您可以查詢內建的歷史記錄功能，查看先前插入、更新和刪除的車輛登記文件更新和刪除文件的全部三個修訂版本。

主題

- 歷史功能
- 歷史查詢範例

歷史功能

QLDB 中的歷史記錄函數是 PartiQL 擴充功能，可從資料表的系統定義檢視中傳回修訂版本。因此，它將您的數據和關聯的元數據包含在與提交視圖相同的模式中。

Syntax (語法)

```
SELECT * FROM history( table_name | 'table_id' [, `start-time` [, `end-time` ] ] ) AS h
[ WHERE h.metadata.id = 'id' ]
```

引數

####|「*## ID*」

資料表名稱或資料表 ID。資料表名稱是 PartiQL 識別碼，您可以使用雙引號或不加引號來表示。資料表 ID 是字串常值，而且必須以一對單引號括住。若要進一步了解如何使用資料表 ID，請參閱查詢非作用中表格的歷史記錄。

`*###*間` , `*####*`

(選擇性) 指定任何修訂作用中的時間範圍。這些參數不會指定修訂確認交易中分錄的時間範圍。

開始和結束時間是離子時間戳文字，可以用反引號 (`` `...` ``) 表示。如需進一步了解，請參閱 在亞馬遜 QLDB 中使用 PartiQL 查詢離子。

這些時間參數具有下列行為：

- 開始時間和結束時間都包括在內。該值必須採用 ISO 8601 日期和時間格式，並以國際標準時間 (UTC) 表示。
- 開始時間必須小於或等於結束時間，並且可以是過去的任意日期。
- 結束時間必須小於或等於目前的 UTC 日期和時間。
- 如果您指定開始時間而非結束時間，則查詢會將結束時間預設為目前的日期和時間。如果兩者都不指定，則查詢會傳回整個歷史記錄。

'*###*'

(選擇性) 您要查詢其修訂歷程記錄的文件 ID，以單引號表示。

> **Tip**
>
> 最佳作法是使用日期範圍 (開始時間和結束時間) 和文件 ID (`metadata.id`) 來限定歷史查詢。在 QLDB 中，每個`SELECT`查詢都會在交易中處理，並受到交易逾時限制的限制。
> 歷程查詢不會使用您在資料表上建立的索引。QLDB 歷史記錄僅由文件 ID 編製索引，而且您目前無法建立其他歷史記錄索引。包含開始時間和結束時間的歷史查詢可獲得日期範圍限定的好處。

歷史查詢範例

若要查詢車輛登記文件的歷史記錄，請使用您先前儲存的更新和刪除文件。id例如，下列歷史記錄查詢會傳回2019-06-05T00:00:00Z與之間曾經使用過ADR2Ll1fGsU4Jr4EqTdnQF的文件 ID 的任何修訂2019-06-05T23:59:59Z。

> **Note**
>
> 請記住，開始時間和結束時間參數不會指定修訂確認至交易中分錄的時間範圍。例如，如果修訂版本在該開始時間之前已確認，2019-06-05T00:00:00Z且在該開始時間之前仍保持作用中狀態，則此範例查詢會在結果中傳回該修訂。

請務必視情況以id您自己的值取代、開始時間和結束時間。

```
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
```

您的查詢結果看起來應如下所示。

```
{
    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"
    }
```

```
}
```

輸出包含中繼資料屬性，這些屬性提供了每個項目修改時間以及何種交易的詳細資訊。從這些資料中，您可以看到下列內容：

- 文件由其系統指定的`id`:唯一識別`ADR2Ll1fGsU4Jr4EqTdnQF`。這是以 Base62 編碼字串表示的 UUID。
- `INSERT`陳述式會建立文件 (版本`0`) 的初始修訂版本。
- 每次後續更新都會建立具有相同文件`id`和遞增版本號碼的新修訂版本。
- 此`txId`欄位會指出認可每個修訂的作業事件，並`txTime`顯示每個修訂的確認時間。
- `DELETE`陳述式會建立文件的新版本，但最終的修訂版本。此最終修訂版僅包含中繼資料。

若要瞭解如何永久刪除修訂，請繼續執行編輯文件修訂版本。

編輯文件修訂版本

在 Amazon QLDB 中，`DELETE`陳述式只會以邏輯方式刪除文件，方法是建立將文件標記為已刪除的新修訂。QLDB 也支援資料編輯作業，可讓您永久刪除表格歷程記錄中非使用中的文件修訂版本。

> **Note**
>
> 在 2021 年 7 月 22 日之前建立的任何分類帳目前都不符合編輯資格。您可以在 Amazon QLDB 主控台上檢視分類帳的建立時間。

密文作業只會刪除指定修訂版本中的使用者資料，並保持分錄序列與文件中繼資料不變。這會維護分類帳的整體資料完整性。

在開始使用 QLDB 中的資料編輯之前，請確定您已密文考量與限制在 Amazon QLDB PartiQL 參考資料中進行檢閱。

主題

- 密文預存程序
- 檢查密文是否完成
- 密文範例
- 刪除和標記使用中的版本修訂

- 標記版本中的特定欄位

密文預存程序

您可以使用修訂版預存程序，永久刪除分類帳中個別失效的版次。此預存程序會刪除索引儲存體和日誌儲存中指定修訂版本中的所有使用者資料。不過，它會保持日誌序列和文件中繼資料 (包括文件 ID 和雜湊) 不變。此操作是不可逆轉的。

指定的文件修訂版本必須是記錄中的非使用中修訂版本。文件的最新使用中修訂版本不符合密文的資格。

若要編輯多個修訂版本，您必須針對每個修訂執行一次預存程序。您可以為每個異動編輯一個修訂。

Syntax (語法)

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

引數

`####`

要編輯之文件修訂版本的分錄區塊位置。地址是具有兩個字段的 Amazon 離子結構：strandId和sequenceNo。

這是由反引號表示的離子文字值。例如：

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

'#####'

您要編輯其文件修訂版本之表格的唯一 ID，以單引號表示。

「#####」

要編輯之修訂的唯一文件 ID，以單引號表示。

檢查密文是否完成

當您執行預存程序來提交密文要求時，QLDB 會以非同步方式處理資料的編輯。完成後，修訂版本中的使用者資料 (以結data構表示) 會永久移除。若要檢查密文請求是否已完成，您可以使用下列其中一項：

- 分錄匯出
- 期刊串流
- GetBlock API 操作
- GetRevision API 操作
- 歷史功能— 附註：在日誌中完成密文後，歷史記錄查詢可能需要一些時間才能顯示密文結果。當非同步編輯完成時，您可能會看到某些修訂版本在其他修訂版之前進行編輯，但歷程查詢最終會顯示完成的結果。

修訂版本密文完成後，版本修訂的data結構會被新dataHash欄位取代。此欄位的值是已移除data結構的離子雜湊值，如下列範例所示。因此，分類帳會維護其整體資料完整性，並透過現有的驗證 API 作業維持密碼編譯驗證。要進一步了解驗證，請參閱Amazon QLDB 中的數據驗證。

密文範例

考慮您之前在中審閱的車輛登記文件查詢修訂記錄。假設您想要密文第二個修訂版本 (version:1)。下列查詢範例會在密文前顯示此修訂。在查詢結果中，要編輯的data結構會以###體反白顯示。

```
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"
    }
}
```

請注意查詢結果blockAddress中的，因為您需要將此值傳遞給REDACT_REVISION預存程序。然後，透過查詢系統目錄來尋找資VehicleRegistration料表的唯一 ID，如下所示。

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

使用此表格 ID 以及要執行的文件 ID 和區塊位址REDACT_REVISION。資料表 ID 和文件 ID 是字串常值，必須以單引號括住，而區塊位址是以反引號括住的 Ion 常值。請務必視情況使用您自己的值取代這些引數。

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

> **Tip**
>
> 當您使用 QLDB 主控台或 QLDB 殼層查詢資料表識別碼或文件識別碼 (或任何字串常值) 時，傳回的值會以雙引號括住。不過，當您指定REDACT_REVISION預存程序的資料表 ID 和文件 ID 引數時，必須以單引號括住這些值。
>
> 這是因為您以 PartiQL 格式撰寫陳述式，但 QLDB 會以亞馬遜離子格式傳回結果。如需有關 QLDB 中 PartiQL 語法和語意的詳細資訊，請參閱使用 PartiQL 查詢離子。

有效的密文請求會傳回代表您編輯的文件修訂版本的 Ion 結構，如下所示。

```
{
  blockAddress: {
    strandId: "JdxjkR9bSYB5jMHWcI464T",
    sequenceNo: 17
  },
  tableId: "5PLf9SXwndd63lPaSIa0O6",
```

```
  documentId: "ADR2Ll1fGsU4Jr4EqTdnQF",
  version: 1
}
```

當您執行這個預存程序時，QLDB 會以非同步方式處理您的密文要求。完成密文後，結data構會永久移除，並由新*dataHash*欄位取代。此欄位的值是已移除data結構的離子雜湊值，如下所示。

> **Note**
>
> 此dataHash範例僅供參考，並非真正的計算雜湊值。

```
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:17
    },
    hash:{{LGSFZ4iEYWZeMwmAqcxxNyT4wbCtuMOmFCj8pEd6Mp0=}},
    dataHash: {{s83jd7sfhsdfhksj7hskjdfjfpIPP/DP2hvionas2d4=}},
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:1,
        txTime:2019-06-05T21:01:442d-3Z,
        txId:"9cArhIQV5xf5Tf5vtsPwPq"
    }
}
```

刪除和標記使用中的版本修訂

使用中的文件修訂版 (也就是每個文件的最新未刪除修訂) 不符合資料密文的資格。您必須先更新或刪除它，才能密文使用中的修訂版本。這會將先前使用中的修訂版移至記錄，並使其符合密文的資格。

如果您的使用案例需要將整份文件標示為已刪除，請先使用 DELETE 陳述式。例如，下面的語句在邏輯上刪除帶有 VIN 的VehicleRegistration文檔1HVBBAANXWH544237。

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

然後，如先前所述，在刪除之前密文先前的修訂版本。如有必要，您也可以個別標記任何先前的修訂版本。

如果您的使用案例要求文件保持使用中狀態，您必須先使用 UPDATE 或 F ROM 陳述式來隱藏或移除您要編輯的欄位。此程序如下節所述。

標記版本中的特定欄位

QLDB 不支援文件修訂版中特定欄位的編輯。若要這麼做，您可以先使用更新-移除或 FROM-移除陳述式，從修訂中移除現有欄位。例如，下列陳述式會使用的 VIN 從`VehicleRegistration`文件中移除`LicensePlateNumber`欄位`1HVBBAANXWH544237`。

```
UPDATE VehicleRegistration AS r
REMOVE r.LicensePlateNumber
WHERE r.VIN = '1HVBBAANXWH544237'
```

然後，如先前所述，在移除之前密文先前的修訂版。如果需要，您還可以單獨編輯包含此現在已刪除字段的任何先前修訂。

若要瞭解如何最佳化您的查詢，請繼續執行最佳化查詢效能。

最佳化查詢效能

Amazon QLDB 旨在滿足高效能線上交易處理 (OLTP) 工作負載需要達成效能線上交易處理 (OLTP) 工作負載 這表示 QLDB 已針對一組特定的查詢模式進行最佳化，即使它支援類似 SQL 的查詢功能。設計應用程式及其資料模型以使用這些查詢模式至關重要。否則，隨著資料表的成長，您將會遇到重大的效能問題，包括查詢延遲、交易逾時和並行衝突。

本節說明 QLDB 中的查詢條件約束，並針對這些限制提供撰寫最佳查詢的指引。

主題

- 交易逾時上限
- 并发冲突
- 最佳查詢模式
- 需要避免的查詢模式
- 監控效能

交易逾時上限

在 QLDB 中，每個 PartiQL 陳述式 (包括每個`SELECT`查詢) 都會在交易中處理，並受到交易逾時限制的限制。在認可之前，交易最多可以執行 30 秒。在此限制之後，QLDB 會拒絕在交易上完成的任何工

作，並捨棄執行交易的工作階段。此限制可透過啟動交易而不提交或取消工作階段來保護服務的用戶端不會洩漏工作階段。

并发冲突

QLDB 通過使用樂觀並發控制（OCC）實現並發控制。次優查詢也可能導致更多的 OCC 衝突。如需 OCC 的相關資訊，請參閱Amazon QLLDB 並行模型。

最佳查詢模式

最佳做法是，您應該執行含有WHERE述詞子句的陳述式，以篩選索引欄位或文件 ID。QLDB 需要在索引欄位上使用相等運算子 (=或IN)，才能有效率地查詢文件。

以下是使用者檢視中最佳化查詢模式的範例。

```
--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'
```

任何不遵循這些模式的查詢都會叫用完整資料表掃描。對於大型資料表或傳回大型結果集的查詢，資料表掃描可能會造成交易逾時。它們還可能導致 OCC 與競爭交易衝突。

高基數索引

我們建議您對包含高基數值的欄位進行索引。例如，資VehicleRegistration料表中的VIN和LicensePlateNumber欄位都是唯一的索引欄位。

避免索引低基數欄位，例如狀態碼、地址州或省和郵遞區號。如果您為這類欄位建立索引，您的查詢可能會產生較大的結果集，這些結果集較有可能導致交易逾時或造成意外的 OCC 衝突。

已提交檢視查詢

您在已提交檢視中執行的查詢遵循與使用者檢視查詢相同的最佳化準則。您在資料表上建立的索引也會用於已提交檢視中的查詢。

歷史函數查詢

歷史記錄函數查詢不使用您在表上創建的索引。QLDB 歷史記錄僅由文件 ID 編製索引，而且您目前無法建立其他歷史記錄索引。

最佳做法是使用日期範圍 (開始時間和結束時間) 和文件 ID (`metadata.id`) 來限定歷史記錄查詢。包含開始時間和結束時間的歷史查詢可獲得日期範圍限定的好處。

內部聯結查詢

對於內部聯結查詢，請使用聯結準則，其中至少包含聯結右側資料表的索引欄位。如果沒有聯結索引，聯結查詢會叫用多個資料表掃描 — 對於聯結左側資料表中的每個文件，查詢會完全掃描右側資料表。最佳做法是加入每個您要加入的資料表索引的欄位，以及為至少一個資料表指定`WHERE`相等述詞。

例如，下列查詢會聯結各自`VIN`欄位上的`VehicleRegistration`和`Vehicle`資料表，這兩個欄位都會建立索引。這個查詢也有一個相等謂詞`VehicleRegistration.VIN`。

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

為聯結準則和聯結查詢中的相等述詞選擇高基數索引。

需要避免的查詢模式

以下是一些次優陳述式的範例，這些陳述式對於 QLDB 中較大的資料表而言無法很好地擴充。我們強烈建議您不要依賴這些類型的查詢，因為您的查詢最終會導致交易逾時。由於表格包含大小不同的文件，因此很難定義非索引查詢的精確限制。

```
--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)
```

一般而言，我們不建議在 QLDB 中針對生產使用案例執行下列類型的查詢模式：

- 線上分析處理 (OLAP) 查詢
- 沒有謂詞子句的探索性查詢
- 報告查詢
- 文字搜尋

相反地，我們建議您將資料串流至針對分析使用案例進行最佳化的專用資料庫服務。例如，您可以將 QLDB 資料串流至亞馬遜 OpenSearch 服務，以便在文件上提供全文搜尋功能。如需示範此使用案例的範例應用程式，請參閱 GitHub 儲存庫 aws-Samples/amazon-qldb-streaming-amazon-opensearch-service-sample-python。如需 QLDB 串流的相關資訊，請參閱從 Amazon QLDB 串流日誌資料。

監控效能

QLDB 驅動程式會在陳述式的結果物件中提供耗用的 I/O 使用和計時資訊。您可以使用這些指標來識別效率低下的 PartiQL 陳述式。要了解更多信息，請繼續取得 PartiQL 陳述式統計。

您也可以使用 Amazon CloudWatch 追蹤分類帳的資料操作效能。監督`CommandLatency`指定的`LedgerName`和的測量結果`CommandType`。如需詳細資訊，請參閱使用 Amazon 監控 CloudWatch。若要瞭解 QLDB 如何使用指令來管理資料作業，請參閱驅動程式的工作階段管理。

取得 PartiQL 陳述式統計

Amazon QLDB 提供陳述式執行統計資料，可協助您執行更有效率的 PartiQL 陳述式，將 QLDB 的使用最佳化。QLDB 與語句的結果一起返回這些統計信息。它們包括量化耗用 I/O 使用量的測量結果和伺服器端處理時間，可用來識別效率低下的陳述式。

此功能目前可在 QLDB 主控台的 PartiQL 編輯器、QLDB 命令介面中使用，以及所有支援語言的最新版 QLDB 驅動程式。您也可以在主控台上檢視查詢歷史記錄的陳述式統計資料。

主題

I/O 用量

I/O 使用狀況測量結果說明讀取 I/O 要求的數目。如果讀取 I/O 要求的數目高於預期，則表示陳述式未最佳化，例如缺少索引。我們建議您在上一個主題最佳查詢模式中檢閱「最佳化查詢效能」。

> **Note**
>
> 當您在非空白的表格上執行`CREATE INDEX`敘述句時，I/O 使用狀況測量結果只會包含同步索引建立呼叫的讀取要求。
> QLDB 會以非同步方式建立資料表中任何現有文件的索引。這些非同步讀取要求不會包含在陳述式結果的 I/O 使用量度中。非同步讀取要求會分別收費，並在索引建置完成後新增至總讀取 I/O 中。

使用 QLDB 主控台

若要使用 QLDB 主控台取得陳述式的讀取 I/O 使用量，請執行下列步驟：

1. 開啟亞馬遜 QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在導覽窗格中，選擇 PartiQL 編輯器。
3. 從分類帳的下拉式清單中選擇分類帳。
4. 在查詢編輯器視窗中，輸入您選擇的任何陳述式，然後選擇 [執行]。以下是查詢範例。

```
SELECT * FROM testTable WHERE firstName = 'Jim'
```

若要執行陳述式，您也可以使用鍵盤快速鍵Ctrl + (適用Enter於 Windows) 或Cmd + (適用Return於 macOS)。如需更多鍵盤快速鍵，請參閱編 PartiQL 鍵盤快捷鍵。

5. 在查詢編輯器視窗下方，您的查詢結果包括讀取 I/O，也就是陳述式發出的讀取要求數目。

您也可以執行下列步驟，檢視查詢歷史記錄的讀取 I/O：

1. 在功能窗格中，選擇 PartiQL 編輯器下的最近查詢。
2. 「讀取 I/O」欄會顯示每個敘述句發出的讀取要求數目。

使用 QLDB 驅動程式

若要使用 QLDB 驅動程式取得陳述式的 I/O 使用量，請呼叫結果的串流游標或緩衝游標的`getConsumedIOs`作業。

下列程式碼範例示範如何從陳述式結果資料流游標取得讀取 I/O。

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**
>
> 若要轉換為同步程式碼，請移除`await`和`async`關鍵字，然後將`IAsyncResult`類型變更為`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))
```

下列程式碼範例示範如何從陳述式結果的緩衝游標取得讀取 I/O。這會傳回讀取來源`ExecuteStatement`和`FetchPage`要求的總讀取 I/O。

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**
>
> 若要轉換為同步程式碼，請移除await和async關鍵字，然後將IAsyncResult類型變更為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**
>
> 流光標是有狀態的，因為它分頁結果集。因此，`getConsumedIOs`和`getTimingInformation`作業會從您呼叫時傳回累積量度。緩衝的游標會緩衝記憶體中的結果集，並傳回累計量的總計量度。

時間資訊

計時資訊測量結果說明伺服器端處理時間 (毫秒)。伺服器端處理時間定義為 QLDB 花在處理陳述式的時間量。這不包括花費在網絡呼叫或暫停上的時間。此測量結果可消除 QLDB 服務端的處理時間與從屬端處理時間的歧義。

使用 QLDB 主控台

若要使用 QLDB 主控台取得陳述式的計時資訊，請執行下列步驟：

1. 開啟亞馬遜 QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在導覽窗格中，選擇 PartiQL 編輯器。

3. 從分類帳的下拉式清單中選擇分類帳。
4. 在查詢編輯器視窗中，輸入您選擇的任何陳述式，然後選擇 [執行]。以下是查詢範例。

```
SELECT * FROM testTable WHERE firstName = 'Jim'
```

若要執行陳述式，您也可以使用鍵盤快速鍵Ctrl + (適用Enter於 Windows) 或Cmd + (適用Return於 macOS)。如需更多鍵盤快速鍵，請參閱編 PartiQL 鍵盤快捷鍵。

5. 在查詢編輯器視窗下方，您的查詢結果包含伺服器端延遲，也就是 QLDB 收到陳述式要求與傳送回應之間的時間長度。這是總查詢持續時間的子集。

您也可以執行下列步驟，檢視查詢歷史記錄的計時資訊：

1. 在功能窗格中，選擇 PartiQL 編輯器下的最近查詢。
2. 執行時間 (毫秒) 資料行會顯示每個陳述式的這項計時資訊。

使用 QLDB 驅動程式

若要使用 QLDB 驅動程式取得陳述式的計時資訊，請呼叫結果串流游標或緩衝游標的`getTimingInformation`作業。

下列程式碼範例示範如何從陳述式結果資料流游標取得處理時間。

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**
>
> 若要轉換為同步程式碼，請移除`await`和`async`關鍵字，然後將`IAsyncResult`類型變更為`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))
```

下列程式碼範例示範如何從陳述式結果的緩衝游標取得處理時間。這將返回來自`ExecuteStatement`和`FetchPage`請求的總處理時間。

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**
>
> 若要轉換為同步程式碼，請移除await和async關鍵字，然後將IAsyncResult類型變更為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**
>
> 流光標是有狀態的，因為它分頁結果集。因此，`getConsumedIOs`和`getTimingInformation`作業會從您呼叫時傳回累積量度。緩衝的游標會緩衝記憶體中的結果集，並傳回累計量的總計量度。

若要瞭解如何查詢系統目錄，請繼續執行查詢系統目錄。

查詢系統目錄

您在 Amazon QLDB 分類帳中建立的每個表格都有一個系統指派的唯一 ID。您可以透過查詢系統目錄表格來尋找資料表的 ID、其索引清單及其他中繼資料`information_schema.user_tables`。

所有系統指派的 ID 都是通用唯一識別碼 (UUID)，每個識別碼都以 Base62 編碼的字串表示。如需詳細資訊，請參閱亞馬遜 QLDB 中的唯一 ID。

下列範例會顯示傳回資料VehicleRegistration表中繼資料屬性的查詢結果。

```
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"
}
```

表中繼資料欄位

- tableId— 資料表的唯一 ID。
- name— 表格名稱。
- indexes— 表格上的索引清單。
  - indexId— 索引的唯一 ID。
  - expr— 已編製索引的文件路徑。此欄位的格式為字串：[fieldName]。
  - status— 索引的目前狀態 (BUILDINGFINALIZINGONLINE、FAILED、或DELETING)。在狀態為之前，QLDB 不會在查詢中使用索引ONLINE。
  - message— 描述索引FAILED狀態之原因的錯誤訊息。此欄位只有失敗索引的欄位。
- status— 表格的目前狀態 (ACTIVE或INACTIVE)。當你它變INACTIVE成一DROP個表。

若要瞭解如何使用DROP TABLE和陳述式來管理資料UNDROP TABLE表，請繼續執行管理資料表。

管理資料表

本節說明如何使用 Amazon QLDB 中的DROP TABLE和UNDROP TABLE陳述式來管理資料表。它也會說明如何在建立表格時標記資料表。您可以建立的使用中資料表和總表格數量的配額定義在中亞馬遜 QLDB 中的配額和限制。

主題

建立時標記資料表

> **Note**
>
> 目前僅在STANDARD權限模式下，分類帳支援在建立時標記表格。

您可以標記資料表資源。若要管理現有資料表的標籤，請使用AWS Management Console或 API 作業TagResourceUntagResource、和ListTagsForResource。如需詳細資訊，請參閱標記 Amazon QLDB 資源。

您也可以在建立資料表時使用 QLDB 主控台，或在CREATE TABLE PartiQL 陳述式中指定資料表標籤來定義資料表標籤。下列範例會建立以標籤命名Vehicle的資料表environment=production。

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

藉由在建立時為資源建立標籤，您可以消除在資源建立後執行自訂標籤指令碼的必要。標記資料表之後，您可以根據這些標籤來控制對資料表的存取。例如，您只能將完整存取權授與具有特定標籤的資料表。如需 JSON 政策範例，請參閱根據表格標籤對所有動作的完整存取權。

刪除表

要刪除表，請使用基本DROP TABLE語句。當您在 QLDB 中放置一個表時，您只是停用它。

例如，下列陳述式會停用資VehicleRegistration料表。

```
DROP TABLE VehicleRegistration
```

DROP TABLE陳述式會傳回系統指派的資料表 ID。的狀態現在VehicleRegistration應該INACTIVE在系統目錄表格資訊 _schema.user_table 中。

```
SELECT status FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

查詢非作用中表格的歷史記錄

除了資料表名稱之外，您也可以歷史功能使用資料表 ID 做為第一個輸入引數來查詢 QLDB。您必須使用資料表 ID 來查詢非使用中資料表的記錄。停用資料表之後，您就無法再使用資料表名稱查詢其歷史記錄。

首先，透過查詢系統目錄資料表來尋找資料表 ID。例如，下列查詢會傳回tableId資VehicleRegistration料表的。

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

然後，您可以使用此 ID 從中運行相同的歷史查詢查詢修訂記錄。下列範例會ADR2Ll1fGsU4Jr4EqTdnQF從資料表 ID 查詢文件 ID 的歷程記錄5PLf9SXwndd63lPaSIa0O6。資料表 ID 是字串常值，而且必須以一對單引號括住。

```
--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'
```

重新啟動表格

停用 QLDB 中的資料表之後，您可以使用取消刪除資料表陳述式重新啟用它。

首先，從中找到表 IDinformation_schema.user_tables。例如，下列查詢會傳回tableId資VehicleRegistration料表的。狀態應該是INACTIVE。

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

然後，使用此 ID 重新啟用表格。下列是取消刪除資料表 ID 的範例5PLf9SXwndd63lPaSIa0O6。在這種情況下，資料表 ID 是您用雙引號括住的唯一識別碼。

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

現在的狀態VehicleRegistration應該是ACTIVE。

若要瞭解如何建立、描述和卸除索引，請繼續執行管理索引。

管理索引

本節說明如何在 Amazon QLDB 中建立、描述和刪除索引。您可以建立之每個資料表索引數量因素而異亞馬遜 QLDB 中的配額和限制。

主題

- 建立索引
- 描述索引
- 刪除索引
- 常見錯誤

建立索引

如中所述建立資料表及索引，您可以使用 CREATE INDEX 陳述式在指定的頂層欄位的資料表上建立索引，如下所示。資料表名稱和索引欄位名稱都區分大小寫。

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

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

您在資料表上建立的每個索引都有系統指派的唯一 ID。若要尋找此索引 ID，請參閱下一節描述索引。

> **Important**
>
> QLDB 需要一個索引來有效地查找文檔。如果沒有索引，QLDB 需要在讀取文檔時進行全表掃描。這可能會造成大型資料表的效能問題，包括並行衝突和交易逾時。

若要避免資料表掃描，您必須在索引欄位或文件 ID 上使用相等運算子 (=或IN) 執行具有述WHERE詞子句的陳述式。如需詳細資訊，請參閱最佳化查詢效能。

建立索引時，請注意下列條件約束：

- 索引只能在單一頂層欄位上建立。不支援複合、巢狀、唯一和以函數為基礎的索引。
- 您可以在任何 Ion 資料類型上建立索引，包括`list`和`struct`。但是，無論 Ion 類型如何，您都只能通過整個 Ion 值的相等性來進行索引查找。例如，當使用`list`類型作為索引時，您不能通過列表中的一個項目進行索引查找。
- 只有在使用相等述詞時才會改善查詢效能；例如，`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`。

  QLDB 不尊重查詢謂詞中的不等式。因此，範圍篩選的掃描不會實作。
- 索引欄位的名稱區分大小寫，且最大長度可為 128 個字元。
- QLDB 中的索引建立是非同步的。完成建立非空白資料表索引所需的時間量因資料表大小而異。如需詳細資訊，請參閱管理索引。

描述索引

QLDB 中的索引建立是非同步的。完成建立非空白資料表索引所需的時間量因資料表大小而異。若要檢查索引建置的狀態，您可以查詢系統目錄資料表資訊 _schema.user_tables。

例如，下列陳述式會查詢表格上所有索引的系統目`VehicleRegistration`錄。

```
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"
}
```

索引欄位

- indexId— 索引的唯一 ID。
- expr— 已編製索引的文件路徑。此欄位的格式為字串：[fieldName]。
- status— 索引的目前狀態。索引的狀態可以是下列其中一個值：
  - BUILDING— 正在積極建立表格的索引。
  - FINALIZING— 已完成建立索引，並開始啟動索引以供使用。
  - ONLINE— 處於作用中狀態且可在查詢中使用。在狀態為連線之前，QLDB 不會在查詢中使用索引。
  - FAILED— 由於無法復原的錯誤，無法建立索引。處於此狀態的索引仍會計入每個資料表的索引配額。如需詳細資訊，請參閱常見錯誤。
  - DELETING— 使用者捨棄索引之後，正在主動刪除索引。
- message— 描述索引FAILED狀態之原因的錯誤訊息。此欄位只有失敗索引的欄位。

使用主控台

您也可以使用AWS Management Console檢查索引的狀態。

檢查索引（主控台）的狀態

1. 登入AWS Management Console，開啟位於 https://console.aws.amazon.com/qldb 的 Amazon QLDB 主控台，開啟位於https://console.aws.amazon.com/qldb的 Amazon QLDB 主控台。
2. 在導覽窗格中，選擇分類帳。
3. 在「分類帳」清單中，選擇您要管理其索引的分類帳名稱。
4. 在分類帳詳細資訊頁面的「表格」標籤下，選擇要檢查其索引的表格名稱。
5. 在資料表詳細資料頁面上，找出 [索引欄位] 卡片。[索引狀態] 資料行會顯示資料表上每個索引的目前狀態。

刪除索引

使用該DROP INDEX語句刪除索引。當您卸除索引時，它會從資料表中永久刪除。

首先，從中找到索引 IDinformation_schema.user_tables。例如，下列查詢會傳回indexId資VehicleRegistration料表上索引LicensePlateNumber欄位的。

```
SELECT indexes.indexId
FROM information_schema.user_tables info, info.indexes indexes
WHERE info.name = 'VehicleRegistration' and indexes.expr = '[LicensePlateNumber]'
```

然後，使用此 ID 刪除索引。下列是卸除索引 ID 的範例4tPW3fUhaVhDinRgKRLhGU。索引 ID 是唯一 ID，應以雙引號括住。

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

> **Note**
>
> 所有DROP INDEX陳述式WITH (purge = true)都需要此子句，而且true是目前唯一支援的值。
> 關鍵字區purge分大小寫，且必須完全使用小寫。

使用主控台

您也可以使AWS Management Console用刪除索引。

刪除索引（控制台）

1. 登入AWS Management Console，開啟位於 https://console.aws.amazon.com/qldb 的 Amazon QLDB 主控台，開啟位於https://console.aws.amazon.com/qldb的 Amazon QLDB 主控台。
2. 在導覽窗格中，選擇分類帳。
3. 在「分類帳」清單中，選擇您要管理其索引的分類帳名稱。
4. 在分類帳詳細資訊頁面的「表格」標籤下，選擇要刪除其索引的表格名稱。
5. 在資料表詳細資料頁面上，找出 [索引欄位] 卡片。選取您要刪除的索引，然後選擇 [刪除索引]。

常見錯誤

本節說明建立索引時可能會遇到的常見錯誤，並建議可能的解決方案。

> **Note**
>
> 狀態為的索引FAILED仍會計入您每個資料表的索引配額。失敗的索引也會防止您修改或刪除造成表格上索引建立失敗的任何文件。
> 您必須明確刪除索引，才能將其從配額中移除。

文檔包含索引字段的多個值：字段 ***fieldName***。

QLDB 無法為指定的欄位名稱建立索引，因為資料表包含相同欄位具有多個值的文件 (也就是重複的欄位名稱)。

您必須先卸除失敗的索引。然後，在重試索引建立之前，請確定表格中的所有文件對每個欄位名稱都只有一個值。您也可以為另一個沒有重複項的欄位建立索引。

如果您嘗試插入包含資料表上已編製索引之欄位的多個值的文件，QLDB 也會傳回這個錯誤。

超出索引限制：資料 ***tableName*** ***#***已經有 ***n*** 個索引，而且無法建立更多索引。

QLDB 會強制執行每個資料表五個索引的限制，包括失敗的索引。您必須先刪除現有索引，然後再建立新索引。

沒有定義的索引與標識符：***## ID***。

您嘗試刪除指定資料表和索引 ID 組合不存在的索引。若要瞭解如何檢查現有索引，請參閱描述索引。

亞馬遜 QLDB 中的唯一 ID

本節說明 Amazon QLDB 中系統指派的唯一識別碼的屬性和使用準則。它也提供 QLDB 唯一識別碼的一些範例。

主題

- 屬性
- 用量
- 範例

屬性

QLDB 中所有系統指派的 ID 都是通用唯一識別碼 (UUID)。每個 ID 具有下列屬性：

- 128 位元無線識別碼
- 以 62 編碼文字表示
- 固定長度為 22 個字元的英數字串 (例如:3Qv67yjXEwB9SjmvkuG6Cp)

用量

在應用程式中使用 QLDB 唯一 ID 時，請注意下列準則：

做

- 將 ID 視為字串。

不

- 嘗試解碼字符串。
- 歸因於字符串的語義含義（例如派生一個時間分量）。
- 按語義順序對字符串進行排序。

範例

下列屬性是 QLDB 中唯一識別碼的一些範例：

- 文件識別碼
- 索引識別碼
- 鏈編號
- 資料表 ID
- 交易 ID

Amazon QLLDB 並行模型

Amazon QLDB 旨在處理高效能線上交易處理 (OLTP) 工作負載。QLDB 支援類似 SQL 的查詢功能，並提供完整的 ACID 交易。此外，QLDB 資料項目也是文件，提供結構描述彈性和直覺式資料建模。以日誌為核心，您可以使用 QLDB 存取資料所有變更的完整且可驗證的歷史記錄，並視需要將一致性交易串流至其他資料服務。

主題

樂觀的並發控制

在 QLDB 中，並發控制使用樂觀並發控制（OCC）來實現。OCC 的運作原則是，多個交易可以經常完成而不會相互干擾。

使用 OCC，QLDB 中的事務不會獲取數據庫資源的鎖定，並以完全可序列化隔離進行操作。QLDB 會以序列方式執行並行異動，因此產生的效果與這些異動是以序列方式啟動一樣。

認可之前，每個交易都會執行驗證檢查，以確保沒有其他認可的交易已修改它所存取的資料。如果此檢查顯示發生衝突的修改，或資料狀態變更，則會拒絕認可交易。但是，可以重新啟動交易。

當事務寫入 QLDB，OCC 模型的驗證檢查由 QLDB 本身實現。如果交易無法寫入日誌，因為在 OCC 的驗證階段失敗，QLDB 會傳回應用程式層。`OccConflictException`應用程式軟體負責確保交易重新啟動。應用模組應中止被拒絕的交易，然後從一開始就重試整個交易。

若要瞭解 QLDB 驅動程式如何處理和重試 OCC 衝突和其他暫時性例外狀況，請參閱瞭解亞馬遜 QLDB 中的驅動程序的重試策略。

使用索引來避免完整表格掃描

在 QLDB 中，每個 PartiQL 陳述式 (包括每個`SELECT`查詢) 都會在交易中處理，並受到交易逾時限制的限制。

最佳做法是，您應該執行含有WHERE述詞子句的陳述式，以篩選索引欄位或文件 ID。QLDB 需要索引欄位上的相等運算子，才能有效率地查找文件；例如，WHERE indexedField = 123或WHERE indexedField IN (456, 789)。

如果沒有這個索引查找，QLDB 需要在讀取文檔時進行全表掃描。這可能會導致查詢延遲和交易逾時，也會增加 OCC 與競爭交易衝突的可能性。

例如，假設名Vehicle為的資料表僅在VIN欄位上有索引。它包含以下文件。

VIN	Make	Model	顏色
"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"

兩個名為 Alice 和 Bob 的並發用戶正在使用分類帳中的同一個表。他們想要更新兩個不同的文件，如下所示。

愛麗絲：

```
UPDATE Vehicle AS v
SET v.Color = 'Blue'
WHERE v.VIN = '1N4AL11D75C109151'
```

鮑勃：

```
UPDATE Vehicle AS v
SET v.Color = 'Red'
```

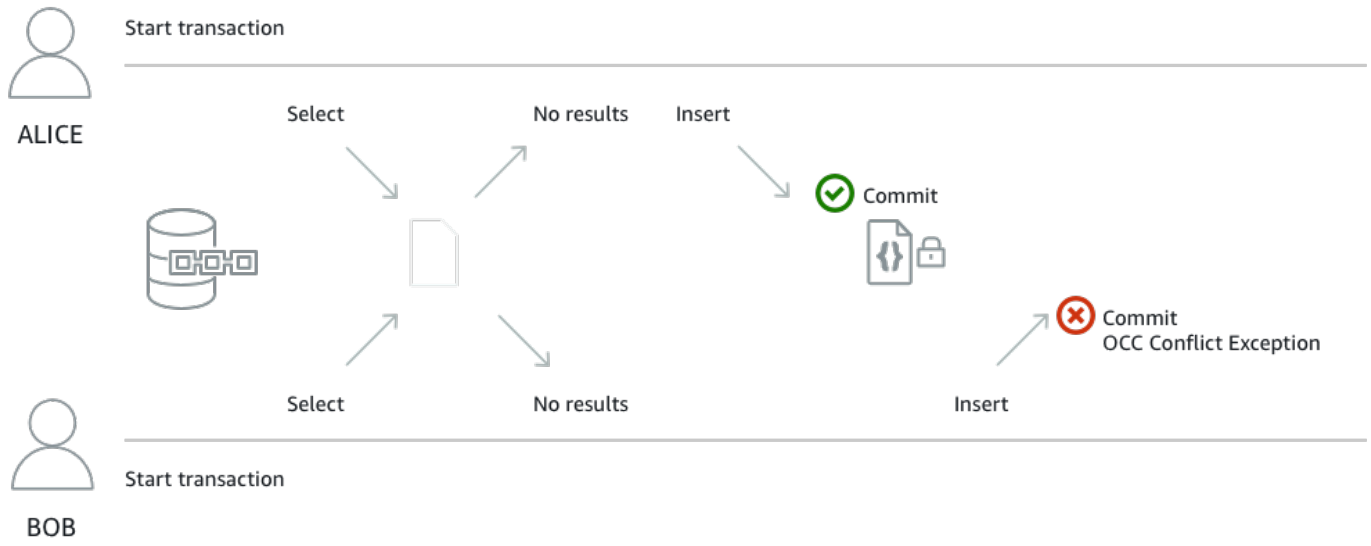
```
WHERE v.Make = 'Tesla' AND v.Model = 'Model S'
```

假設愛麗絲和鮑勃在同一時間開始他們的交易。Alice 的UPDATE陳述式會在VIN欄位上進行索引查找，因此只需要讀取該文件即可。愛麗絲完成並成功提交她的交易第一。

Bob 的陳述式會篩選非索引欄位，因此它會執行資料表掃描並遇到OccConflictException. 這是因為 Alice 認可的交易修改了 Bob 陳述式所存取的資料，其中包括資料表中的每個文件，而不僅僅是 Bob 正在更新的文件。

插入 OCC 衝突

OCC 衝突可能包含新插入的文件，而不僅僅是先前存在的文件。考慮下圖，其中兩名並行使用者 (Alice 和 Bob) 正在處理分類帳中同一個表。他們都希望只在謂詞值尚不存在的情況下插入一個新文檔。



在此範例中，Alice 和 Bob 都會在單一交易中執行下列SELECT和INSERT陳述式。他們的應用程式只會在INSERT陳述式未傳回任何結果時執行SELECT陳述式。

```
SELECT * FROM Vehicle v WHERE v.VIN = 'ABCDE12345EXAMPLE'
```

```
INSERT INTO Vehicle VALUE
{
    'VIN' : 'ABCDE12345EXAMPLE',
    'Type' : 'Wagon',
```

```
    'Year' : 2019,
    'Make' : 'Subaru',
    'Model' : 'Outback',
    'Color' : 'Gray'
}
```

假設愛麗絲和鮑勃在同一時間開始他們的交易。他們的兩個SELECT查詢都不會返回與 aVIN 的現有文檔ABCDE12345EXAMPLE。因此，他們的應用程序繼續進行INSERT聲明。

愛麗絲完成並成功提交她的交易第一。然後，鮑勃試圖提交他的事務，但 QLDB 拒絕它並拋出一個OccConflictException. 這是因為 Alice 認可的交易修改了 BobSELECT 查詢的結果集，而且 OCC 會在認可 Bob 的交易之前偵測到此衝突。

這個事務示例需要SELECT查詢是冪等的。然後 Bob 可以從一開始就重試他的整個交易。但他的下一個SELECT查詢將返回愛麗絲插入的文檔，所以鮑勃的應用程序將無法運行INSERT.

使交易冪等性的效果

上一節中的插入交易也是冪等交易的範例。換句話說，多次運行相同的事務會產生相同的結果。如果 Bob 在INSERT沒有先檢查特定項目是否VIN已存在的情況下執行，則表格最終可能會出現具有重複VIN值的文件。

除了 OCC 衝突之外，請考慮其他重試案例。例如，QLDB 可能成功地在伺服器端提交交易，但用戶端在等待回應時逾時。最佳做法是將寫入交易設為冪等，以避免在並發或重試的情況下出現任何意外的副作用。

密文 OCC 衝突

QLDB 可防止同時編輯相同分錄區塊上的修訂。假設兩個並行使用者 (Alice 和 Bob) 想要編輯分類帳中相同區塊上確認的兩個不同文件修訂版本的範例。首先，Alice 透過執行REDACT_REVISION預存程序來要求編輯一個修訂版本，如下所示。

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

然後，當 Alice 的請求仍在處理中時，Bob 要求編輯另一個修訂版本，如下所示。

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

QLDB 拒絕 Bob 的請求，`OccConflictException`即使他們試圖編輯兩個不同的文檔修訂版。這是因為 Bob 的修訂與 Alice 正在編輯的修訂位於同一個區塊上。在 Alice 的請求完成處理之後，Bob 可以重試其密文請求。

同樣地，如果兩個並行異動嘗試編輯相同的修訂版本，則只能處理一個請求。另一個要求失敗，並出現 OCC 衝突例外狀況，直到密文完成為止。之後，任何編輯相同版本的請求都會導致錯誤，指出修訂版本已經被編輯。

管理並行階段

如果您有使用關聯式資料庫管理系統 (RDBMS) 的經驗，您可能熟悉並行連線限制。QLDB 不具有傳統的 RDBMS 連接的相同概念，因為事務與 HTTP 請求和響應消息運行。

在 QLDB 中，類似的概念是一個活動的會話。工作階段在概念上與使用者登入類似，它會管理您對總帳之資料交易請求的相關資訊。使用中的工作階段是主動執行交易的工作階段。它也可以是最近完成交易的工作階段，服務預期會立即啟動另一個交易。QLDB 支援每個工作階段一個主動執行的交易。

在中定義每個分類帳的並行作用中階段作業限制亞馬遜 QLDB 中的配額和限制。達到此限制之後，任何嘗試啟動交易的工作階段都會導致 error (`LimitExceededException`)。

如需階段作業的生命週期，以及 QLDB 驅動程式在執行資料交易時如何處理工作階段的資訊，請參閱驅動程式的工作階段管理。如需使用 QLDB 驅動程式在應用程式中設定工作階段集區的最佳實務，請參閱 Amazon QLDB 驅動程式建議設定 QldbDriver 對象中的。

Amazon QLDB 中的數據驗證

使用 Amazon QLDB，您可以相信應用程式資料的變更歷史記錄是正確的。QLDB 使用不可變的交易記錄檔 (稱為日誌) 來儲存資料。日誌會追蹤已提交資料的每項變更，並維護一段時間內完整且可驗證的變更歷程記錄。

QLDB 使用 SHA-256 雜湊函式搭配以 Merkle 樹狀結構為基礎的模型來產生日誌的加密表示法，稱為摘要。摘要作為資料整個變更歷史記錄的唯一簽名，截至某個時間點。您可以使用摘要來驗證文件修訂相對於該簽名的完整性。

主題

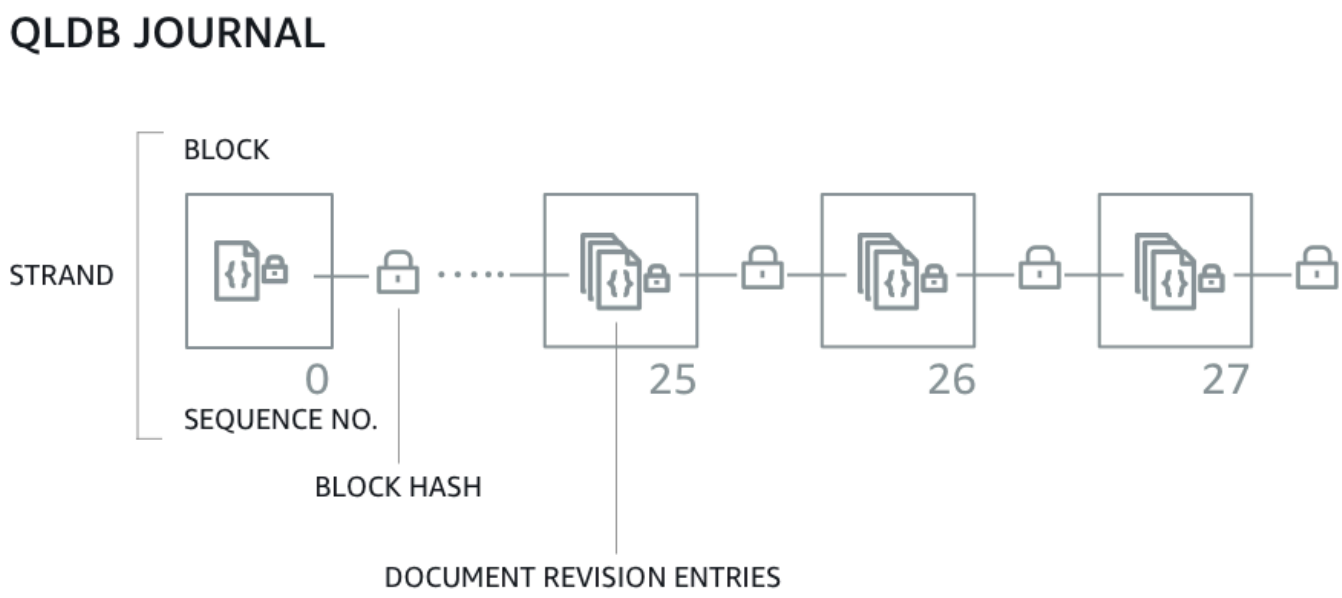
你可以在 QLDB 中驗證什麼樣的數據？

在 QLDB 中，每個分類帳只有一個期刊。一個日誌可以有多個股，這些鏈是分錄的分區。

> **Note**
>
> QLDB 目前僅支援單鏈的期刊。

區塊是在交易期間認可至日誌鏈的物件。此區塊包含項目物件，代表傳遞所產生的文件修訂版本。您可以驗證 QLDB 中的個別修訂或整個分錄區塊。

下圖說明此分錄結構。



此圖表顯示交易已確認為包含文件修訂分錄的區塊。它還表明每個塊都被哈希鏈接到後續塊，並具有一個序列號來指定其在鏈中的地址。

若要取得有關圖塊中資料內容的資訊，請參閱〈〉亞馬遜 QLDB 中的期刊內容。

資料完整性是什麼意思？

QLDB 中的數據完整性意味著分類帳的日誌實際上是不可變的。換句話說，您的資料 (特別是每個文件修訂版) 處於下列條件成立的狀態：

1. 它存在於您的期刊中首次寫入的相同位置。
2. 自從寫入以來，它沒有以任何方式被改變。

驗證如何運作？

若要瞭解驗證在 Amazon QLDB 中的運作方式，您可以將概念分解為四個基本元件。

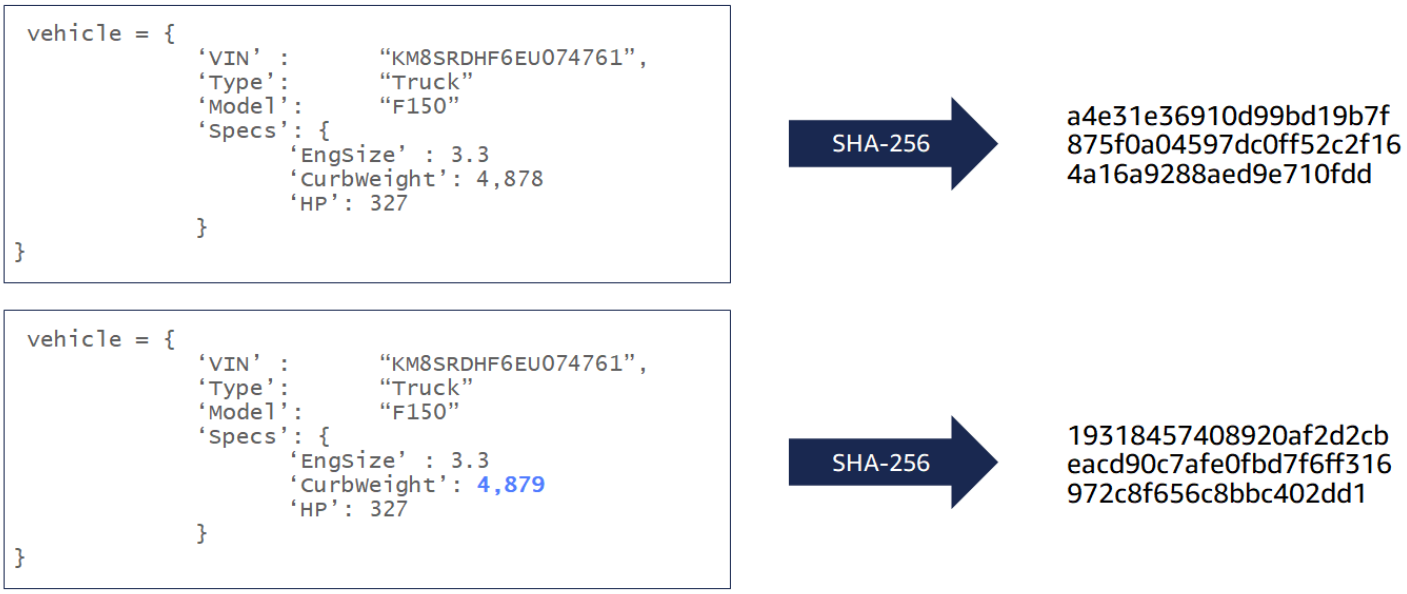
- 雜湊
- 摘要
- 默克尔树

- 證明

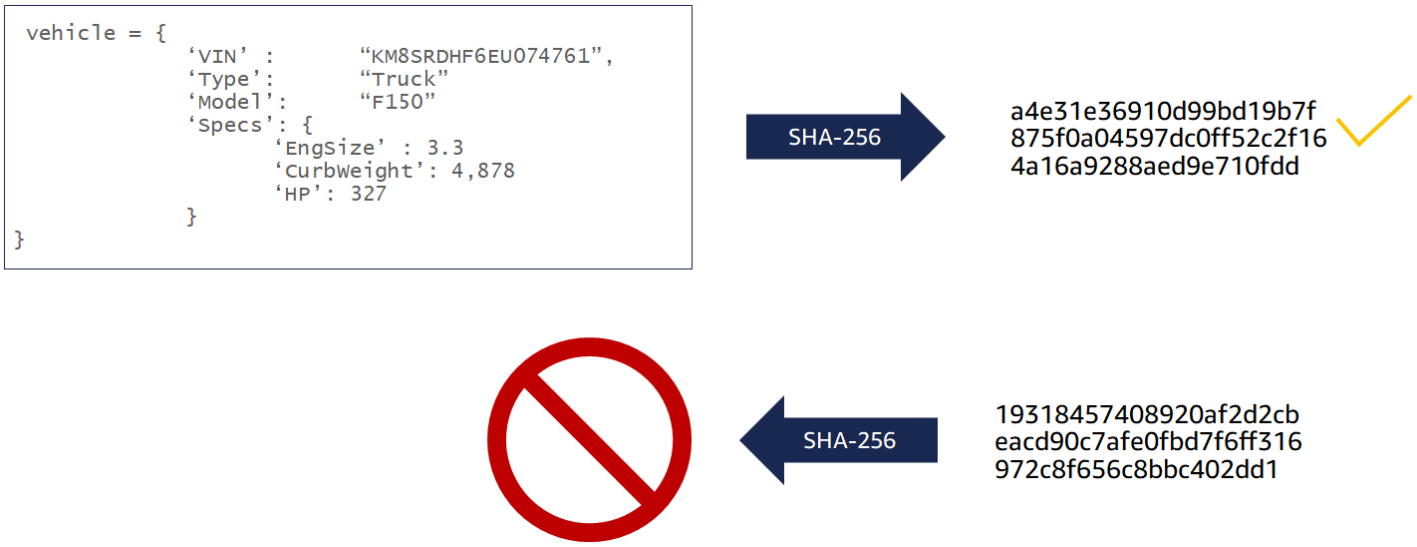
雜湊

QLDB 使用 SHA-256 密碼編譯雜湊函數來建立 256 位元雜湊值。散列充當任意數量的輸入數據的唯一固定長度簽名。如果您變更輸入的任何部分 (即使是單一字元或位元)，則輸出雜湊會完全變更。

下圖顯示了 SHA-256 哈希函數創建完全唯一的哈希值兩個 QLDB 文檔，由只有一個數字不同。



SHA-256 哈希函數是一種方式，這意味著在給定輸出時計算輸入在數學上是不可行的。下圖顯示了給定輸出散列值時，計算輸入 QLDB 文檔是不可行的。



以下數據輸入在 QLDB 中進行了哈希處理，以進行驗證：

- 文件修訂
- PartiQL 陳述式
- 修訂項目
- 分錄區塊

摘要

摘要是在某個時間點對分類帳的整個期刊進行加密表示。日誌是僅附加的，並且日誌塊被排序和哈希鏈類似於區塊鏈。

您可以隨時要求分類帳的摘要。QLDB 生成摘要並將其作為安全輸出文件返回給您。然後，您可以使用該摘要來驗證在先前時間點提交的文件修訂版本的完整性。如果您從修訂版開始並以摘要結尾來重新計算雜湊，您可以證明您的資料在兩者之間沒有被變更。

默克尔树

隨著分類帳的大小增加，重新計算日誌的完整雜湊鏈進行驗證的效率變得越來越低。QLDB 使用默克爾樹模型來解決這種低效率的問題。

默克爾樹是一種樹數據結構，其中每個葉節點表示數據塊的哈希值。每個非葉節點都是其子節點的哈希值。Merkle 樹通常用於區塊鏈，可幫助您通過審核證明機制有效地驗證大型數據集。有關默克爾樹的更多信息，請參閱默克爾樹維基百科頁面。要了解有關 Merkle 審計證明的更多信息以及示例用例，請參閱證書透明度網站上的日誌校樣如何工作。

Merkle 樹的 QLDB 實現是從日誌的完整哈希鏈構建的。在此模型中，葉節點是所有個別文件修訂版雜湊的集合。根節點表示截至某個時間點的整個日誌摘要。

使用 Merkle 稽核證明，您可以僅檢查分類帳修訂歷史記錄的一小部分來驗證修訂。您可以通過遍歷樹從給定的葉節點（修訂）到其根（摘要）來完成此操作。沿著這個遍歷路徑，您可以遞歸散列節點對兄弟節點以計算它們的父哈希，直到以摘要結束為止。此遍歷具有樹中`log(n)`節點的時間複雜度。

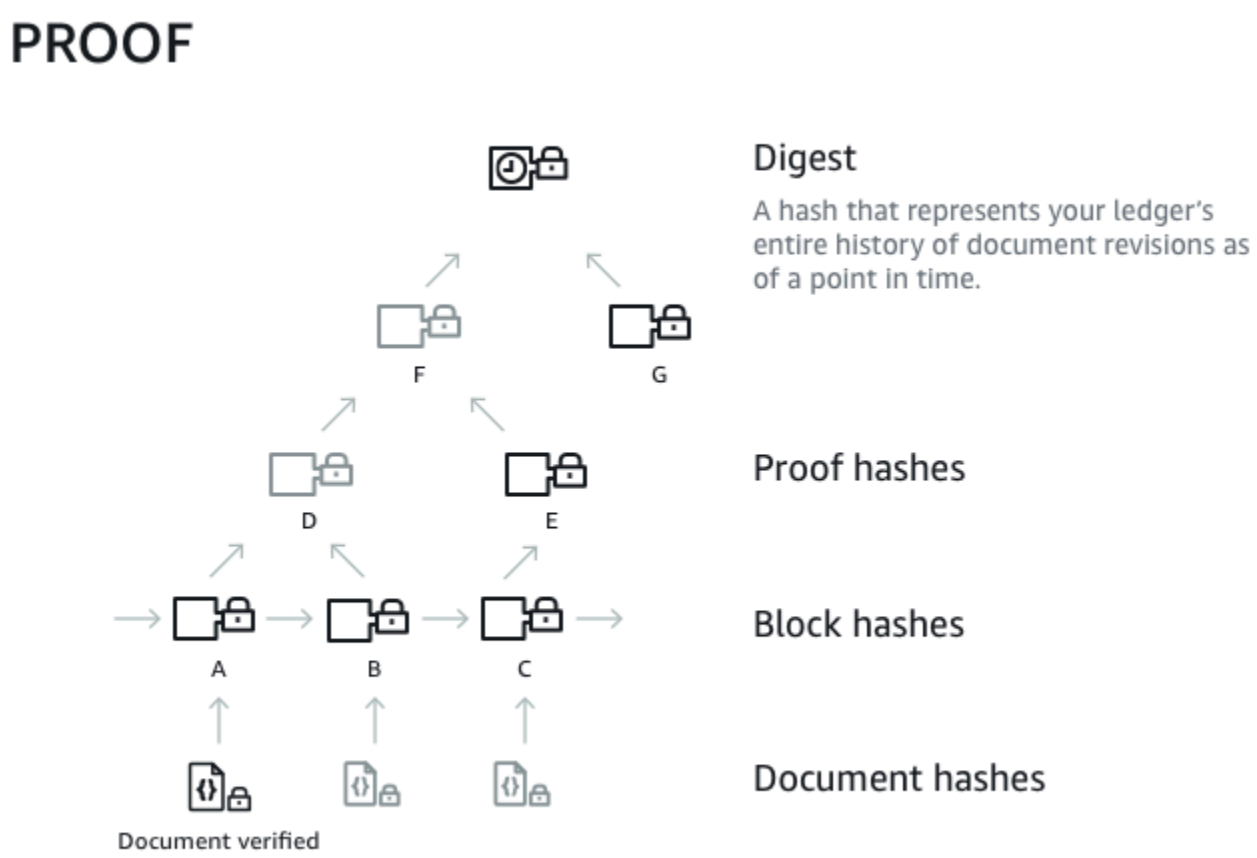
證明

證明是 QLDB 返回給定摘要和文檔修訂版的節點哈希的有序列表。它由 Merkle 樹模型將給定的葉節點哈希（修訂版）鏈接到根散列（摘要）所需的哈希組成。

在修訂版本和摘要之間變更任何已提交的資料都會破壞日誌的雜湊鏈，並且無法產生證明。

驗證範例

下圖說明了 Amazon QLDB 哈希樹模型。它顯示了一組塊哈希捲起到頂部根節點，它代表一個日誌鏈的摘要。在具有單鏈日誌的分類帳中，此根節點也是整個分類帳的摘要。



假設節點 A 是包含要驗證其哈希的文檔修訂的塊。下列節點代表 QLDB 在您的證明中提供的有序雜湊清單：B、E、G。 這些散列需要重新計算從散列 A 的摘要。

若要重新計算摘要，請執行下列動作：

1. 從哈希 A 開始，並將其與哈希 B 連接。 然後，散列結果以計算 D。
2. 使用 D 和 E 來計算 F。
3. 使用 F 和 G 計算摘要。

如果您重新計算的摘要符合預期值，則驗證成功。給定修訂散列和摘要，在證明中對哈希進行反向工程是不可行的。因此，本練習證明您的修訂確實寫在此期刊位置相對於摘要。

資料編輯如何影響驗證？

在 Amazon QLDB 中，DELETE陳述式只會以邏輯方式刪除文件，方法是建立將文件標記為已刪除的新修訂。QLDB 也支援資料編輯作業，可讓您永久刪除表格歷程記錄中非使用中的文件修訂版本。

密文作業只會刪除指定修訂版本中的使用者資料，並保持分錄序列與文件中繼資料不變。編輯修訂之後，版本修訂中的使用者資料 (由data結構表示) 會被新dataHash欄位取代。此欄位的值是已移除data結構的 Amazon Ion 雜湊值。如需密文操作的詳細資訊和範例，請參閱編輯文件修訂版本。

因此，分類帳會維護其整體資料完整性，並透過現有的驗證 API 作業維持密碼編譯驗證。您仍然可以按預期使用這些 API 操作來請求 digest (GetDigest)，請求證明（ GetBlock或 GetRevision ），然後使用返回的對象運行驗證算法。

重新計算修訂雜湊

如果您計劃透過重新計算雜湊來驗證個別文件修訂版本，您必須有條件地檢查修訂是否已編輯。如果修訂已編輯，您可以使用dataHash欄位中提供的雜湊值。如果未編輯，則可以使用該字段重新計算哈希值。data

透過執行此條件檢查，您可以識別已編輯的修訂並採取適當的動作。例如，您可以記錄資料操作事件以進行監視。

開始使用驗證

在驗證數據之前，您必須從分類帳中請求摘要並保存以備日後使用。在摘要涵蓋的最新區塊之前提交的任何文件修訂版本都有資格根據該摘要進行驗證。

然後，您向 Amazon QLDB 要求證明，以取得您要驗證的合格修訂。使用此證明，您可以呼叫用戶端 API 來重新計算摘要，從修訂雜湊開始。只要先前儲存的摘要在 QLDB 之外已知且受信任，如果重新計算的摘要雜湊符合儲存的摘要雜湊，就會證明文件的完整性。

> **Important**
>
> - 您特別證明的是，文檔修訂版在您保存此摘要的時間和運行驗證之間沒有更改。您可以在稍後要驗證的修訂確認至分錄後，立即要求並儲存摘要。
> - 作為最佳實踐，我們建議您定期請求摘要並將其存儲在分類帳之外。根據您在分類帳中確認修訂的頻率，決定您請求摘要的頻率。

如需在實際使用案例中討論加密驗證價值的詳細 AWS 部落格文章，請參閱 Amazon QLDB 的真實世界加密驗證。

有關如何從分類帳請求摘要，然後驗證數據的 step-by-step 指南，請參閱以下內容：

- 步驟 1：請求 QLDB 中的摘要
- 步驟 2：驗證您在 QLDB 中的資料

步驟 1：請求 QLDB 中的摘要

Amazon QLDB 提供一個 API 來請求摘要，其中涵蓋分類帳中日誌目前提示的摘要。日誌的提示是指 QLDB 收到您的請求時最新的已提交區塊。您可以使用 AWS Management Console、 AWS SDK 或 AWS Command Line Interface (AWS CLI) 來取得摘要。

主題

- AWS Management Console
- QLDB API

AWS Management Console

請依照下列步驟使用 QLDB 主控台要求摘要。

要求摘要（控制台）

1. 登入 AWS Management Console，然後開啟 Amazon QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在瀏覽窗格中，選擇「分類帳」。
3. 在分類帳清單中，選取您要請求摘要的分類帳名稱。
4. 選擇取得摘要。「取得摘要」對話方塊會顯示下列摘要詳細資訊：
   - 摘要 — 您要求之摘要的 SHA-256 雜湊值。
   - 摘要提示位址 — 您要求摘要涵蓋的日誌中最新的區塊位置。地址具有以下兩個字段：
     - `strandId`— 包含區塊之日誌鏈的唯一 ID。
     - `sequenceNo`— 指定圖塊在鏈中位置的索引號碼。

- 分類帳 — 您請求摘要的分類帳名稱。
- 日期 — 您請求摘要時的時間戳記。

5. 複查摘要資訊。然後選擇 Save (儲存)。您可以保留預設檔案名稱，或輸入新名稱。

> **Note**
>
> 您可能會注意到，即使您未修改分類帳中的任何資料，摘要雜湊值和提示位址值也會發生變化。這是因為每次您在 PartiQL 編輯器中執行查詢時，主控台都會擷取分類帳的系統目錄。這是一個讀取交易，會認可至日誌，並導致最新的區塊位址發生變更。

此步驟會以 Amazon Ion 格式儲存內容的純文字檔案。檔案的副檔名為，`.ion.txt`並包含前面對話方塊中列出的所有摘要資訊。以下是摘要檔案內容的範例。欄位的順序可能會因您的瀏覽器而有所不同。

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

6. 將此檔案儲存在 future 可存取的位置。稍後，您可以使用此檔案來驗證文件的修訂版本。

> **Important**
>
> 您稍後驗證的文件修訂版本必須涵蓋在您儲存的摘要中。也就是說，文檔地址的序列號必須小於或等於摘要提示地址的序列號。

QLDB API

您也可以使用 Amazon QLDB API 搭配 AWS 開發套件或. AWS CLI QLDB API 提供下列作業供應用程式使用：

- GetDigest— 傳回期刊中最新提交區塊中分類帳的摘要。該響應包括 256 位哈希值和塊地址。

如需有關使用要求摘要的資訊 AWS CLI，請參閱《命令參考》中的 get-digest 命AWS CLI 令。

範例應用程式

有關 Java 代碼示例，請參閱 GitHub 存儲庫 aws-樣amazon-qldb-dmv-sample本/-java。如需如何下載和安裝此範例應用程式的指示，請參閱安裝亞馬遜 QLDB Java 範例應用程式。請求摘要之前，請務必遵循中的步驟 1-3 Java 教學 來建立範例分類帳，並使用範例資料載入分類帳。

類別中的教學課程程式碼GetDigest提供了從範例分類帳要求摘要的`vehicle-registration`範例。

若要使用您儲存的摘要來驗證文件修訂版本，請繼續執行步驟 2：驗證您在 QLDB 中的資料。

步驟 2：驗證您在 QLDB 中的資料

Amazon QLDB 提供一個 API 來請求指定文件 ID 及其關聯區塊的證明。您也必須提供先前儲存之摘要的提示位址，如中所述步驟 1：請求 QLDB 中的摘要。您可以 AWS Management Console使用、AWS SDK 或取 AWS CLI 得證明。

然後，您可以使用 QLDB 返回的校樣，使用客戶端 API 根據保存的摘要驗證文檔修訂。這可讓您控制用來驗證資料的演算法。

主題

- AWS Management Console
- QLDB API

AWS Management Console

本節說明使用 Amazon QLDB 主控台根據先前儲存的摘要驗證文件修訂的步驟。

開始之前，請務必遵循中的步驟步驟 1：請求 QLDB 中的摘要。驗證需要先前儲存的摘要，其中涵蓋您要驗證的修訂。

驗證文件修訂版本 (主控台)

1. 開啟 Amazon QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 首先，查詢您要核對`blockAddress`的版次`id`與分類帳。這些欄位包含在文件的中繼資料中，您可以在已提交的檢視中進行查詢。

   文件`id`是系統指派的唯一 ID 字串。這`blockAddress`是一個 Ion 結構，用於指定提交修訂的塊位置。

在瀏覽窗格中，選擇 PartiQL 編輯器。

3. 選擇您要核對版次的分類帳名稱。
4. 在查詢編輯器視窗中，使用下列語法輸入SELECT陳述式，然後選擇 [執行]。

```
SELECT metadata.id, blockAddress FROM _ql_committed_table_name
WHERE criteria
```

例如，下列查詢會從中建立的範例分類帳中的VehicleRegistration表格傳回文件Amazon QLDB 主控台。

```
SELECT r.metadata.id, r.blockAddress FROM _ql_committed_VehicleRegistration AS r
WHERE r.data.VIN = 'KM8SRDHF6EU074761'
```

5. 複製並儲存id查詢傳回的 AND blockAddress 值。請務必省略id欄位的雙引號。在 Amazon Ion 中，字串資料類型會以雙引號分隔。例如，您必須僅複製下列程式碼片段中的英數字元文字。

   "*LtMNJYNjSwzBLgf7sLifrG*"
6. 現在您已經選擇了文件修訂版，您可以開始驗證它的過程。

   在功能窗格中，選擇 [驗證]。
7. 在 [驗證文件] 表單的 [指定您要驗證的文件] 下，輸入下列輸入參數：

   - 分類帳 — 您要在其中驗證版次的分類帳。
   - 區塊位址 — 您在步驟 4 中查詢傳回的blockAddress值。
   - 文件 ID — 您在步驟 4 中查詢傳回的id值。
8. 在「指定用於驗證的摘要」下，選擇「選擇摘要」來選取先前儲存的摘要。如果該文件有效，這會自動填充控制台上的所有摘要字段。或者，您可以直接從摘要文件中手動複製並粘貼以下值：

   - 摘要 — 摘要檔案中的digest值。
   - 摘要提示位址 — 摘要檔案中的digestTipAddress值。
9. 檢閱文件和摘要輸入參數，然後選擇 [驗證]。

   控制台會為您自動執行兩個步驟：

   a. 向 QLDB 索取您指定文件的證明。
   b. 使用 QLDB 傳回的證明來呼叫用戶端 API，該 API 會根據提供的摘要驗證文件修訂版本。若要檢查此驗證演算法，請參閱以下章節QLDB API以下載程式碼範例。

主控台會在「驗證結果」卡片中顯示您要求的結果。如需詳細資訊，請參閱 驗證結果。

QLDB API

您也可以使用 Amazon QLDB API 搭配 AWS 開發套件或. AWS CLI QLDB API 提供下列作業供應用程式使用：

- `GetDigest`— 傳回期刊中最新提交區塊中分類帳的摘要。該響應包括 256 位哈希值和塊地址。
- `GetBlock`— 傳回日誌中指定位址的區塊物件。如果提供，還返回指定塊的證明進`DigestTipAddress`行驗證。
- `GetRevision`— 傳回指定文件 ID 和區塊位址的修訂資料物件。如`DigestTipAddress`果有提供，也會傳回指定修訂版本的證明以進行驗證。

如需這些 API 作業的完整說明，請參閱Amazon QLDB API 參考參考.

若要取得有關使用驗證資料的資訊 AWS CLI，請參閱《AWS CLI 指令參考》。

範例應用程式

有關 Java 代碼示例，請參閱 GitHub 存儲庫 aws-樣amazon-qldb-dmv-sample本/-java。如需如何下載和安裝此範例應用程式的指示，請參閱安裝亞馬遜 QLDB Java 範例應用程式。進行驗證之前，請確定您遵循中的步驟 1-3 Java 教學 來建立範例分類帳，並使用範例資料載入分類帳。

類別中的教學課程程式碼GetRevision提供了要求文件修訂的校樣，然後驗證該修訂版的範例。此類別會執行下列步驟：

1. 請求範例分類帳中的新摘要`vehicle-registration`。
2. 請求從`vehicle-registration`分類帳`VehicleRegistration`表格中的樣本文件修訂的證明。
3. 使用傳回的摘要和校樣來驗證範例修訂。

驗證結果

本節說明上的 Amazon QLDB 資料驗證請求傳回的結果。 AWS Management Console如需如何提交驗證要求的詳細步驟，請參閱步驟 2：驗證您在 QLDB 中的資料。

在 QLDB 主控台的 [驗證] 頁面上，您要求的結果會顯示在 [驗證結果] 卡片中。「校樣」標籤會顯示 QLDB 針對您指定的文件修訂版本和摘要傳回的校樣內容。它包括以下詳細信息：

- 修訂雜湊 — SHA-256 值，唯一代表您正在驗證的文件修訂版本。
- 校對雜湊 — QLDB 提供的排序雜湊清單，用來重新計算指定的摘要。控制台從修訂散列開始，並按順序將其與每個校樣哈希結合起來，直到以重新計算的摘要結束。

  該列表默認情況下是折疊的，因此您可以將其展開以顯示哈希值。或者，您可以按照中所述的步驟自行嘗試哈希計算使用證明重新計算摘要。
- 已計算摘要 — 在修訂雜湊上完成的一系列雜湊計算所產生的雜湊值。如果此值與之前儲存的摘要相符，則驗證成功。

封鎖索引標籤會顯示封鎖內容，其中包含您正在驗證的修訂。它包括以下詳細信息：

- 交易 ID — 認可此區塊之交易的唯一 ID。
- 交易時間 — 此區塊提交至串的時間戳記。
- 區塊雜湊 — 唯一代表此區塊及其所有內容的 SHA-256 值。
- 區塊位址 — 確認此區塊的分類帳分錄中的位置。地址具有以下兩個字段：
  - 串 ID — 包含此區塊之日誌鏈的唯一 ID。
  - 序號 — 指定此圖塊在鏈中位置的索引號碼。
- 「陳述式」— 為了確認此區塊中的項目而執行的 PartiQL 陳述式。

> **Note**
>
> 如果您以程式設計方式執行參數化陳述式，則會使用繫結參數而非常值資料將它們記錄在日誌區塊中。例如，您可能會在日誌區塊中看到下列陳述式，其中問號 (?) 是文件內容的變數預留位置。
>
> ```
> INSERT INTO Vehicle ?
> ```

- 文件條目 — 在此區塊中提交的文件修訂版本。

如果您的請求無法驗證文件修訂版本，請參閱以取驗證的常見錯誤得可能原因的詳細資訊。

使用證明重新計算摘要

QLDB 傳回文件驗證要求的證明後，您可以嘗試自行執行雜湊計算。本節說明使用提供的校樣重新計算摘要的高階步驟。

首先，將修訂雜湊與 Proof 雜湊清單中的第一個雜湊配對。然後，執行以下步驟。

1. 排序兩個雜湊。通過散列按照小端順序的帶符號字節值進行比較。
2. 按排序順序連接兩個哈希。
3. 使用 SHA-256 散列生成器哈希串聯的對。
4. 將您的新雜湊與校樣中的下一個雜湊配對，然後重複步驟 1-3。在您處理最後一個校樣雜湊之後，您的新雜湊就是重新計算的摘要。

如果您重新計算的摘要符合先前儲存的摘要，則會成功驗證您的文件。

如需具有示範這些驗證步驟之 step-by-step 程式碼範例的教學課程，請繼續執行教學課程：使用 AWS SDK 驗證資料。

教學課程：使用 AWS SDK 驗證資料

在本教學中，您可以透過 SDK 使用 QLDB API 來驗證 Amazon QLDB 分類帳中的文件修訂雜湊和日誌區塊雜湊。 AWS 您也可以使用 QLDB 驅動程式來查詢文件修訂版本。

考慮一個例子，其中您有一個文件修訂版，其中包含車輛識別號碼（VIN）為的車輛的數據`KM8SRDHF6EU074761`。文件修訂版本位於名為的分類帳中的`VehicleRegistration`表格中`vehicle-registration`。假設您要核對此工具的文件版次與包含版次之分錄區塊的完整性。

> **Note**
>
> 如需在實際使用案例中討論加密驗證價值的詳細 AWS 部落格文章，請參閱 Amazon QLDB 的真實世界加密驗證。

主題

- 必要條件
- 步驟 1：請求摘要
- 步驟 2：查詢文件修訂

必要條件

在開始之前，請確保您執行以下操作：

1. 完成下的相應先決條件，為您選擇的語言設定 QLDB 驅動程式。開始使用 Amazon QLDB 驅動程式這包括註冊 AWS、授與程式設計存取以供開發使用，以及設定您的開發環境。
2. 遵循中Amazon QLDB 主控台的步驟 1—2，建立名為的分類帳，`vehicle-registration`並使用預先定義的範例資料載入該分類帳。

接下來，檢閱下列步驟以瞭解驗證的運作方式，然後從頭到尾執行完整的程式碼範例。

步驟 1：請求摘要

在驗證資料之前，您必須先從分類帳要求摘要以`vehicle-registration`供日後使用。

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')
```

步驟 2：查詢文件修訂

使用 QLDB 驅動程式來查詢與 VIN 相關聯的區塊位址、雜湊和文件 ID。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)
```

步驟 3：要求修訂證明

重複查詢結果，並使用每個區塊位址和文件 ID 以及分類帳名稱來提交GetRevision請求。若要取得修訂的校樣，您還必須提供先前儲存摘要中的提示位址。此 API 作業會傳回包含文件修訂版本和修訂版本校樣的物件。

如需有關版本修訂結構及其內容的資訊，請參閱查詢文件元資料。

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)
```

然後，擷取要求修訂的校樣。

該 QLDB API 返回證明作為節點哈希的有序列表的字符串表示。若要將此字串轉換為節點雜湊的二進位表示清單，您可以使用 Amazon Ion 程式庫中的 Ion 讀取器。有關使用 Ion 庫的更多信息，請參閱 Amazon 離子食譜。

Java

在此範例中，您可IonReader以使用進行二進位轉換。

```
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

在此範例中，您可IonLoader以使用將校樣載入 Ion 資料報中。

```
string proofText = revisionResponse.Proof.IonText;
IIonDatagram proofValue = IonLoader.Default.Load(proofText);
```

Go

在此範例中，您可以使用 Ion 讀取器將證明轉換為二進位檔案，並逐一查看證明的節點雜湊清單。

```
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

在此範例中，您可以使用load函數來執行二進位轉換。

```
let proofValue: dom.Value = load(revisionResponse.Proof.IonText);
```

Python

在此範例中，您可以使用loads函數來執行二進位轉換。

```
proof_text = proof_response.get('Proof').get('IonText')
proof_hashes = loads(proof_text)
```

步驟 4：重新計算修訂的摘要

使用證明的雜湊清單重新計算摘要，從修訂雜湊開始。只要先前儲存的摘要在 QLDB 之外已知且受信任，如果重新計算的摘要雜湊符合儲存的摘要雜湊，就會證明文件修訂的完整性。

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))
```

步驟 5：請求日誌區塊的證明

接下來，您要驗證包含文件修訂版本的分錄區塊。

使用您在步驟 1 中儲存的摘要中的封鎖位址和提示位址來提交GetBlock請求。與步驟 2 中的GetRevision要求類似，您必須再次提供儲存摘要中的提示位址，以取得區塊的證明。此 API 作業會傳回包含區塊和區塊校樣的物件。

如需有關日誌區塊結構及其內容的資訊，請參閱亞馬遜 QLDB 中的期刊內容。

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)
```

然後，從結果中檢索塊哈希和證明。

Java

在此範例中，您可IonLoader以使用將區塊物件載入IonDatagram容器中。

```
String blockText = getBlockResult.getBlock().getIonText();

IonDatagram datagram = SYSTEM.getLoader().load(blockText);
ionStruct = (IonStruct)datagram.get(0);

final byte[] blockHash = ((IonBlob)ionStruct.get("blockHash")).getBytes();
```

您也可IonLoader以使用將校樣載入到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

在此範例中，您可IonLoader以使用將區塊和校樣載入到每個區塊的 Ion 資料報中。

```
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

在此範例中，您可以使用 Ion 讀取器將證明轉換為二進位檔案，並逐一查看證明的節點雜湊清單。

```
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

在此範例中，您可以使用load函數將區塊和校樣轉換為二進位。

```
const blockValue: dom.Value = load(getBlockResponse.Block.IonText)
let blockHash: Uint8Array = blockValue.get("blockHash").uInt8ArrayValue();

proofValue = load(getBlockResponse.Proof.IonText);
```

Python

在此範例中，您可以使用loads函數將區塊和校樣轉換為二進位。

```
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)
```

步驟 6：重新計算區塊中的摘要

使用證明的哈希列表重新計算摘要，從塊哈希開始。只要先前儲存的摘要在 QLDB 之外已知且受信任，如果重新計算的摘要雜湊符合儲存的摘要雜湊，就會證明區塊的完整性。

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))
```

上述程式碼範例會在重新計算摘要時使用下列dot函數。該函數接受兩個哈希的輸入，對它們進行排序，將它們連接起來，然後返回串聯數組的哈希值。

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
```

執行完整的程式碼範例

執行完整程式碼範例，如下所示，從開始到結束執行上述所有步驟。

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))
```

驗證的常見錯誤

本節說明 Amazon QLDB 針對驗證請求所擲回的執行階段錯誤。

以下是由服務返回的常見異常的列表。每個異常都包含特定的錯誤消息，然後是可以拋出它的 API 操作，簡短描述以及可能的解決方案的建議。

IllegalArgumentException

訊息：提供的離子值無效且無法剖析。

API 作業：`GetDigest, GetBlock, GetRevision`

在重試請求之前，請確保您提供了有效的 Amazon Ion 值。

IllegalArgumentException

訊息：提供的區塊位址無效。

API 作業：`GetDigest, GetBlock, GetRevision`

在重試請求之前，請確保您提供了有效的阻止地址。區塊位址是具有兩個欄位的 Amazon Ion 結構：`strandId`和`sequenceNo`。

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

IllegalArgumentException

消息：提供的摘要提示地址的序列號超出了鏈的最新提交記錄。

API 作業：`GetDigest, GetBlock, GetRevision`

您提供的摘要提示位址的序號必須小於或等於日誌串的最新已提交記錄的序號。在重試您的請求之前，請確定您提供的摘要提示位址具有有效的序號。

IllegalArgumentException

訊息：所提供區塊位址的串 ID 無效。

API 作業：`GetDigest, GetBlock, GetRevision`

您提供的區塊位址必須具有與分錄串 ID 相符的串 ID。在重試請求之前，請確保您提供了一個帶有效鏈 ID 的塊地址。

IllegalArgumentException

訊息：所提供區塊位址的序號超出線條的最新提交記錄。

API 作業：`GetBlock, GetRevision`

您提供的塊地址的序列號必須小於或等於鏈的最新提交記錄的序列號。在重試請求之前，請確保您提供了一個包含有效序號的塊地址。

IllegalArgumentException

消息：提供的塊地址的鏈 ID 必須與提供的摘要提示地址的 Strand ID 匹配。

API 作業：`GetBlock, GetRevision`

如果文件修訂或區塊存在於與您提供的摘要相同的日誌鏈中，您才能驗證該文件修訂或區塊。

IllegalArgumentException

訊息：所提供區塊位址的序號不得大於所提供摘要提示位址的序號。

API 作業：`GetBlock, GetRevision`

只有在您提供的摘要涵蓋的文件修訂或區塊時，才能驗證該文件修訂或區塊。這意味著它是在摘要提示地址之前提交給日記的。

IllegalArgumentException

訊息：在指定區塊位址的區塊中找不到提供的文件 ID。

API 操作：`GetRevision`

您提供的文件 ID 必須存在於您提供的區塊位址中。在重試您的請求之前，請確保這兩個參數是一致的。

從 Amazon QLDB 匯出日誌資料

Amazon QLDB 使用不可變的交易日誌 (稱為日誌) 來儲存資料。日誌會追蹤已提交資料的每項變更，並維護一段時間內完整且可驗證的變更歷程記錄。

您可以出於各種目的存取分類帳中的分錄內容，包括分析、稽核、資料保留、驗證以及匯出至其他系統。下列主題說明如何將日誌區塊從分類帳匯出到您的 Amazon Simple Storage Service (Amazon S3) 儲存貯體 AWS 帳戶。日誌匯出任務會以 Amazon Ion 格式的文字或二進位表示形式或 JSON 行文字格式的物件，將您的資料寫入 Amazon S3 中。

在 JSON 行格式中，匯出資料物件中的每個區塊都是以換行符分隔的有效 JSON 物件。您可以使用此格式將 JSON 匯出與 Amazon Athena 等分析工具直接整合， AWS Glue 因為這些服務可以自動剖析以換行符分隔的 JSON。如需有關格式的詳細資訊，請參閱 JSON 行。

如需 Amazon S3 的相關資訊，請參閱 Amazon 簡易儲存服務使用者指南。

> **Note**
>
> 如果您指定 JSON 做為匯出工作的輸出格式，QLDB 會在匯出的資料物件中將 Ion 日誌資料向下轉換為 JSON。如需詳細資訊，請參閱 向下轉換為 JSON 格式。

主題

在 QLDB 中要求匯出分錄

Amazon QLDB 提供一個 API，可針對指定的日期和時間範圍以及指定的 Amazon S3 儲存貯體目的地請求匯出日誌區塊。日誌匯出任務可以使用 Amazon Ion 格式的文字或二進位表示法或 JSON 行文字格式來寫入資料物件。您可以使用 AWS Management Console、 AWS SDK 或 AWS Command Line Interface (AWS CLI) 來建立匯出工作。

主題

- AWS Management Console
- QLDB API
- 匯出工作到期日

AWS Management Console

請依照下列步驟，使用 QLDB 主控台提交 QLDB 中的分錄匯出請求。

要求匯出 (主控台)

1. 登入 AWS Management Console，然後開啟 Amazon QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在導覽窗格中，選擇 [匯出]。
3. 選擇 [建立匯出工作]。
4. 在 [建立匯出工作] 頁面上，輸入下列匯出設定：
   - 分類帳 — 您要匯出其分錄區塊的分類帳。
   - 開始日期與時間 — 要匯出之日誌區塊範圍的國際標準時間 (UTC) 中的包含開始時間戳記。此時間戳記必須早於 [結束] 日期和時間。如果您提供的開始時間戳記早於分類帳`CreationDateTime`，QLDB 會將其預設為分類帳。`CreationDateTime`
   - 結束日期與時間 — 要匯出之日誌區塊範圍的獨佔結束時間戳記 (UTC)。這個日期和時間不能在 future。
   - 日誌區塊的目的地 — 匯出任務寫入資料物件時所使用的 Amazon S3 儲存貯體和前置詞名稱。使用以下 Amazon S3 URI 格式。

     ```
     s3://DOC-EXAMPLE-BUCKET/prefix/
     ```

     您必須為輸出物件指定 S3 儲存貯體名稱和選用的前置詞名稱。以下是範例。

     ```
     s3://DOC-EXAMPLE-BUCKET/journalExport/
     ```

     儲存貯體名稱和前綴必須同時符合 Amazon S3 命名規則和慣例。如需儲存貯體命名的詳細資訊，請參閱 Amazon S3 開發人員指南中的儲存貯體限制和限制。如需索引鍵名稱前置詞的詳細資訊，請參閱物件索引鍵和中繼資料。

> **Note**
>
> 不支援跨區域匯出。指定的 Amazon S3 儲存貯體必須與您的分類帳位於 AWS 區域 相同。

- S3 加密 — 匯出任務用於在 Amazon S3 儲存貯體中寫入資料的加密設定。若要進一步了解 Amazon S3 中的伺服器端加密選項，請參閱 Amazon S3 開發人員指南中的使用伺服器端加密保護資料。
  - 儲存貯體預設加密 — 使用指定 Amazon S3 儲存貯體的預設加密設定。
  - AES-256 — 使用伺服器端加密搭配 Amazon S3 受管金鑰 (SSE-S3)。
  - AWS-KMS — 使用伺服器端加密搭配 AWS KMS 受管理金鑰 (SSE-KMS)。

    如果您選擇此類型和選擇其他 AWS KMS key選項，則還必須以下列 Amazon 資源名稱 (ARN) 格式指定對稱加密 KMS 金鑰。

    ```
    arn:aws:kms:aws-region:account-id:key/key-id
    ```

- 服務存取 — 在 Amazon S3 儲存貯體中授予 QLDB 寫入許可的 IAM 角色。如果適用，IAM 角色還必須授與 QLDB 許可才能使用您的 KMS 金鑰。

  若要在請求日誌匯出時將角色傳遞給 QLDB，您必須擁有對 IAM 角色資源執行*iam:PassRole*動作的權限。
  - 建立和使用新的服務角色 — 讓主控台為您建立具有指定 Amazon S3 儲存貯體所需許可的新角色。
  - 使用現有的服務角色 — 若要了解如何在 IAM 中手動建立此角色，請參閱匯出權限。
- 輸出格式 — 匯出分錄資料的輸出格式
  - 離子文本- (默認) Amazon 離子的文本表示
  - 離子二進制-Amazon 離子的二進制表示
  - JSON — 以換行符分隔的 JSON 文本格式

    如果您選擇 JSON，QLDB 會在匯出的資料物件中將離子日誌資料向下轉換為 JSON。如需詳細資訊，請參閱 向下轉換為 JSON 格式。

5. 如果設定符合您的需求，請選擇 [建立匯出工作]。

完成匯出工作所需的時間，視資料大小而有所不同。如果您的要求提交成功，主控台會返回「匯出」主頁面，並列出匯出工作及其目前狀態。

6. 您可以在 Amazon S3 主控台上看到您的匯出物件。

   前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。

   若要進一步瞭解這些輸出物件的格式，請參閱QLDB 中的分錄匯出輸出。

> **Note**
>
> 匯出工作在完成後七天到期。如需詳細資訊，請參閱 匯出工作到期日。

QLDB API

您也可以使用 Amazon QLDB API 搭配 AWS 開發套件或. AWS CLI QLDB API 提供下列作業供應用程式使用：

- `ExportJournalToS3`— 將日期和時間範圍內的日誌內容從指定分類帳匯出到指定的 Amazon S3 儲存貯體。匯出任務可以將資料寫入為 Amazon Ion 格式的文字或二進位表示法，或是 JSON 行文字格式的物件。
- `DescribeJournalS3Export`— 傳回分錄匯出工作的詳細資訊。輸出包括其目前狀態、建立時間，以及原始匯出請求的參數。
- `ListJournalS3Exports`— 針對與目前 AWS 帳戶 和區域相關聯的所有分類帳，傳回分錄匯出工作摘要清單。每個匯出工作描述的輸出都包含傳回的相同詳細資訊`DescribeJournalS3Export`。
- `ListJournalS3ExportsForLedger`— 傳回指定分類帳的日誌匯出工作說明清單。每個匯出工作描述的輸出都包含傳回的相同詳細資訊`DescribeJournalS3Export`。

如需這些 API 作業的完整說明，請參閱Amazon QLDB API 參考參考.

若要取得有關使用匯出分錄資料的資訊 AWS CLI，請參閱《AWS CLI 指令參考》。

應用程式範例 (Java)

有關基本導出操作的 Java 代碼示例，請參閱 GitHub 存儲庫 aws-樣amazon-qldb-dmv-sample本/-java。如需如何下載和安裝此範例應用程式的指示，請參閱安裝亞馬遜 QLDB Java 範例應用程式。請求匯出之前，請務必遵循中的步驟 1-3 Java 教學 來建立範例分類帳，並使用範例資料載入分類帳。

下列類別中的教學課程程式碼提供建立匯出、檢查匯出狀態以及處理匯出輸出的範例。

類別	描述
ExportJournal	以 10 分鐘前的時間戳記範圍從範vehicle-registration 例分類帳匯出分錄區塊，直到現在為止。將輸出物件寫入指定的 S3 儲存貯體，或建立唯一儲存貯體 (如果未提供)。
DescribeJournalExport	說明範vehicle-registration 例分類帳exportId中指定的分錄匯出工作。
ListJournalExports	傳回vehicle-registration 範例分類帳的分錄匯出工作說明清單。
ValidateQldbHashChain	使用指exportId定的驗證vehicle-registration 範例分類帳的雜湊鏈。如果未提供，請求新的導出用於哈希鏈驗證。

匯出工作到期日

已完成的分錄匯出工作需要 7 天的保留期限。在此限制到期後，系統會自動將它們硬刪除。此到期期限為硬性限制，無法變更。

刪除完成的匯出工作後，您將無法再使用 QLDB 主控台或下列 API 作業來擷取有關工作的中繼資料：

- DescribeJournalS3Export
- ListJournalS3Exports
- ListJournalS3ExportsForLedger

但是，此到期日對匯出的資料本身沒有任何影響。所有元數據都會保留在導出編寫的清單文件中。此到期日旨在為列出日誌匯出工作的 API 作業提供更順暢的體驗。QLDB 會移除舊的匯出工作，以確保您只會看到最近的匯出，而不需要剖析多頁工作。

QLDB 中的分錄匯出輸出

除了包含日誌區塊的資料物件外，Amazon QLDB 日誌匯出任務還會寫入兩個資訊清單檔案。這些都儲存在您在匯出請求中提供的 Amazon S3 儲存貯體中。以下各節說明每個輸出物件的格式和內容。

> **Note**
>
> 如果您指定 JSON 做為匯出任務的輸出格式，QLDB 會在匯出的資料物件中將 Amazon 離子日誌資料向下轉換為 JSON。如需詳細資訊，請前往向下轉換為 JSON 格式。

主題

資訊清單檔案

Amazon QLDB 會針對每個匯出請求，在提供的 S3 儲存貯體中建立兩個資訊清單檔案。一旦您提交匯出請求，就會立即建立初始資訊清單檔案。最終的資訊清單檔案會在匯出完成後寫入。您可以使用這些檔案來檢查 Amazon S3 中匯出任務的狀態。

資訊清單檔案內容的格式對應於要求的匯出格式。

初始清單

初始資訊清單表示您的匯出工作已開始。它包含您傳遞給請求的輸入參數。除了 Amazon S3 目的地以及匯出的開始和結束時間參數之外，此檔案還包含`exportId`. 這`exportId`是 QLDB 指派給每個匯出工作的唯一識別碼。

檔案命名慣例如下。

```
s3://DOC-EXAMPLE-BUCKET/prefix/exportId.started.manifest
```

以下是初始資訊清單檔案及其內容 (Ion) 文字格式的範例。

```
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"
}
```

初始資訊清單`outputFormat`只包括在匯出要求中指定的資訊清單。如果未指定輸出格式，則匯出的資料預設為`ION_TEXT`格式。

DescribeJournalS3 匯出 API 作業和匯出的 Amazon S3 物件的內容類型也會指出輸出格式。

最終清單

最後的資訊清單表示您已完成特定分錄串的匯出工作。匯出工作會為每個鏈條寫入個別的最終資訊清單檔案。

> **Note**
>
> 在 Amazon QLDB 中，鏈條是分類帳期刊的分區。QLDB 目前僅支援單鏈的期刊。

最終資訊清單包括匯出期間寫入的資料物件金鑰的排序清單。檔案命名慣例如下。

```
s3://DOC-EXAMPLE-BUCKET/prefix/exportId.strandId.completed.manifest
```

這`strandId`是 QLDB 指派給鏈的唯一識別碼。以下是 Ion 文本格式的最終清單文件及其內容的示例。

```
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"
  ]
}
```

資料物件

Amazon QLDB 會以 Amazon Ion 格式的文字或二進位表示法或 JSON 行文字格式，在提供的 Amazon S3 儲存貯體中寫入日誌資料物件。

在 JSON 行格式中，匯出資料物件中的每個區塊都是以換行符分隔的有效 JSON 物件。您可以使用此格式將 JSON 匯出與 Amazon Athena 等分析工具直接整合， AWS Glue 因為這些服務可以自動剖析以換行符分隔的 JSON。如需有關格式的詳細資訊，請參閱 JSON 行。

資料物件名稱

分錄匯出工作會以下列命名慣例寫入這些資料物件。

```
s3://DOC-EXAMPLE-BUCKET/prefix/yyyy/mm/dd/hh/strandId.startSn-endSn.ion|.json
```

- 每個匯出工作的輸出資料都會分解為區塊。
- `yyyy/mm/dd/hh`— 您提交導出請求的日期和時間。在同一小時內匯出的物件會分組在相同的 Amazon S3 前置詞下。
- `strandId`— 包含要匯出之日誌區塊之特定鏈的唯一 ID。
- `startSn-endSn`— 包含在物件中的序號範圍。序號指定圖塊在鏈中的位置。

例如，假設您指定了下列路徑。

```
s3://DOC-EXAMPLE-BUCKET/journalExport/
```

您的匯出任務會建立類似下列內容的 Amazon S3 資料物件。此範例顯示 Ion 格式的物件名稱。

```
s3://DOC-EXAMPLE-BUCKET/journalExport/2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.1-5.ion
```

資料物件內容

每個資料物件都包含具有以下格式的分錄圖塊物件。

```
{
  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
    }
  ]
}
```

區塊是在交易期間認可至日誌的物件。區塊包含交易中繼資料，以及代表交易中認可之文件修訂的項目，以及認可它們的 PartiQL 陳述式。

以下是具有 Ion 文本格式樣本數據的塊的示例。若要取得有關圖塊物件中功能變數的資訊，請參閱〈〉亞馬遜 QLDB 中的期刊內容。

> **Note**
>
> 此區塊範例僅供參考。顯示的哈希值不是實際計算的哈希值。

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

在revisions欄位中，某些版本修訂物件可能只包含hash值，而不包含其他屬性。這些是不包含使用者資料的僅限內部系統修訂。匯出工作會將這些修訂包含在各自的區塊中，因為這些修訂的雜湊是日誌完整雜湊鏈的一部分。密碼編譯驗證需要完整的雜湊鏈。

向下轉換為 JSON 格式

如果您指定 JSON 做為匯出任務的輸出格式，QLDB 會在匯出的資料物件中將 Amazon 離子日誌資料向下轉換為 JSON。但是，在某些情況下，您的數據使用 JSON 中不存在的豐富 Ion 類型的情況下，將 Ion 轉換為 JSON 是有損的。

有關離子到 JSON 轉換規則的詳細信息，請參閱 Amazon 離子食譜中的向下轉換為 JSON。

匯出處理器程式庫 (Java)

QLDB 提供適用於 Java 的可擴充架構，可簡化 Amazon S3 中的匯出處理作業。這個框架庫處理讀取導出的輸出並按順序遍歷導出的塊的工作。要使用此導出處理器，請參閱 GitHub 存儲庫 awslab amazon-qldb-export-processor s/-java。

QLDB 中的分錄匯出權限

在 Amazon QLDB 中提交日誌匯出請求之前，您必須在指定的 Amazon S3 儲存貯體中提供具有寫入許可的 QLDB。如果您選擇受管的客戶 AWS KMS key 做為 Amazon S3 儲存貯體的物件加密類型，您還必須向 QLDB 提供使用指定對稱加密金鑰的許可。Amazon S3 不支援非對稱 KMS 金鑰。

若要為您的匯出任務提供必要的許可，您可以讓 QLDB 擔任具有適當許可政策的 IAM 服務角色。服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

> **Note**
>
> 若要在請求日誌匯出時將角色傳遞給 QLDB，您必須擁有對 IAM 角色資源執行`iam:PassRole`動作的權限。這是 QLDB 分類帳資源的`qldb:ExportJournalToS3`權限之外。
>
> 要了解如何使用 IAM 控制對 QLDB 的存取，請參閱。Amazon QLDB 如何與 IAM 合作如需 QLDB 原則範例，請參閱。Amazon QLDB 以身分識別為基礎的政策範例

在此範例中，您建立了一個角色，讓 QLDB 代表您將物件寫入 Amazon S3 儲存貯體。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

如果您是第一次將 QLDB 日誌匯出到您 AWS 帳戶 的，您必須先執行下列動作以建立具有適當政策的 IAM 角色。或者，您可以使用 QLDB 主控台自動為您建立角色。否則，您可以選擇先前建立的角色。

主題

- 建立許可政策
- 建立 IAM 角色

建立許可政策

請完成下列步驟，為 QLDB 日誌匯出工作建立權限原則。此範例顯示授與 QLDB 許可以將物件寫入指定儲存貯體的 Amazon S3 儲存貯體政策。如果適用，此範例也會顯示允許 QLDB 使用對稱加密 KMS 金鑰的金鑰原則。

如需 Amazon S3 儲存貯體政策的詳細資訊，請參閱 Amazon 簡單儲存體服務使用者指南中的使用儲存貯體政策和使用者政策。若要進一步了解關 AWS KMS 鍵原則，請參閱AWS Key Management Service 開發人員指南 AWS KMS中的使用金鑰政策。

> **Note**
>
> 您的 Amazon S3 儲存貯體和 KMS 金鑰必須與 QLDB 分類帳位於 AWS 區域 相同。

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側的導覽欄中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 選擇 JSON 標籤。
5. 輸入 JSON 政策文件。
   - 如果您使用客戶受管 KMS 金鑰進行 Amazon S3 物件加密，請使用下列範例政策文件。*#### ##########################us-east-1#12345678 9012 # 1234 ##*

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

- 對於其他加密類型，請使用下列範例原則文件。若要使用此政策，請以您自己的 Amazon S3 *########### DOC-EXAMP* LE 儲存貯體。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permission",
            "Action": [
                "s3:PutObjectAcl",
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

6. 選擇檢閱政策。

> **Note**
>
> 您可以隨時切換 Visual editor (視覺化編輯器) 與 JSON 標籤。不過，如果您進行更改或在 Visual editor (視覺編輯工具) 索引標籤中選擇 Review policy (檢閱政策)，IAM 可能會調整您的政策結構以針對視覺編輯工具進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在 Review policy (檢閱政策) 頁面上，為您正在建立的政策輸入選用的 Name (名稱) 與 Description (描述)。檢閱政策 Summary (摘要) 來查看您的政策所授予的許可。然後選擇 Create policy (建立政策) 來儲存您的工作。

建立 IAM 角色

為 QLDB 日誌匯出工作建立許可政策後，您可以建立 IAM 角色並將政策附加到該角色。

若要建立 QLDB (IAM 主控台) 的服務角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 對於服務或使用案例，請選擇 QLDB，然後選擇 QL DB 使用案例。
5. 選擇下一步。
6. 選取您在先前步驟中建立的原則旁邊的方塊。
7. (選用) 設定許可界限。這是進階功能，可用於服務角色，而不是服務連結的角色。

   a. 開啟 [設定權限界限] 區段，然後選擇 [使用權限界限] 控制最大角色權限。

      IAM 在您的帳戶中包含受 AWS 管政策和客戶管理政策的清單。

   b. 選取用於許可界限的政策。
8. 選擇下一步。
9. 輸入角色名稱或角色名稱尾碼，以協助您識別角色的用途。

   > **Important**
   >
   > 命名角色時，請注意下列事項：
   >
   > - 角色名稱在您的內部必須是唯一的 AWS 帳戶，並且不能根據大小寫將其唯一。
   >
   >   例如，請勿建立同時命名為**PRODROLE**和的角色**prodrole**。當角色名稱用於策略中或作為 ARN 的一部分時，角色名稱會區分大小寫，但是當主控台中的客戶 (例如在登入程序期間) 顯示角色名稱時，角色名稱不區分大小寫。
   > - 您無法在建立角色之後編輯該角色的名稱，因為其他實體可能會參照該角色。
10. (選擇性) 在說明中，輸入角色的說明。
11. (選擇性) 若要編輯角色的使用案例和權限，請在步驟 1：選取信任的實體或步驟 2：新增權限區段中，選擇編輯。

12. (選擇性) 若要協助識別、組織或搜尋角色，請將標籤新增為鍵值配對。如需有關在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的標記 IAM 資源。
13. 檢閱角色，然後選擇 Create role (建立角色)。

下列 JSON 文件是信任政策的範例，該政策允許 QLDB 假設具有附加特定許可的 IAM 角色。

```
{
    "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**
>
> 此信任原則範例顯示如何使用aws:SourceArn和aws:SourceAccount全域條件內容索引鍵來防止混淆的副問題。透過此信任原則，QLDB 只能擔任帳號中任何 QLDB 資源的角色。123456789012
> 如需詳細資訊，請參閱 預防跨服務混淆代理人。

建立 IAM 角色後，返回 QLDB 主控台並重新整理 [建立匯出任務] 頁面，以便找到您的新角色。

分錄匯出的常見錯誤

本節說明 Amazon QLDB 針對日誌匯出請求所擲回的執行階段錯誤。

以下是由服務返回的常見異常的列表。每個例外都包含特定的錯誤訊息，後面接著簡短的描述和可能解決方案的建議。

AccessDeniedException

###############IAM: PassRole #####roleAR N

您沒有將 IAM 角色傳遞給 QLDB 服務的許可。QLDB 需要所有日誌匯出要求的角色，而且您必須擁有將此角色傳遞給 QLDB 的權限。此角色為 QLDB 提供指定 Amazon S3 儲存貯體中的寫入許可。

確認您定義的 IAM 政策授與對 QLDB 服務 () `qldb.amazonaws.com` 的指定 IAM 角色資源執行 `PassRole` API 作業的權限。如需政策範例，請參閱「Amazon QLDB 以身分識別為基礎的政策範例」。

IllegalArgumentException

###QLDB ## S3 ########errorCode errorMessage

此錯誤的可能原因是提供的 Amazon S3 儲存貯體不存在於 Amazon S3 中。或者，QLDB 沒有足夠的許可將物件寫入指定的 Amazon S3 儲存貯體。

確認您在匯出任務請求中提供的 S3 儲存貯體名稱正確無誤。如需有關儲存貯體命名的詳細資訊，請參閱 Amazon 簡單儲存服務使用者指南中的儲存貯體限制和限制。

此外，請確認您是否為指定儲存貯體定義政策，以`PutObjectAcl`授`PutObject`與 QLDB 服務 () `qldb.amazonaws.com`。如需進一步了解，請參閱匯出權限。

IllegalArgumentException

訊息：驗證 S3 組態時，來自 Amazon S3 的意外回應。*S3 ####errorCode errorMessage*

嘗試將日誌匯出資料寫入提供的 S3 儲存貯體失敗，並顯示提供的 Amazon S3 錯誤回應。如需有關可能原因的詳細資訊，請參閱 Amazon 簡單儲存服務使用者指南中的疑難排解 Amazon S3。

IllegalArgumentException

訊息：Amazon S3 儲存貯體前綴不得超過 128 個字元

分錄匯出要求中提供的字首超過 128 個字元。

IllegalArgumentException

訊息：開始日期不得大於結束日期

`InclusiveStartTime`和`ExclusiveEndTime`必須使用 ISO 8601 日期和時間格式，以及國際標準時間 (UTC)。

IllegalArgumentException

訊息：結束日期不可為 future

`InclusiveStartTime`和`ExclusiveEndTime`必須使用`ISO 8601`日期和時間格式以及 UTC。

IllegalArgumentException

訊息：提供的物件加密設定 (S3EncryptionConfiguration) 與 AWS Key Management Service (AWS KMS) 金鑰不相容

您提`KMSKeyArn`供了一個`ObjectEncryptionType`NO_ENCRYPTION或`SSE_S3`。您只能提供 AWS KMS key 針對物件加密類型所管理的客戶`SSE_KMS`。若要進一步了解 Amazon S3 中的伺服器端加密選項，請參閱 Amazon S3 開發人員指南中的使用伺服器端加密保護資料。

LimitExceededException

訊息：已超過 2 個同時執行之分錄匯出工作的限制

QLDB 會強制執行兩個並行分錄匯出工作的預設限制。

從 Amazon QLDB 串流日誌資料

Amazon QLDB 使用不可變的交易日誌 (稱為日誌) 來儲存資料。日誌會追蹤已提交資料的每項變更，並維護一段時間內完整且可驗證的變更歷程記錄。

您可以在 QLDB 中建立串流，擷取提交給日誌的每個文件修訂版本，並以近乎即時的方式將這些資料傳送到 Amazon Kinesis 資料串流。QLDB 串流是從總帳日誌傳送到 Kinesis 資料串流資源的連續資料流。

然後，您可以使用 Kinesis 串流平台或 Kinesis 用戶端程式庫來使用串流、處理資料記錄以及分析資料內容。QLDB 串流會以三種記錄類型將您的資料寫入 Kinesis 資料串流：控制、區塊摘要和修訂詳細資料。如需詳細資訊，請參閱 Kinesis 中的 QLDB 串流記錄。

主題

常用案例

串流可讓您使用 QLDB 做為單一可驗證的事實來源，同時將日誌資料與其他服務整合。以下是 QLDB 日誌串流支援的一些常見使用案例：

- 事件驅動架構 — 使用解耦元件以事件驅動架構樣式建置應用程式。例如，銀行可以使用 AWS Lambda 函數來實施通知系統，當客戶的帳戶餘額降至閾值以下時，會向客戶發出警報。在這樣的系統中，科目餘額維護在 QLDB 分類帳中，任何餘額變更都會記錄在分錄中。此 AWS Lambda 函數可在使用已提交至日誌並傳送至 Kinesis 資料串流的平衡更新事件時觸發通知邏輯。

- 即時分析 — 建置 Kinesis 消費者應用程式，以針對事件資料執行即時分析。透過這項功能，您可以近乎即時地獲得洞察，並快速回應不斷變化的商業環境。例如，電子商務網站可以分析產品銷售數據，並在銷售達到限制後立即停止折扣產品的廣告。
- 歷史分析 — 透過重播歷史事件資料，充分利用 Amazon QLDB 的日誌導向架構。您可以選擇從過去的任何時間點開始 QLDB 串流，其中自該時間以來的所有修訂都會傳送至 Kinesis Data Streams。使用此功能，您可以建置在歷史資料上執行分析任務的 Kinesis 消費者應用程式。例如，電子商務網站可以根據需要運行分析，以生成以前未捕獲的過去銷售指標。
- 複寫至特定目的資料庫 — 使用 QLDB 日誌串流，將 QLDB 分類帳 Connect 至其他專用資料存放區。例如，使用 Kinesis 串流資料平台與 Amazon OpenSearch 服務整合，該平台可為 QLDB 文件提供全文搜尋功能。您也可以建置自訂 Kinesis 消費者應用程式，將日誌資料複寫到其他提供不同具體化視觀表的專用資料庫。例如，將關聯式資料複寫到 Amazon Aurora，或複寫到 Amazon Neptune 以取得以圖形為基礎的資料。

使用您的直播

使用 Kinesis Data Streams 持續使用、處理和分析大型資料串流記錄。除了 Kinesis Data Streams 之外，Kinesis 串流資料平台還包括 Amazon 資料 Firehose 和 Amazon Apache Flink 管理服務。您可以使用此平台將資料記錄直接傳送到 Amazon OpenSearch 服務、Amazon Redshift、Amazon S3 或 Splunk 等服務。如需詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南中的 Kinesis 資料串流取用者。

您也可以使用 Kinesis 用戶端程式庫 (KCL) 建置串流取用者應用程式，以自訂方式處理資料記錄。KCL 會在低階 Kinesis Data Streams API 上提供有用的抽象，可以簡化程式碼。若要進一步了解 KCL，請參閱 Amazon Kinesis 資料串流開發人員指南中的使用 Kinesis 用戶端程式庫。

交貨保證

QLDB 串流提供at-least-once交付保證。QLDB 串流產生的每個資料記錄至少會傳送至 Kinesis Data Streams 一次。相同的記錄可以多次出現在 Kinesis 資料串流中。因此，如果您的用例需要，您必須在消費者應用程序層中具有重複數據刪除邏輯。

也沒有訂購保證。在某些情況下，QLDB 區塊和修訂可能會依序在 Kinesis 資料串流中產生。如需詳細資訊，請參閱 處理重複和 out-of-order記錄。

傳遞延遲考量

QLDB 串流通常會以近乎即時的速度提供 Kinesis Data Streams 的更新。不過，下列案例可能會在新認可的 QLDB 資料傳送至 Kinesis 資料串流之前，產生額外的延遲：

- Kinesis 可以限制從 QLDB 串流處理的資料，具體取決於您的 Kinesis Data Streams 佈建。例如，如果您有多個 QLDB 串流寫入單一 Kinesis 資料串流，且 QLDB 的要求速率超過 Kinesis 串流資源的容量，就可能會發生這種情況。如果輸送量在不到 15 分鐘的時間內增長到先前峰值的兩倍以上，則使用隨需佈建時，Kinesis 中的節流也可能發生。

  您可以透過監控 Kinesis 指標`WriteProvisionedThroughputExceeded`來測量此超出輸送量。如需詳細資訊和可能的解決方案，請參閱如何疑難排解 Kinesis 資料串流中的節流錯誤？ 。
- 使用 QLDB 串流，您可以建立不限期的串流，其中包含過去的開始日期和時間，而且沒有結束日期和時間。根據設計，只有在指定開始日期和時間的所有先前資料順利交付之後，QLDB 才會開始將新認可的資料傳送至 Kinesis Data Streams。如果您在此情況下感覺到額外的延遲，您可能需要等待先前的資料傳遞，或者您可以從稍後的開始日期和時間開始串流。

開始使用串流

以下是開始將日誌資料串流到 Kinesis 資料串流所需步驟的高階概觀：

1. 建立 Kinesis Data Streams 資源。如需指示，請參閱 Amazon Kinesis 資料串流開發人員指南中的建立和更新資料串流。
2. 建立可讓 QLDB 假設 Kinesis 資料串流寫入權限的 IAM 角色。如需說明，請參閱QLDB 中的串流權限。
3. 建立 QLDB 日誌串流。如需說明，請參閱在 QLDB 中建立和管理串流。
4. 使用 Kinesis 資料串流，如上一節使用您的直播所述。如需示範如何使用 Kinesis 用戶端程式庫的程式碼範例 AWS Lambda，請參閱使用 QLDB 中的串流進行開發。

在 QLDB 中建立和管理串流

Amazon QLDB 提供 API 操作，可建立和管理從分類帳到 Amazon Kinesis 資料串流的日誌資料串流。QLDB 串流會擷取提交至您期刊的每個文件修訂版，並將其傳送至 Kinesis 資料串流。

您可以使用 AWS Management Console、 AWS SDK 或 AWS Command Line Interface (AWS CLI) 來建立日誌串流。此外，您也可以使用AWS CloudFormation範本建立串流。若要取得更多資訊，請參閱《AWS CloudFormation 使用指南》中的AWS::QLDB::Stream資源。

主題

串流參數

若要建立 QLDB 日誌串流，您必須提供下列組態參數：

分類帳名稱

您要將其日誌資料串流至 Kinesis Data Streams 的 QLDB 分類帳。

串流名稱

您要指派給 QLDB 日誌串流的名稱。使用者定義的名稱可協助識別和指示串流用途。

對於特定分類帳，您的串流名稱在其他作用中串流間必須是唯一的。如中所定義，串流名稱與分類帳名稱具有相同的命名限制亞馬遜 QLDB 中的配額和限制。

除了串流名稱之外，QLDB 會為您建立的每個 QLDB 串流指派一個串流 ID。串流 ID 在指定分類帳的所有串流中是唯一的，無論其狀態為何。

開始日期和時間

開始串流日誌資料的日期和時間。此值可以是過去的任何日期和時間，但不能在 future。

結束日期和時間

(選擇性) 指定串流結束時間的日期和時間。

如果您建立沒有結束時間的無限期串流，則必須手動取消串流才能結束串流。您也可以取消尚未到達指定結束日期和時間的活動有限串流。

目的地 Kinesis 資料串流

Kinesis 資料串流目標資源，串流將資料記錄寫入的目標資源。若要了解如何建立 Kinesis 資料串流，請參閱 Amazon Kinesis 資料串流開發人員指南中的建立和更新資料串流。

> **Important**
>
> - 不支援跨區域和跨帳戶串流。指定的 Kinesis 資料串流必須 AWS 區域 與分類帳在同一個帳戶中。
> - Kinesis Data Streams 中的記錄彙總預設為啟用。此選項可讓 QLDB 在單一 Kinesis 資料串流記錄中發佈多個資料記錄，從而增加每個 API 呼叫傳送的記錄數量。
>
>   記錄彙總對於處理記錄具有重要意義，並且需要在串流取用者中進行解除彙總。若要進一步了解，請參閱 Amazon Kinesis Data Streams 開發人員指南中的 KPL 關鍵概念和消費者去彙總。

IAM 角色

允許 QLDB 承擔 Kinesis 資料串流寫入許可的 IAM 角色。您可以使用 QLDB 主控台自動建立此角色，也可以在 IAM 中手動建立此角色。若要瞭解如何手動建立它，請參閱串流權限。

若要在請求日誌串流時將角色傳遞至 QLDB，則您必須擁有針對 IAM 角色資源執行 `iam:PassRole` 動作的許可。

串流 ARN

每個 QLDB 日誌串流都是分類帳的子資源，並以 Amazon 資源名稱 (ARN) 唯一識別。以下是 QLDB 串流的 ARN 範例，其資料流識別碼為名`IiPT4brpZCqCq3f4MTHbYy`為的分類帳。`exampleLedger`

```
arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/IiPT4brpZCqCq3f4MTHbYy
```

下一節說明如何使用建立和取消 QLDB 串流。 AWS Management Console

AWS Management Console

請依照下列步驟使用 QLDB 主控台建立或取消 QLDB 串流。

若要建立串流 (主控台)

1. 登入 AWS Management Console，然後開啟 Amazon QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在導覽窗格中，選擇 Data Streams (資料串流)。
3. 選擇 [建立 QLDB 串流]。
4. 在 [建立 QLDB 串流] 頁面上，輸入下列設定：

   - 串流名稱 — 您要指派給 QLDB 串流的名稱。
   - 分類帳 — 您要串流其分錄資料的分類帳。
   - 開始日期和時間 — 以國際標準時間 (UTC) 表示的包含時間戳記，從該時間戳記開始串流日誌資料。此時間戳記預設為目前的日期和時間。它不能在 future，而且必須早於結束日期和時間。
   - 結束日期和時間 — (選用) 獨佔時間戳記 (UTC)，用於指定串流結束時間。如果將此參數保持空白，則串流會無限期執行，直到您取消為止。
   - 目標串流 — Kinesis Data Stre ams 目標資源，您的串流將資料記錄寫入的目標資源。請使用下列 ARN 格式。

     ```
     arn:aws:kinesis:aws-region:account-id:stream/kinesis-stream-name
     ```

     以下是範例。

     ```
     arn:aws:kinesis:us-east-1:123456789012:stream/stream-for-qldb
     ```

     不支援跨區域和跨帳戶串流。指定的 Kinesis 資料串流必須 AWS 區域 與分類帳在同一個帳戶中。
   - 在 Kinesis Data Streams 中啟用記錄彙總 — (預設為啟用) 可讓 QLDB 在單一 Kinesis 資料串流記錄中發佈多個資料記錄，從而增加每次 API 呼叫傳送的記錄數量。
   - 服務存取權 — 將 QLDB 寫入權限授與 Kinesis 資料串流的 IAM 角色。

     若要在請求日誌串流時將角色傳遞給 QLDB，您必須擁有對 IAM 角色資源執行*iam:PassRole*動作的權限。

     - 建立並使用新的服務角色 — 讓主控台為您建立具有指定 Kinesis 資料串流所需權限的新角色。
     - 使用現有的服務角色 — 若要了解如何在 IAM 中手動建立此角色，請參閱串流權限。

- 標籤 — (選用) 將標籤附加為索引鍵值配對，以將中繼資料新增至串流。您可以在直播中新增標記，以協助整理和識別它們。如需詳細資訊，請參閱 標記 Amazon QLDB 資源。

  選擇 [新增標籤]，然後視需要輸入任何索引鍵值配對。

5. 如果設定符合您的需求，請選擇 [建立 QLDB 串流]。

   如果您的要求提交成功，主控台會返回「串流」主頁面，並列出您的 QLDB 串流及其目前狀態。

6. 串流作用中後，請使用 Kinesis 透過消費者應用程式處理串流資料。

   開啟 Kinesis Data Streams 主控台，網址為 https://console.aws.amazon.com/kinesis/。

如需有關串流資料記錄格式的資訊，請參閱Kinesis 中的 QLDB 串流記錄。

若要瞭解如何處理導致錯誤的串流，請參閱處理受損的流。

若要取消串流 (主控台)

取消 QLDB 串流後，您無法重新啟動它。若要繼續將資料傳遞至 Kinesis Data Streams，您可以建立新的 QLDB 串流。

1. 開啟 Amazon QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在導覽窗格中，選擇 Data Streams (資料串流)。
3. 在 QLDB 串流清單中，選取要取消的作用中串流。
4. 選擇 [取消串流]。在提供的方塊**`cancel stream`**中輸入以確認此操作。

如需 AWS CLI 將 QLDB API 與 AWS SDK 搭配使用，或建立和管理日誌串流的相關資訊，請參閱。使用 QLDB 中的串流進行開發

流狀態

QLDB 串流的狀態可以是下列其中一種：

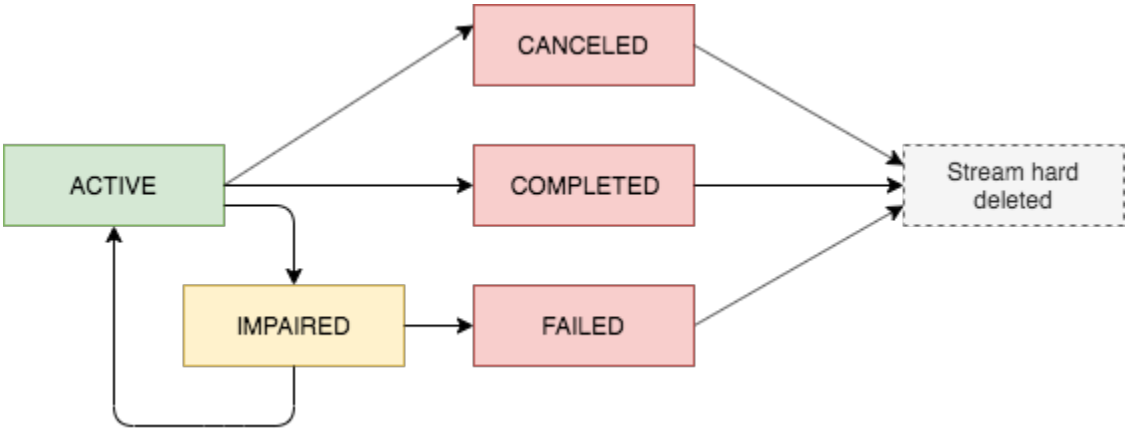
- `ACTIVE`— 目前正在流式傳輸或等待流數據（對於沒有結束時間的無限期流）。
- `COMPLETED`— 已成功完成在指定時間範圍內的所有日誌區塊串流。這是一個終端狀態。
- `CANCELED`— 在指定的結束時間之前由使用者要求結束，且不再主動串流資料。這是一個終端狀態。

- IMPAIRED— 由於發生需要您採取動作的錯誤，因此無法將記錄寫入 Kinesis。這是一個可恢復的非終端狀態。

  如果您在一小時內解決錯誤，串流會自動移至ACTIVE狀態。如果錯誤在一小時後仍未解決，串流會自動移至FAILED狀態。

- FAILED— 由於錯誤而無法將記錄寫入 Kinesis，且處於無法復原的終止狀態。

下圖說明 QLDB 串流資源如何在狀態之間轉換。



終端機串流的到期日

處於終止狀態 (CANCELEDCOMPLETED、和FAILED) 的串流資源會受到 7 天的保留期限。在此限制到期後，系統會自動將它們硬刪除。

刪除終端機串流後，您將無法再使用 QLDB 主控台或 QLDB API 來描述或列出串流資源。

處理受損的流

如果您的串流遇到錯誤，它會先移至IMPAIRED狀態。QLDB 會持續重試IMPAIRED串流長達一小時。

如果您在一小時內解決錯誤，串流會自動移至ACTIVE狀態。如果錯誤在一小時後仍未解決，串流會自動移至FAILED狀態。

受損或失敗的串流可能會造成下列其中一個錯誤原因：

- KINESIS_STREAM_NOT_FOUND— 目的地 Kinesis Data Streams 資源不存在。確認您在 QLDB 串流要求中提供的 Kinesis 資料串流是否正確。然後，移至 Kinesis 並建立您指定的資料串流。

- `IAM_PERMISSION_REVOKED`— QLDB 沒有足夠的權限將資料記錄寫入指定的 Kinesis 資料串流。確認您是否為指定的 Kinesis 資料串流定義原則，以授與下列動作的 QLDB 服務 (`qldb.amazonaws.com`) 權限：
  - `kinesis:PutRecord`
  - `kinesis:PutRecords`
  - `kinesis:DescribeStream`
  - `kinesis:ListShards`

監控受損的流

如果串流受損，QLDB 主控台會顯示橫幅，其中會顯示有關串流及其所遇到錯誤的詳細資料。您也可以使用 `DescribeJournalKinesisStream` API 操作來獲取流的狀態和基本錯誤原因。

此外，您可以使 CloudWatch 用 Amazon 建立監控串流指`IsImpaired`標的警示。如需使用監督 QLDB 測量結果的相關資訊 CloudWatch，請參閱。Amazon QLDB 維度和指標

使用 QLDB 中的串流進行開發

本節概述了您可以與 AWS 開發套件搭配使用的 API 操作，或在 AWS CLI Amazon QLDB 中建立和管理日誌串流。同時也說明示範這些作業並使用 Kinesis 用戶端程式庫 (KCL) 或實作串流取用者的範例應 AWS Lambda 用程式。

您可以使用 KCL 為 Amazon Kinesis Data Streams 建置消費者應用程式。KCL 會在低階 Kinesis Data Streams API 上提供有用的抽象，可以簡化程式碼。若要進一步了解 KCL，請參閱 Amazon Kinesis 資料串流開發人員指南中的使用 Kinesis 用戶端程式庫。

內容

QLDB 日誌串流 API

QLDB API 提供下列日誌串流作業，供應用程式使用：

- `StreamJournalToKinesis`— 為指定的 QLDB 分類帳建立日誌資料流。串流會擷取提交至分類帳期刊的每個文件修訂版，並將資料傳送至指定的 Kinesis Data Streams 資源。
  - Kinesis Data Streams 中的記錄彙總預設為啟用。此選項可讓 QLDB 在單一 Kinesis 資料串流記錄中發佈多個資料記錄，從而增加每個 API 呼叫傳送的記錄數量。

    記錄彙總對於處理記錄具有重要意義，並且需要在串流取用者中進行解除彙總。若要進一步了解，請參閱 Amazon Kinesis Data Streams 開發人員指南中的 KPL 關鍵概念和消費者去彙總。
- `DescribeJournalKinesisStream`— 傳回有關指定 QLDB 日誌串流的詳細資訊。輸出包括 ARN、串流名稱、目前狀態、建立時間，以及原始串流建立要求的參數。
- `ListJournalKinesisStreamsForLedger`— 傳回指定分類帳之所有 QLDB 日誌串流描述子的清單。每個串流描述元的輸出都包含傳回的相同詳細資料`DescribeJournalKinesisStream`。
- `CancelJournalKinesisStream`— 結束指定的 QLDB 日誌串流。在可以取消流之前，其當前狀態必須是`ACTIVE`。

  取消串流後，您無法重新啟動該串流。若要繼續將資料傳遞至 Kinesis Data Streams，您可以建立新的 QLDB 串流。

如需這些 API 作業的完整說明，請參閱Amazon QLDB API 參考參考.

如需使用建立和管理日誌串流的相關資訊 AWS CLI，請參閱命AWS CLI 令參考。

範例應用程式

QLDB 提供範例應用程式，示範使用日誌串流的各種作業。這些應用程式是AWS 範例 GitHub 網站上的開放原始碼。

主題

- 基本操作
- 與 OpenSearch 服務 Python 合
- 與 Amazon SNS 和 Amazon SQS（Python）集成

基本操作

如需示範 QLDB 日誌串流基本作業的 Java 程式碼範例，請參閱 GitHub 儲存庫 aw amazon-qldb-dmv-sample s-範例 /-java。如需如何下載和安裝此範例應用程式的指示，請參閱安裝亞馬遜 QLDB Java 範例應用程式。

> **Note**
>
> 安裝應用程式後，請勿繼續執行 Java 教學課程的步驟 1 來建立分類帳。此用於流式傳輸的示例應用程序為您創建`vehicle-registration`分類帳。

此範例應用程式會封裝來自Java 教學及其相依性的完整原始程式碼，包括下列模組：

- AWS SDK for Java— 建立和刪除 QLDB 和 Kinesis Data Streams 資源，包括分類帳、QLDB 日誌串流和 Kinesis 資料串流。
- 用於 Java 的亞馬遜 QLDB 驅動程序— 使用 PartiQL 陳述式在分類帳上執行資料交易，包括建立表格和插入文件。
- Kinesis 用戶端程式庫 — 使用和處理 Kinesis 資料串流中的資料。

執行程式碼

此StreamJournal類別包含示範下列作業的教學課程程式碼：

1. 建立名為的分類帳`vehicle-registration`、建立表格，並使用範例資料載入表格。

   > **Note**
   >
   > 在執行此程式碼之前，請確定您尚未擁有名為的作用中總帳`vehicle-registration`。

2. 建立 Kinesis 資料串流、允許 QLDB 假設 Kinesis 資料串流寫入權限的 IAM 角色，以及 QLDB 日誌串流。
3. 使用 KCL 啟動處理 Kinesis 資料串流的串流讀取器，並記錄每個 QLDB 資料記錄。
4. 使用串流資料來驗證`vehicle-registration`範例分類帳的雜湊鏈。
5. 停止串流讀取器、取消 QLDB 日誌串流、刪除分類帳以及刪除 Kinesis 資料串流，以清理所有資源。

要運行`StreamJournal`教程代碼，請從項目根目錄輸入以下 Gradle 命令。

```
./gradlew run -Dtutorial=streams.StreamJournal
```

與 OpenSearch 服務 Python 合

如需示範如何將 QLDB 串流與 Amazon OpenSearch 服務整合的 Python 範例應用程式，請參閱 GitHub 儲存庫 aw amazon-qldb-streaming-amazon s-範例/-。opensearch-service-sample-python此應用程式使用 AWS Lambda 函數來實作 Kinesis Data Streams 取用者。

若要複製存放庫，請輸入下列git命令。

```
git clone https://github.com/aws-samples/amazon-qldb-streaming-amazon-opensearch-
service-sample-python.git
```

若要執行範例應用程式，請參閱的 README 中 GitHub 的說明。

與 Amazon SNS 和 Amazon SQS（Python）集成

如需示範如何將 QLDB 串流與亞馬遜簡單通知服務 (Amazon SNS) 整合的 Python 範例應用程式，請參閱 GitHub 儲存庫 aw amazon-qldb-streams-dmv s-範例/-。sample-lambda-python

此應用程式使用 AWS Lambda 函數來實作 Kinesis Data Streams 取用者。它將消息發送到 Amazon SNS 主題，該主題已訂閱 Amazon Simple Queue Service (Amazon SQS) 隊列。

若要複製存放庫，請輸入下列git命令。

```
git clone https://github.com/aws-samples/amazon-qldb-streams-dmv-sample-lambda-
python.git
```

若要執行範例應用程式，請參閱的 README 中 GitHub 的說明。

Kinesis 中的 QLDB 串流記錄

Amazon QLDB 串流會將三種類型的資料記錄寫入指定的 Amazon Kinesis Data Streams 資源：控制、區塊摘要和修訂詳細資料。所有三種記錄類型都是以 Amazon Ion 格式的二進位表示法編寫。

控制記錄表示 QLDB 串流的開始和完成。每當修訂送至您的日誌時，QLDB 串流會將所有相關聯的日誌區塊資料寫入區塊摘要和修訂詳細資料記錄中。

這三種記錄類型是多態的。它們都包含包含 QLDB 串流 ARN、記錄類型和記錄裝載的通用頂層記錄。此最上層記錄的格式如下。

```
{
  qldbStreamArn: string,
```

```
  recordType: string,
  payload: {
    //control | block summary | revision details record
  }
}
```

該recordType字段可以有三個值之一：

- CONTROL
- BLOCK_SUMMARY
- REVISION_DETAILS

下列各節說明每個個別承載資料記錄的格式和內容。

> **Note**
>
> QLDB 會以 Amazon 離子的二進位表示法，將所有串流記錄寫入 Kinesis 資料串流。Ion 的文本表示中提供了以下實例，以可讀格式說明記錄內容。

主題

控制記錄

一個 QLDB 流寫入控制記錄，以指示其開始和完成事件。以下是每個控制記錄的範例及範例資料的範例controlRecordType：

- CREATED— QLDB 串流寫入 Kinesis 以指出您新建立的串流處於作用中狀態的第一筆記錄。

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
```

```
    controlRecordType:"CREATED"
  }
}
```

- COMPLETED— QLDB 串流寫入 Kinesis 以指出串流已到達指定結束日期和時間的最後一筆記錄。如果您取消串流，則不會寫入此記錄。

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
    controlRecordType:"COMPLETED"
  }
}
```

封鎖摘要記錄

區塊摘要記錄代表已確認文件修訂的分錄區塊。區塊是在交易期間認可到 QLDB 日誌的物件。

區塊摘要記錄的承載包含認可區塊之交易的區塊位址、時間戳記和其他中繼資料。它也包括區塊中修訂的摘要屬性，以及提交修訂版本的 PartiQL 陳述式。以下是包含範例資料的區塊摘要記錄範例。

> **Note**
>
> 此區塊摘要範例僅供參考。顯示的哈希值不是實際計算的哈希值。

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

在revisionSummaries欄位中，某些修訂可能沒有documentId. 這些是不包含使用者資料的僅限內部系統修訂。一個 QLDB 流包括這些修訂在其各自的塊摘要記錄，因為這些修訂的哈希是該日誌的完整哈希鏈的一部分。密碼編譯驗證需要完整的雜湊鏈。

只有具有文件 ID 的修訂版本才會發佈在個別的修訂版本詳細資訊記錄中，如下一節所述。

修訂明細記錄

修訂明細記錄代表已確認至分錄的文件修訂。有效負載包含修訂版本認可檢視中的所有屬性，以及關聯的資料表名稱和表格 ID。以下是具有範例資料的修訂記錄範例。

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

處理重複和 out-of-order記錄

QLDB 串流可以將重複資料和 out-of-order 記錄發佈到 Kinesis Data Streams。因此，消費者應用程序可能需要實現自己的邏輯來識別和處理這種情況。區塊摘要與修訂明細記錄包含可用於此目的的的欄位。結合下游服務的功能，這些欄位可以指出唯一識別和記錄的嚴格順序。

例如，假設將 QLDB 與索引整合的串流，以提供文件 OpenSearch 的全文檢索搜尋功能。在此使用案例中，您需要避免索引文件的過時 (out-of-order) 修訂版本。若要強制執行排序與重複資料刪除，您可以使用中的文件 ID 和外部版本欄位 OpenSearch，以及修訂明細記錄中的文件 ID 和版本欄位。

如需將 QLDB 與 Amazon OpenSearch 服務整合的範例應用程式中重複資料刪除邏輯的範例範例，請參閱 GitHub 儲存庫 aws-範例/-。amazon-qldb-streaming-amazon opensearch-service-sample-python

QLDB 中的串流權限

在建立 Amazon QLDB 串流之前，您必須向 QLDB 提供指定 Amazon Kinesis 資料串流資源的寫入許可。如果您使用 AWS KMS key 針對 Kinesis 串流伺服器端加密而管理的客戶，您還必須向 QLDB 提供使用指定對稱加密金鑰的權限。Kinesis Data Streams 不支援非對稱 KMS 金鑰。

若要為您的 QLDB 串流提供必要的許可，您可以讓 QLDB 擔任具有適當許可政策的 IAM 服務角色。服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

> **Note**
>
> 若要在請求日誌串流時將角色傳遞至 QLDB，則您必須擁有針對 IAM 角色資源執行 `iam:PassRole` 動作的許可。這是除了 QLDB 串流子資源的`qldb:StreamJournalToKinesis`權限之外。

要了解如何使用 IAM 控制對 QLDB 的存取，請參閱。Amazon QLDB 如何與 IAM 合作如需 QLDB 原則範例，請參閱。Amazon QLDB 以身分識別為基礎的政策範例

在此範例中，您會建立一個角色，讓 QLDB 代表您將資料記錄寫入 Kinesis 資料串流。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

如果您 AWS 帳戶 是第一次串流 QLDB 日誌，則必須先執行下列動作以建立具有適當政策的 IAM 角色。或者，您可以使用 QLDB 主控台自動為您建立角色。否則，您可以選擇先前建立的角色。

主題

建立許可政策

完成下列步驟，即可建立 QLDB 串流的權限原則。此範例顯示 Kinesis Data Streams 政策，該政策授與 QLDB 權限，以將資料記錄寫入指定的 Kinesis 資料串流。如果適用，此範例也會顯示允許 QLDB 使用對稱加密 KMS 金鑰的金鑰原則。

如需 Kinesis Data Streams 政策的詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南中的使用 IAM 和許可來控制 Amazon Kinesis Data Streams 資源的存取權限以使用使用者產生的 KMS 金鑰。若要進一步了解關 AWS KMS 鍵原則，請參閱AWS Key Management Service 開發人員指南 AWS KMS中的使用金鑰政策。

> **Note**
>
> 您的 Kinesis 資料串流和 KMS 金鑰必須與您的 QLDB 分類帳位於相同 AWS 區域 的帳戶中。

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側的導覽欄中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。

3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 選擇 JSON 標籤。
5. 輸入 JSON 政策文件。
   - 如果您使用客戶管理的 KMS 金鑰來進行 Kinesis 串流的伺服器端加密，請使用下列範例政策文件。*############################ 1#123456789012 # 1234 ####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"
        },
        {
            "Sid": "QLDBStreamKMSPermission",
            "Action": [ "kms:GenerateDataKey" ],
            "Effect": "Allow",
            "Resource": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

   - 否則，請使用下列範例原則文件。若要使用此原則，請以您自己的資訊取代 *us-east-1*、*123456789012*，並在此範例中。*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. 選擇檢閱政策。

> **Note**
>
> 您可以隨時切換 Visual editor (視覺化編輯器) 與 JSON 標籤。不過，如果您進行更改或在 Visual editor (視覺編輯工具) 索引標籤中選擇 Review policy (檢閱政策)，IAM 可能會調整您的政策結構以針對視覺編輯工具進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在 Review policy (檢閱政策) 頁面上，為您正在建立的政策輸入選用的 Name (名稱) 與 Description (描述)。檢閱政策 Summary (摘要) 來查看您的政策所授予的許可。然後選擇 Create policy (建立政策) 來儲存您的工作。

建立 IAM 角色

為 QLDB 串流建立許可政策後，您可以建立 IAM 角色並將政策附加到該角色。

若要建立 QLDB (IAM 主控台) 的服務角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 對於服務或使用案例，請選擇 QLDB，然後選擇 QL DB 使用案例。
5. 選擇下一步。
6. 選取您在先前步驟中建立的原則旁邊的方塊。
7. (選用) 設定許可界限。這是進階功能，可用於服務角色，而不是服務連結的角色。
   a. 開啟 [設定權限界限] 區段，然後選擇 [使用權限界限] 控制最大角色權限。

      IAM 在您的帳戶中包含受 AWS 管政策和客戶管理政策的清單。
   b. 選取用於許可界限的政策。
8. 選擇下一步。

9. 輸入角色名稱或角色名稱尾碼，以協助您識別角色的用途。

> **Important**
>
> 命名角色時，請注意下列事項：
>
> - 角色名稱在您的內部必須是唯一的 AWS 帳戶，並且不能根據大小寫將其唯一。
>
> 例如，請勿建立同時命名為**PRODROLE**和的角色**prodrole**。當角色名稱用於策略中或作為 ARN 的一部分時，角色名稱會區分大小寫，但是當主控台中的客戶 (例如在登入程序期間) 顯示角色名稱時，角色名稱不區分大小寫。
> - 您無法在建立角色之後編輯該角色的名稱，因為其他實體可能會參照該角色。

10. (選擇性) 在說明中，輸入角色的說明。
11. (選擇性) 若要編輯角色的使用案例和權限，請在步驟 1：選取信任的實體或步驟 2：新增權限區段中，選擇編輯。
12. (選擇性) 若要協助識別、組織或搜尋角色，請將標籤新增為鍵值配對。如需有關在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的標記 IAM 資源。
13. 檢閱角色，然後選擇 Create role (建立角色)。

下列 JSON 文件是信任政策的範例，該政策允許 QLDB 假設具有附加特定許可的 IAM 角色。

```
{
    "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**
>
> 此信任原則範例顯示如何使用aws:SourceArn和aws:SourceAccount全域條件內容索引鍵來防止混淆的副問題。透過此信任原則，QLDB 只能擔任分類帳帳戶123456789012中任何 QLDB 串流的角色。myExampleLedger
> 如需詳細資訊，請參閱 預防跨服務混淆代理人。

建立 IAM 角色後，返回 QLDB 主控台並重新整理「建立 QLDB 串流」頁面，以便找到您的新角色。

QLDB 中日誌串流的常見錯誤

本節說明 Amazon QLDB 針對日誌串流請求所擲回的執行階段錯誤。

以下是由服務返回的常見異常的列表。每個例外都包含特定的錯誤訊息，後面接著簡短的描述和可能解決方案的建議。

AccessDeniedException

###############IAM: PassRole #####roleAR N

您沒有將 IAM 角色傳遞給 QLDB 服務的許可。QLDB 需要所有日誌串流要求的角色，而且您必須擁有將此角色傳遞給 QLDB 的權限。此角色為 QLDB 提供指定 Amazon Kinesis 資料串流資源中的寫入許可。

確認您定義的 IAM 政策授與對 QLDB 服務 () qldb.amazonaws.com 的指定 IAM 角色資源執行 PassRole API 作業的權限。如需政策範例，請參閱「Amazon QLDB 以身分識別為基礎的政策範例」。

IllegalArgumentException

###QLDB ### Kinesis Data Streams ######## Kinesis ####errorCode errorMessage

此錯誤的可能原因是提供的 Kinesis Data Streams 資源不存在。或者，QLDB 沒有足夠的權限將資料記錄寫入指定的 Kinesis 資料串流。

確認您在串流請求中提供的 Kinesis 資料串流是否正確。如需詳細資訊，請參閱 Amazon Kinesis 資料串流開發人員指南中的建立和更新資料串流。

此外，請確認您是否為指定的 Kinesis 資料串流定義原則，以授與下列動作的 QLDB 服務 (`qldb.amazonaws.com`) 權限。如需詳細資訊，請參閱 串流權限。

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

訊息：在驗證 Kinesis 組態時，Kinesis 資料串流產生非預期的回應。*`## Kinesis ## ##errorCode errorMessage`*

嘗試將資料記錄寫入提供的 Kinesis 資料串流失敗，並顯示提供的 Kinesis 錯誤回應。如需有關可能原因的詳細資訊，請參閱 Amazon Kinesis Data Streams 開發人員指南中的 Amazon Kinesis Data Streams 生產者疑難排解。

IllegalArgumentException

訊息：開始日期不得大於結束日期。

`InclusiveStartTime`和`ExclusiveEndTime`必須使用 ISO 8601 日期和時間格式，以及國際標準時間 (UTC)。

IllegalArgumentException

訊息：開始日期不可為 future。

`InclusiveStartTime`和`ExclusiveEndTime`必須使用`ISO 8601`日期和時間格式以及 UTC。

LimitExceededException

訊息：已超過 5 個同時執行 Kinesis 資料串流的日誌串流上限

QLDB 會強制執行五個並行日誌串流的預設限制。

亞馬遜 QLDB 中的分類帳管理

本章說明如何使用 QLDB APIAWS Command Line Interface (AWS CLI)，以及如AWS CloudFormation何在 Amazon QLDB 中執行總帳管理操作。

您也可以使用 AWS Management Console來執行這些相同的任務。如需詳細資訊，請參閱 使用主控台存取 Amazon QLDB。

主題

- Amazon QLDB 分類帳的基本操作
- 使用建立亞馬遜 QLDB 資源AWS CloudFormation
- 標記 Amazon QLDB 資源

Amazon QLDB 分類帳的基本操作

您可以使用 QLDB API 或AWS Command Line Interface (AWS CLI) 在 Amazon QLDB 中建立、更新和刪除分類帳。您也可以列出您帳戶中的所有分類帳，或是取得特定分類帳的相關資料。

下列主題提供簡短程式碼範例，其中顯示使用AWS SDK for Java和的分類帳作業的一般步驟AWS CLI。

主題

- 建立分類帳
- 描述分類帳
- 更新分類帳
- 更新分類帳權限模式
- 刪除分類帳
- 清單分類帳

如需在完整範例應用程式中示範這些作業的程式碼範例，請參閱下列開始使用驅動程式教學課程和 GitHub 儲存庫：

- Java: 教程 | GitHub 儲存庫
- Node.js: 教學課程 | GitHub 儲存庫
- 蟒蛇：教程 | GitHub 存儲庫

建立分類帳

使用此作`CreateLedger`業在您的帳戶中建立分類帳AWS 帳戶。您必須提供下列資料：

- 分類帳名稱 — 您要在帳戶中建立分類帳的名稱。在目前分類帳中，名稱必須是唯一的AWS 區域。

  分類帳名稱的命名限制是在亞馬遜 QLDB 中的配額和限制。
- 許可模式 — 要指派給分類帳的許可模式。請選擇下列其中一個選項：
  - 允許全部 — 一種舊版許可模式，可使用 API 層級精細程度啟用對分類帳的存取控制。

    此模式允許具有指定分類帳 SendCommand API 許可的使用者 (因此為 ALLOW_ALL)，在此分類帳中的任何資料表上執行所有 PartiQL 命令。此模式會忽略您為分類帳建立之任何資料表層級或命令層級 IAM 許可政策。
  - STAND@@ ARD — (建議) 一種許可模式，可使用更細的層級啟用對分類帳、資料表和 PartiQL 命令的存取控制。強烈建議您使用此許可模式來最大化分類帳資料的安全性。

    根據預設，此模式會拒絕所有在此分類帳中任何資料表上執行任何 PartiQL 命令的請求。若要允許 PartiQL 命令，除了分類帳的SendCommand API 許可之外，您還必須為特定資料表資源和 PartiQL 動作建立 IAM 許可政策。如需相關資訊，請參閱 開始使用 Amazon QLDB 中的標準許可模式。
- 刪除保護 — (選用) 防止任何使用者刪除總帳的標記。如果您在建立分類帳時未指定，則預設會啟用此功能 (`true`)。

  如果已啟用刪除保護，則必須先停用該功能才能刪除分類帳。您可以使用`UpdateLedger`操作將標記設為來停用它`false`。
- AWS KMS key— (選擇性) inAWS Key Management Service (AWS KMS) 用於靜態資料加密的金鑰。請選擇下列其中一種類型AWS KMS keys：
  - AWS擁有的 KMS 金鑰 — 使用由代表您擁有和管理AWS的 KMS 金鑰。

    如果您在建立分類帳時未定義此參數，則分類帳會使用此型態的索引鍵。您也可以使用字串指`AWS_OWNED_KMS_KEY`定此金鑰類型。此選項不需要額外的設定。
  - 客戶受管 KMS 金鑰：使用在您帳戶中建立、擁有和管理的對稱加密 KMS 金鑰。QLDB 不支援非對稱金鑰。

    此選項需要您建立 KMS 金鑰或使用帳戶中的現有金鑰。如需建立客戶受管金鑰的指示，請參閱AWS Key Management Service開發人員指南中的建立對稱加密 KMS 金鑰。

您可以使用 ID、別名或 Amazon Resource Name (ARN)，指定客戶受管 KMS 金鑰。若要深入了解，請參閱AWS Key Management Service開發人員指南中的金鑰識別碼 (KeyId)。

> **Note**
>
> 不支援跨區域金鑰。指定的 KMS 金鑰必須位於與分類帳AWS 區域相同的。

如需詳細資訊，請參閱 Amazon QLDB 中的靜態加密。

- Tags (選用) 藉由連接標籤作為鍵值對，將中繼資料新增至分類帳。您可以將標籤新增至分類帳，協助整理和識別。如需詳細資訊，請參閱 標記 Amazon QLDB 資源。

分類帳尚未準備就緒，直到 QLDB 建立分類帳並將其狀態設為ACTIVE。

建立分類帳 (Java)

若要使用AWS SDK for Java

1. 建立 AmazonQLDB 類別的執行個體。
2. 建立 CreateLedgerRequest 類別的執行個體，以提供請求資訊。

   您必須提供分類帳名稱和權限模式。
3. 以參數形式提供請求物件，以便執行 createLedger 方法。

createLedger請求會傳回包含分類帳相關資訊的CreateLedgerResult物件。如需在建立分類帳後使用DescribeLedger作業檢查分類帳狀態的範例，請參閱下一節。

下列範例會示範上述步驟。

Example — 使用預設組態設定

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
CreateLedgerRequest request = new CreateLedgerRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
CreateLedgerResult result = client.createLedger(request);
```

> **Note**
>
> 如果未指定分類帳，則會使用下列預設設定：
>
> - 刪除保護 — 已啟用 (`true`)。
> - KMS 金鑰 —AWS 擁有 KMS 金鑰。

Example — 禁用刪除保護，使用客戶託管的 KMS 密鑰，並附加標籤

```
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);
```

創建分類帳（AWS CLI）

使用預設組態設定建`vehicle-registration`立名為的新分類帳。

Example

```
aws qldb create-ledger --name vehicle-registration --permissions-mode STANDARD
```

> **Note**
>
> 如果未指定分類帳，則會使用下列預設設定：
>
> - 刪除保護 — 已啟用 (`true`)。
> - KMS 金鑰 —AWS 擁有 KMS 金鑰。

或者，建立名為停用刪除保護vehicle-registration的新分類帳，並使用指定的客戶受管 KMS 金鑰，並具有指定的標記。

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
```

創建分類帳（AWS CloudFormation）

您也可以使用AWS CloudFormation樣版來建立分類帳。若要取得更多資訊，請參閱《AWS CloudFormation使用指南》中的AWS::QLDB::Ledger資源。

描述分類帳

使用此DescribeLedger操作檢視分類帳的詳細資料。您必須提供分類帳名稱。DescribeLedger中輸出的格式與 CreateLedger 相同。其包含下列資訊：

- 分類帳名稱 — 您要描述分類帳的名稱。
- ARN — 分類帳的 Amazon Resource Name (ARN)，格式為下列格式。

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

- 刪除保護 — 指出刪除保護功能是否已啟用的旗標。
- 建立日期和時間 — 建立分類帳時間的日期和時間 (以紀元時間格式)。
- 狀態 — 分類帳的目前狀態。這可以是下列其中一個值：
  - CREATING
  - ACTIVE
  - DELETING
  - DELETED
- 權限模式 — 指派給分類帳的權限模式。這可以是下列其中一個值：
  - ALLOW_ALL— 一種舊版許可模式，可使用 API 層級精細程度啟用對分類帳的存取控制。

  - STANDARD— 一種許可模式，可使用更細的層級啟用對分類帳、資料表和 PartiQL 命令的存取控制。
- 加密說明 — 分類帳中靜態資料加密的相關資料。這包含下列項目：
  - AWS KMS keyARN — 客戶管理的 KMS 金鑰的 ARN，分類帳用於靜態加密。如果尚未定義，分類帳會使用AWS擁有的 KMS 金鑰進行加密。
  - 加密狀態 — 分類帳的靜態加密目前的狀態。這可以是下列其中一個值：
    - ENABLED— 使用指定的金鑰完全啟用加密。
    - UPDATING— 正在處理指定的金鑰變更。

      QLDB 中的關鍵變更是非同步的。在處理金鑰變更時，可以完全存取分類帳，而不會造成任何效能影響。更新金鑰所需的時間量因分類帳大小而異。
    - KMS_KEY_INACCESSIBLE— 無法存取指定的客戶管理 KMS 金鑰，且分類帳會受損。金鑰已停用或刪除，或是金鑰的授權遭到撤銷。當分類帳受損時，它無法訪問，並且不接受任何讀取或寫入請求。

      在您還原金鑰的授權之後，或重新啟用已停用的金鑰之後，受損的分類帳會自動返回作用中狀態。但是，刪除客戶受管 KMS 金鑰則無法復原。刪除金鑰之後，您就再也無法存取以該金鑰保護的分類帳，而且該資料已無法復原。
  - 無法存取AWS KMS key — 發生錯誤時，KMS 金鑰首次無法存取時的日期和時間 (以紀元時間格式表示)。

    如果 KMS 金鑰可存取，則未定義此選項。

  如需詳細資訊，請參閱 Amazon QLDB 中的靜態加密。

> **Note**
>
> 建立 QLDB 分類帳之後，當其狀態從變更為時，該分類帳CREATING便可供使用ACTIVE。

描述一個分類帳（Java）

若要使用描述分類帳AWS SDK for Java

1. 建立 AmazonQLDB 類別的執行個體。或者，您可以使用您為CreateLedger請求實例化的AmazonQLDB客戶端的相同實例。

2. 建立DescribeLedgerRequest類別的執行個體，並提供您要描述分類帳名稱。
3. 以參數形式提供請求物件，以便執行 describeLedger 方法。
4. describeLedger請求會傳回包含分類帳目前相關資訊的DescribeLedgerResult物件。

下列程式碼範例示範前述步驟。您可以隨時調用客戶端的describeLedger方法以獲取分類帳信息。

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());
```

描述分類帳（AWS CLI）

描述您剛才建立vehicle-registration分類帳。

Example

```
aws qldb describe-ledger --name vehicle-registration
```

更新分類帳

此UpdateLedger作業目前可讓您變更現有分類帳的下列組態設定：

- 刪除保護 — 防止任何使用者刪除總帳的標記。如果已啟用此功能，則必須先將標記設為來停用該功能，false然後才能刪除分類帳。

  如果您未定義此參數，則不會對分類帳的刪除保護設定進行變更。
- AWS KMS key— 中的密鑰AWS Key Management Service（AWS KMS）用於靜態數據的加密。如果您未定義此參數，則不會對分類帳的 KMS 金鑰進行變更。

> **Note**
>
> 亞馬遜 QLDB 於 2021 年 7 月 22 日推出了AWS KMS keys對管理客戶的支援。依預設，在啟動之前建立的任何分類帳都會AWS 擁有的金鑰受到保護，但目前不符合使用客戶管理金鑰進行靜態加密的資格。
> 您可以在 QLDB 主控台上檢視分類帳的建立時間。

使用下列其中一個選項：

- AWS擁有的 KMS 金鑰 — 使用由代表您擁有和管理AWS的 KMS 金鑰。若要使用此類型的金鑰，請指定此`AWS_OWNED_KMS_KEY`參數的字串。此選項不需要額外的設定。
- 客戶受管 KMS 金鑰：使用在您帳戶中建立、擁有和管理的對稱加密 KMS 金鑰。QLDB 不支援非對稱金鑰。

  此選項需要您建立 KMS 金鑰或使用帳戶中的現有金鑰。如需建立客戶受管金鑰的指示，請參閱AWS Key Management Service開發人員指南中的建立對稱加密 KMS 金鑰。

  您可以使用 ID、別名或 Amazon Resource Name (ARN)，指定客戶受管 KMS 金鑰。若要深入了解，請參閱AWS Key Management Service開發人員指南中的金鑰識別碼 (KeyId)。

  > **Note**
  >
  > 不支援跨區域金鑰。指定的 KMS 金鑰必須位於與分類帳AWS 區域相同的。

QLDB 中的關鍵變更是非同步的。在處理金鑰變更時，可以完全存取分類帳，而不會造成任何效能影響。

您可以視需要經常切換金鑰，但更新金鑰所花費的時間長度視分類帳大小而有所不同。您可以使用此`DescribeLedger`操作來檢查靜態加密狀態。

如需詳細資訊，請參閱 Amazon QLDB 中的靜態加密。

`UpdateLedger` 中輸出的格式與 `CreateLedger` 相同。

更新分類帳 (Java)

若要更新分類帳，請使用AWS SDK for Java

1. 建立 AmazonQLDB 類別的執行個體。
2. 建立 UpdateLedgerRequest 類別的執行個體，以提供請求資訊。

   您必須提供分類帳名稱以及用於刪除保護的新布林值或 KMS 金鑰的新字串值。
3. 以參數形式提供請求物件，以便執行 updateLedger 方法。

下列程式碼範例會示範上述步驟。updateLedger請求會傳回已更新分類帳相關資訊的UpdateLedgerResult物件。

Example — 停用刪除保護

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withDeletionProtection(false);
UpdateLedgerResult result = client.updateLedger(request);
```

Example — 使用客戶受管 KMS 金鑰

```
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 — 使用AWS擁有的 KMS 密鑰

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withKmsKey("AWS_OWNED_KMS_KEY")
UpdateLedgerResult result = client.updateLedger(request);
```

更新分類帳 (AWS CLI)

如果您的vehicle-registration分類帳已啟用刪除保護，則必須先停用該功能才能刪除分類帳。

Example

```
aws qldb update-ledger --name vehicle-registration --no-deletion-protection
```

您也可以變更分類帳的靜態加密設定，以使用客戶管理的 KMS 金鑰。

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

或者，您可以將靜態加密設定變更為使用AWS擁有的 KMS 金鑰。

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key AWS_OWNED_KMS_KEY
```

更新分類帳權限模式

此UpdateLedgerPermissionsMode作業可讓您變更現有分類帳的權限模式。請選擇下列其中一個選項：

- 允許全部 — 一種舊版許可模式，可使用 API 層級精細程度啟用對分類帳的存取控制。

  此模式允許具有指定分類帳 SendCommand API 許可的使用者 (因此為 ALLOW_ALL)，在此分類帳中的任何資料表上執行所有 PartiQL 命令。此模式會忽略您為分類帳建立之任何資料表層級或命令層級 IAM 許可政策。
- STAND@@ ARD — (建議) 一種許可模式，可使用更細的層級啟用對分類帳、資料表和 PartiQL 命令的存取控制。強烈建議您使用此許可模式來最大化分類帳資料的安全性。

  根據預設，此模式會拒絕所有在此分類帳中任何資料表上執行任何 PartiQL 命令的請求。若要允許 PartiQL 命令，除了分類帳的SendCommand API 許可之外，您還必須為特定資料表資源和 PartiQL 動作建立 IAM 許可政策。如需相關資訊，請參閱 開始使用 Amazon QLDB 中的標準許可模式。

> **Important**
>
> 在切換到STANDARD許可模式之前，您必須先建立所有必要的 IAM 政策和表格標記，以避免對使用者造成干擾。要了解更多信息，請繼續移轉至標準權限模式。

更新分類帳權限模式 (Java)

若要更新分類帳權限模式，請使用AWS SDK for Java

1. 建立 AmazonQLDB 類別的執行個體。
2. 建立 UpdateLedgerPermissionsModeRequest 類別的執行個體，以提供請求資訊。

   您必須提供分類帳名稱以及權限模式的新字串值。
3. 以參數形式提供請求物件，以便執行 updateLedgerPermissionsMode 方法。

下列程式碼範例會示範上述步驟。updateLedgerPermissionsMode請求會傳回已更新分類帳相關資訊的UpdateLedgerPermissionsModeResult物件。

Example — 分配標準權限模式

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerPermissionsModeRequest request = new UpdateLedgerPermissionsModeRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
UpdateLedgerPermissionsModeResult result = client.updateLedgerPermissionsMode(request);
```

更新分類帳權限模式 (AWS CLI)

將STANDARD權限模式指定給vehicle-registration分類帳。

Example

```
aws qldb update-ledger-permissions-mode --name vehicle-registration --permissions-mode
 STANDARD
```

移轉至標準權限模式

若要移轉至STANDARD許可模式，我們建議您分析 QLDB 存取模式，並新增 IAM 政策，以授予使用者存取其資源的適當權限。

在切換到STANDARD許可模式之前，您必須先建立所有必要的 IAM 政策和表格標記。否則，切換許可模式可能會中斷使用者，直到您建立正確的 IAM 政策或將許可模式還原為止ALLOW_ALL。如需建立這些政策的詳細資料，請參閱開始使用 Amazon QLDB 中的標準許可模式。

您也可以使用AWS受管理的策略來授與所有 QLDB 資源的完整存取權。AmazonQLDBFullAccess和受AmazonQLDBConsoleFullAccess管理的原則包括所有 QLDB 動作，包括所有 PartiQL 動作。將

其中一個原則附加至主參與者等同於該主參與者的ALLOW_ALL權限模式。如需詳細資訊，請參閱 AWS Amazon QLDB 的受管政策。

刪除分類帳

使用此DeleteLedger作業刪除分類帳及其所有內容。刪除分類帳是無法回復的作業。

如果已啟用分類帳的刪除保護，則必須先停用該功能才能刪除分類帳。

當您發出DeleteLedger請求時，分類帳的狀態會從變更ACTIVE為DELETING。刪除分類帳可能需要一段時間，具體取決於分類帳使用的存儲量。DeleteLedger操作結束時，QLDB 中就不再存在分類帳。

刪除分類帳 (Java)

若要刪除分類帳，請使用AWS SDK for Java

1. 建立 AmazonQLDB 類別的執行個體。
2. 建立DeleteLedgerRequest類別的執行環境，並提供您要刪除分類帳的名稱。
3. 以參數形式提供請求物件，以便執行 deleteLedger 方法。

下列程式碼範例示範前述步驟。

Example

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
DeleteLedgerRequest request = new DeleteLedgerRequest().withName(ledgerName);
DeleteLedgerResult result = client.deleteLedger(request);
```

刪除分類帳 (AWS CLI)

刪除分vehicle-registration類帳。

Example

```
aws qldb delete-ledger --name vehicle-registration
```

清單分類帳

ListLedgers作業會傳回目前AWS 帳戶與區域之所有 QLDB 分類帳的彙總資訊。

列出分類帳 (Java)

若要使用下列步驟列出您帳戶中的分類帳，AWS SDK for Java

1. 建立 AmazonQLDB 類別的執行個體。
2. 建立 ListLedgersRequest 類別的執行個體。

   如果您在先前ListLedgers呼叫的回應NextToken中收到的值，您必須在此要求中提供該值，才能取得下一頁結果。
3. 以參數形式提供請求物件，以便執行 listLedgers 方法。
4. 該listLedgers請求返回一個ListLedgersResult對象。這個對象有一個對LedgerSummary象的列表和一個分頁令牌，指示是否有更多的結果可用：

   - 如果NextToken為空，則表示結果的最後一頁已處理，並且沒有更多結果。
   - 如果不NextToken是空的，則會有更多可用的結果。若要擷取下一頁結果，請使用在後續ListLedgers呼叫NextToken中的值。

下列程式碼範例示範前述步驟。

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);
```

列出分類帳 (AWS CLI)

列出目前AWS 帳戶與區域中的所有分類帳。

Example

```
aws qldb list-ledgers
```

使用建立亞馬遜 QLDB 資源AWS CloudFormation

Amazon QLDB 已與整合AWS CloudFormation，這項服務可協助您建立AWS資源的模型和設定，以減少建立和管理資源和基礎設施的時間。您可以建立一個範本，描述所有您想要的AWS資源 (例如 QLDB:)，就會為您AWS CloudFormation佈建和設定那些資源。

當您使用時AWS CloudFormation，您可以重複使用您的範本，重複、一致的設定您的 QLDB 資源。只需描述一次您的資源，即可在多個 AWS 帳戶 帳戶與區域內重複佈建相同資源。

QLDB 和AWS CloudFormation模板

若要佈建和設定 QLDB 和相關服務的資源，則必須了解AWS CloudFormation範本。範本是以 JSON 或 YAML 格式化的文本檔案。而您亦可以透過這些範本的說明，了解欲在 AWS CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML，您可以使用 AWS CloudFormation Designer 協助您開始使用 AWS CloudFormation 範本。如需詳細資訊，請參閱 AWS CloudFormation 使用者指南中的什麼是 AWS CloudFormation Designer？。

QLDB 支援在中建立分類帳和分錄資料流AWS CloudFormation。如需更多詳細資訊 (包括分類帳和資料流的 JSON 和 YAML 範本範例)，請參閱AWS CloudFormation使用者指南中的下列資源類型參考：

- AWS: QLDB: QLDB: QLedger
- AWS: QLDB: QLDB: QLDB: QLDB:

進一步了解 AWS CloudFormation

若要進一步了解 AWS CloudFormation，請參閱下列資源：

- AWS CloudFormation
- 《AWS CloudFormation 使用者指南》
- AWS CloudFormation API 參考
- 《AWS CloudFormation 命令列介面使用者指南》

標記 Amazon QLDB 資源

標籤是一種自訂屬性標籤，可讓您指派或由 AWS 指派給 AWS 資源。每個標籤有兩個部分：

- 標籤鍵 (例如，`CostCenter`、`Environment` 或 `Project`)。標籤鍵會區分大小寫。

- 選用欄位，稱為標籤值 (例如 111122223333 或 Production)。忽略標籤值基本上等同於使用空字串。與標籤鍵相同，標籤值會區分大小寫。

標籤可協助您執行以下操作：

- 識別和組織您的 AWS 資源。許多 AWS 服務支援標記，因此您可以對來自不同服務的資源指派相同的標籤，指出資源是相關的。例如，您可以將指派給 Amazon S3 儲存貯體的相同標籤指派給 Amazon QLDB 分類帳
- 追蹤您的 AWS 成本。您可以在 AWS Billing and Cost Management 儀表板上啟用這些標籤。AWS 會使用標籤分類您的成本，並交付每月成本分配報告給您。如需詳細資訊，請參閱《AWS Billing 使用者指南》https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/中的使用成本分配標籤。
- 使用AWS Identity and Access Management (IAM) 控制對AWS資源的存取。如需詳細資訊，請參閱本開發人員指南基於屬性的訪問控制（ABAC）與 QLDB中的和 IAM 使用者指南中的使用 IAM 標籤控制存取。

如需使用標籤的提示，請參閱 AWS 答案部落格上的 AWS 標記策略文章。

下列各節會提供 Amazon QLDB 標籤的詳細資訊。

主題

- Amazon QLDB 中支援的資源
- 標籤命名和使用慣例
- 管理標籤
- 建立時標記資源

Amazon QLDB 中支援的資源

Amazon QLDB 中的下列資源支援標記：

- 分類帳
- table
- 期刊串留

如需新增和管理標籤的詳細資訊，請參閱管理標籤。

標籤命名和使用慣例

下列基本命名和使用慣例適用於搭配 Amazon QLDB 資源使用標籤：

- 每個資源的上限為 50 個標籤。
- 對於每一個資源，每個標籤金鑰必須是唯一的，且每個標籤金鑰只能有一個值。
- 最大標籤索引鍵長度為 128 個 UTF-8 形式的 Unicode 字元。
- 最大標籤值長度為 256 個 UTF-8 形式的 Unicode 字元。
- 允許的字元包括可用 UTF-8 表示的英文字母、數字、空格，還有以下特殊字元：. : + = @ _ / - (連字號)。
- 標籤鍵與值皆區分大小寫。做為最佳實務，請決定大寫標籤的策略，並一致地在所有資源類型中實作該策略。例如，決定要使用 `Costcenter`、`costcenter` 還是 `CostCenter`，並針對所有標籤使用相同的慣例。避免針對相似的標籤使用不一致的大小寫處理。
- 此 `aws:` 字首已保留供 AWS 使用。如果標籤具有字首為 `aws:` 的標籤金鑰時，您無法編輯或刪除標籤的金鑰或值。具此字首的標籤，不算在受資源限制的標籤計數內。

管理標籤

標籤是由資源上的 `Key` 和 `Value` 屬性組成。您可以使用 Amazon QLDB 主控台AWS CLI、或 QLDB API 來新增、編輯或刪除這些屬性的值。您也可以使用AWS Resource Groups標籤編輯器來管理標籤。

如需使用標籤的詳細資訊，請參閱下列 API 作業：

- ListTagsForResource在亞馬遜 QLDB API 參考
- TagResource在亞馬遜 QLDB API 參考
- UntagResource在亞馬遜 QLDB API 參考

若要使用 QLDB 標籤面板 (控制台)

1. 登入AWS Management Console，開啟位於 https://console.aws.amazon.com/qldb 的 Amazon QLDB 主控台。
2. 在導覽窗格中，選擇分類帳
3. 在「分類帳」清單中，選擇您要管理其盤點單的分類帳名稱。
4. 在分類帳詳細資料頁面上，找到「標籤」卡，然後選擇「管理標籤」。

5. 在管理標籤頁面上，您可以新增、編輯或移除分類帳 當您滿意標籤鍵和值時，請選擇 Save (儲存)。

建立時標記資源

對於支援標記的 QLDB 資源，您可以在建立資源時使用AWS Management ConsoleAWS CLI、或 QLDB API 來定義標籤。藉由在建立時為資源建立標籤，您可以消除在資源建立後執行自訂標籤指令碼的必要。

標記資源後，您可以根據這些標籤控制對資源的存取。例如，您只能授與具有特定標籤的表格資源的完整存取權。如需 JSON 政策範例，請參閱根據表格標籤對所有動作的完整存取權。

> **Note**
>
> 表和流資源不會繼承其根分類帳資源的標籤。

您也可以在CREATE TABLE PartiQL 陳述式中指定資料表標籤來定義資料表標籤。如需進一步了解，請參閱標籤表格。

Amazon QLDB 中的安全性

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，該架構專為滿足對安全性最敏感的組織的需求而打造。

安全是 AWS 與您之間共同承擔的責任。共同責任模型 將此描述為雲端的安全和雲端內的安全：

- 雲端的安全性 — AWS 負責保護 AWS 服務 中執行的基礎架構 AWS 雲端。 AWS 還為您提供可以安全使用的服務。在 AWS 合規計畫中，第三方稽核員會定期測試並驗證我們的安全功效。若要了解適用於 Amazon QLDB 的合規計劃，請參閱合規計劃適用範圍的AWS 服務。
- 雲端中的安全性 — 您的責任取決於您使用的資料。 AWS 服務 您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您瞭解如何在使用 QLDB 時套用共同的責任模式。下列主題說明如何設定 QLDB 以符合安全性和合規性目標。您也會學到如何使用其 AWS 服務 他協助您監控和保護您的 QLDB 資源。

主題

- Amazon QLDB 中的資料保護
- Amazon QLDB 的 Identity and Access Management
- 在 Amazon QLDB 中進行記錄和監控
- Amazon QLDB 的合規驗證
- Amazon QLDB 的韌性
- Amazon QLDB 中的基礎設施安全

Amazon QLDB 中的資料保護

AWS 共同責任模型適用於 Amazon QLDB 中的資料保護。如此模型所述， AWS 負責保護執行所有 AWS 雲端. 您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源進行通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案以及其中的所有默認安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在透過命令列介面或 API 存取時需要經 AWS 過 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用主控台、API 或開發套件 AWS 服務 使用 QLDB 或 AWS 其他工作時。 AWS CLI您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

> **Note**
>
> 關於避免使用敏感資訊的標籤或任意格式欄位的指引，是指 QLDB 分類帳資源的中繼資料，而非儲存在分類帳中的資料。儲存在 QLDB 分類帳資源中的資料不會用於帳單或診斷記錄。

主題

Amazon QLDB 中的靜態加密

根據預設，所有儲存在 Amazon QLDB 中的資料都會在靜態時完全加密。靜態 QLDB 加密透過使用 () 中 AWS Key Management Service 的加密金鑰加密所有靜態總帳資料，以提供增強的安全性。AWS KMS此功能協助降低了保護敏感資料所涉及的操作負擔和複雜性。透過靜態加密，您可以建置符合嚴格加密合規性和法規要求的安全敏感型總帳應用程式。

靜態加密與 AWS KMS 管理用於保護 QLDB 分類帳的加密金鑰整合。如需詳細資訊 AWS KMS，請參閱AWS Key Management Service 開發人員指南中的AWS Key Management Service 概念。

在 QLDB 中，您可以指定每個分類帳資源 AWS KMS key 的型態。建立新分類帳或更新現有分類帳時，您可以選擇下列其中一種 KMS 金鑰類型來保護分類帳資料：

- AWS 擁有的金鑰— 預設的加密類型。密鑰由 QLDB 擁有（不收取額外費用）。
- 客戶管理金鑰 — 金鑰儲存在您的金鑰中， AWS 帳戶 並由您建立、擁有及管理。您可以完全控制密鑰（AWS KMS 收費）。

> **Note**
>
> Amazon QLDB 於 2021 年 7 月 22 日推出了 AWS KMS keys 對管理客戶的支援。依預設，在啟動之前建立的任何分類帳都會 AWS 擁有的金鑰 受到保護，但目前不符合使用客戶管理金鑰進行靜態加密的資格。
> 您可以在 QLDB 主控台上檢視分類帳的建立時間。

存取分類帳時，QLDB 會以透明方式解密資料。您可以在任何給定時間在 AWS 擁有的金鑰 和客戶管理的金鑰之間切換。您不需要變更任何程式碼或應用程式即可使用或管理加密資料。

您可以在建立新分類帳時指定加密金鑰 AWS Management Console，或使用 QLDB API 或 AWS Command Line Interface ()AWS CLI變更現有分類帳上的加密金鑰。如需詳細資訊，請參閱 在 Amazon QLDB 中使用客戶受管金鑰。

> **Note**
>
> 根據預設，Amazon QLDB 會自動啟用 AWS 擁有的金鑰 靜態加密，無需額外付費。但是，使用客戶管理的金鑰需要 AWS KMS 支付費用。如需定價的詳細資訊，請參閱 AWS Key Management Service 定價。
> QLDB 靜態加密可在所有 QLDB 可用的 AWS 區域 地方使用。

主題

靜態加密：它在 Amazon QLDB 中的工作原理

靜態 QLDB 加密會使用 256 位元進階加密標準 (AES-256) 來加密您的資料。這有助於保護您的數據免受未經授權的訪問基礎存儲。依預設，儲存在 QLDB 分類帳中的所有資料都會在靜態時加密。伺服器端加密是透明的，這表示不需要變更應用程式。

靜態加密與 AWS Key Management Service (AWS KMS) 整合，用於管理用來保護 QLDB 分類帳的加密金鑰。建立新分類帳或更新現有分類帳時，您可以選擇下列其中一種 AWS KMS 索引鍵型態：

- AWS 擁有的金鑰— 預設的加密類型。密鑰由 QLDB 擁有（不收取額外費用）。
- 客戶管理金鑰 — 金鑰儲存在您的金鑰中， AWS 帳戶 並由您建立、擁有及管理。您可以完全控制密鑰（AWS KMS 收費）。

主題

- AWS 擁有的金鑰
- 客戶受管金鑰
- Amazon QLDB 如何使用贈款 AWS KMS
- 恢復補助金 AWS KMS
- 靜態加密注意事項

AWS 擁有的金鑰

AWS 擁有的金鑰 不會儲存在您的 AWS 帳戶. 它們是 AWS 擁有和管理以供多個使用的 KMS 金鑰集合的一部分 AWS 帳戶。 AWS 服務 可用 AWS 擁有的金鑰 來保護您的資料。

您不需要建立或管理 AWS 擁有的金鑰。但是，您無法查看或跟踪 AWS 擁有的金鑰或審核其使用情況。我們不會向您收取月費或使用費 AWS 擁有的金鑰，也不會計入您帳戶的 AWS KMS 配額中。

如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的 AWS 擁有的金鑰。

客戶受管金鑰

客戶受管金鑰是您 AWS 帳戶 建立、擁有和管理的 KMS 金鑰。您可以完全控制這些 KMS 金鑰。QLDB 僅支援對稱式加密 KMS 金鑰。

使用客戶受管金鑰來取得下列功能：

- 設定和維護金鑰政策、IAM 政策和授權，以控制金鑰的存取權
- 啟用和停用金鑰
- 旋轉密碼材料的鑰匙
- 建立金鑰標籤和別名
- 排定要刪除的金鑰

- 匯入您自己的金鑰材料，或使用您擁有和管理的自訂金鑰存放區
- 使用 AWS CloudTrail 和 Amazon CloudWatch 日誌來追蹤 QLDB 代表您傳送 AWS KMS 的請求

如需更多資訊，請參閱 AWS Key Management Service 開發人員指南中的客戶受管金鑰。

客戶受管金鑰會針對每次 API 呼叫產生費用，而且這些 KMS 金鑰會套用 AWS KMS 配額。如需詳細資訊，請參閱AWS KMS 資源或要求配額。

當您將客戶受管金鑰指定為分類帳的 KMS 金鑰時，日誌儲存和索引儲存體中的所有總帳資料都會使用相同的客戶管理金鑰來保護。

無法存取的客戶管理

如果停用客戶管理的金鑰、排定要刪除的金鑰，或撤銷金鑰的授權，則分類帳加密的狀態會變為`KMS_KEY_INACCESSIBLE`。在這種狀態下，分類帳受損，不接受任何讀取或寫入請求。無法存取的金鑰可防止所有使用者和 QLDB 服務加密或解密資料，並且無法在分類帳中執行讀取和寫入作業。QLDB 必須具有 KMS 金鑰的存取權，以確保您可以繼續存取分類帳並防止資料遺失。

> **Important**
>
> 在您還原金鑰的授權之後，或重新啟用已停用的金鑰之後，受損的分類帳會自動返回作用中狀態。
>
> 但是，刪除客戶管理的金鑰是不可逆轉的。刪除金鑰後，您將無法再存取受該金鑰保護的分類帳，而且資料永久無法復原。

若要檢查分類帳的加密狀態，請使用 AWS Management Console 或 DescribeLedgerAPI 作業。

Amazon QLDB 如何使用贈款 AWS KMS

QLDB 需要授權才能使用您的客戶管理金鑰。當您建立受客戶管理金鑰保護的分類帳時，QLDB 會將CreateGrant請求傳送至以代表您建立授權。 AWS KMS中的授權 AWS KMS 用於將 QLDB 存取權授與客戶中的 KMS 金鑰。 AWS 帳戶如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的使用授權。

QLDB 要求授權使用您的客戶管理金鑰進行下列 AWS KMS 作業：

- DescribeKey— 確認指定的對稱加密 KMS 金鑰有效。
- GenerateDataKey— 產生唯一的對稱資料金鑰，QLDB 用來加密分類帳中的靜態資料。

- 解密 — 解密由客戶管理金鑰加密的資料金鑰。
- 加密 — 使用客戶管理的金鑰將明文加密為密文。

您可以隨時撤銷授權，以移除服務對客戶管理金鑰的存取權。如果這樣做，金鑰就會變得無法存取，而且 QLDB 無法存取任何受客戶管理金鑰保護的總帳資料。在這種狀態下，分類帳會受損，並且在您恢復密鑰的授權之前不接受任何讀取或寫入請求。

恢復補助金 AWS KMS

若要還原客戶受管金鑰的授權並復原 QLDB 中分類帳的存取權，您可以更新分類帳並指定相同的 KMS 金鑰。如需說明，請參閱更新現 AWS KMS key 有分類帳。

靜態加密注意事項

當您在 QLDB 中使用靜態加密時，請考慮下列事項：

- 依預設，所有 QLDB 分類帳資料都會啟用伺服器端靜態加密，且無法停用。您無法僅加密分類帳中的資料子集。
- 只有持久性儲存媒體上的資料是靜態 (靜止) 時，靜態加密才會加密資料。如果資料安全性是傳輸中資料或使用中資料的考量，您可能需要採取下列其他措施：
  - 傳輸中的資料：QLDB 中的所有資料在傳輸過程中都經過加密。根據預設，QLDB 之間的通訊會使用 HTTPS 通訊協定，使用安全通訊端層 (SSL)/傳輸層安全性 (TLS) 加密來保護網路流量。
  - 使用中的資料：在將資料傳送至 QLDB 之前，請先使用用戶端加密來保護您的資料。

若要瞭解如何實作分類帳的客戶管理金鑰，請繼續執行在 Amazon QLDB 中使用客戶受管金鑰。

在 Amazon QLDB 中使用客戶受管金鑰

您可以使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 QLDB API 來 AWS KMS key 為 Amazon QLDB 中的新分類帳和現有分類帳指定。下列主題說明如何管理及監控 QLDB 中客戶管理金鑰的使用情況。

主題

- 必要條件
- AWS KMS key 為新分類帳指定
- 更新現 AWS KMS key 有分類帳
- 監控您的 AWS KMS keys

必要條件

您必須先在 AWS Key Management Service (AWS KMS) 中建立金鑰，才能使用客戶管理的金鑰保護 QLDB 分類帳。您還必須指定允許 QLDB 代表您建立授權的金鑰原則。 AWS KMS key

建立客戶管理的金鑰

若要建立客戶受管金鑰，請依照AWS Key Management Service 開發人員指南中的建立對稱加密 KMS 金鑰中的步驟進行。QLDB 不支援非對稱金鑰。

設定金鑰政策

主要原則是控制中客戶受管金鑰存取的主要方式 AWS KMS。每個客戶管理的金鑰都必須只有一個金鑰政策。金鑰政策文件中的陳述式決定誰有使用 KMS 金鑰的許可以及可以使用它的方式。如需詳細資訊，請參閱中的使用金鑰原則 AWS KMS。

您可以在建立客戶管理金鑰時指定金鑰政策。若要變更現有客戶受管金鑰的金鑰政策，請參閱變更金鑰政策。

若要允許 QLDB 使用您的客戶管理金鑰，金鑰原則必須包含下列 AWS KMS 動作的權限：

- kms：CreateGrant— 將授權新增至客戶管理的金鑰。授與指定 KMS 金鑰的控制權存取權。

  當您使用指定的客戶管理金鑰建立或更新分類帳時，QLDB 會建立授權以允許存取其所需的授權作業。授權操作包括以下內容：

  - GenerateDataKey
  - 解密
  - 加密
- kms：DescribeKey— 傳回有關客戶管理金鑰的詳細資訊。QLDB 會使用此資訊來驗證金鑰。

關鍵政策範例

以下是可用於 QLDB 的金鑰原則範例。此原則允許授權從帳戶使用 QLDB 的主參與者呼叫111122223333資源的DescribeKey和CreateGrant作業。arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab

若要使用此原則，請使用您自己的資訊取代範例中的 *us-east-1*、*111122223333* 及 *12* 34 英文版。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid" : "Allow access to principals authorized to use Amazon QLDB",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "*"
            },
            "Action" : [
                "kms:DescribeKey",
                "kms:CreateGrant"
            ],
            "Resource" : "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "Condition" : {
                "StringEquals" : {
                    "kms:ViaService" : "qldb.us-east-1.amazonaws.com",
                    "kms:CallerAccount" : "111122223333"
                }
            }
        }
    ]
}
```

AWS KMS key 為新分類帳指定

使用 QLDB 主控台或建立新分類帳時，請依照下列步驟指定 KMS 金鑰。 AWS CLI

您可以使用 ID、別名或 Amazon 資源名稱 (ARN) 來指定客戶受管金鑰。若要深入了解，請參閱AWS Key Management Service 開發人員指南中的金鑰識別碼 (KeyId)。

> **Note**
>
> 不支援跨區域金鑰。指定的 KMS 金鑰必須與您的分類帳位於 AWS 區域 相同。

創建分類帳（控制台）

1. 登入 AWS Management Console，然後開啟 Amazon QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 選擇建立分類帳。
3. 在 「建立分類帳」 頁面上，執行下列作業：

- 分類帳資訊 — 輸入在目前 AWS 帳戶 與區域中所有分類帳中唯一的 「分類帳」名稱。
- 權限模式 — 選擇要分配給分類帳的權限模式：
  - 允許全部
  - 標準 (建議)
- 加密靜態資料 — 選擇用於靜態加密的 KMS 金鑰類型：
  - 使用 AWS 擁有的 KMS 金鑰 — 使用代表您擁有和管理 AWS 的 KMS 金鑰。這是預設選項，不需要額外的設定。
  - 選擇不同的金 AWS KMS 鑰 — 在您建立、擁有和管理的帳戶中使用對稱加密 KMS 金鑰。

    若要使用 AWS KMS 主控台建立新金鑰，請選擇 [建立 AWS KMS 金鑰]。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的建立對稱加密 KMS 金鑰。

    若要使用現有的 KMS 金鑰，請從下拉式清單中選擇一個金鑰，或指定 KMS 金鑰 ARN。

4. 如果設定符合您的需求，請選擇 「建立分類帳」。

   當 QLDB 分類帳狀態為「有效」時，您可以存取 QLDB 分類帳。這可能需要幾分鐘的時間。

創建分類帳（AWS CLI）

使用在 AWS CLI QLDB 中建立具有預設金鑰 AWS 擁有的金鑰 或客戶管理金鑰的分類帳。

Example — 使用預設值建立分類帳 AWS 擁有的金鑰

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

Example — 使用客戶管理的密鑰創建分類帳

```
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
```

更新現 AWS KMS key 有分類帳

您也可以使用 QLDB 主控台或隨時 AWS CLI 將現有分類帳的 KMS 金鑰更新為 AWS 擁有的金鑰 或客戶管理的金鑰。

> **Note**
>
> Amazon QLDB 於 2021 年 7 月 22 日推出了 AWS KMS keys 對管理客戶的支援。依預設，在啟動之前建立的任何分類帳都會 AWS 擁有的金鑰 受到保護，但目前不符合使用客戶管理金鑰進行靜態加密的資格。
> 您可以在 QLDB 主控台上檢視分類帳的建立時間。

QLDB 中的關鍵變更是非同步的。在處理金鑰變更時，可完全存取分類帳，而不會造成任何效能影響。更新金鑰所需的時間長度視分類帳大小而有所不同。

您可以使用 ID、別名或 Amazon 資源名稱 (ARN) 來指定客戶受管金鑰。若要深入了解，請參閱AWS Key Management Service 開發人員指南中的金鑰識別碼 (KeyId)。

> **Note**
>
> 不支援跨區域金鑰。指定的 KMS 金鑰必須與您的分類帳位於 AWS 區域 相同。

更新分類帳（控制台）

1. 登入 AWS Management Console，然後開啟 Amazon QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在瀏覽窗格中，選擇「分類帳」。
3. 在分類帳清單中，選取您要更新的分類帳，然後選擇「編輯分類帳」。
4. 在 [編輯分類帳] 頁面上，選擇用於靜態加密的 KMS 金鑰類型：
   - 使用 AWS 擁有的 KMS 金鑰 — 使用代表您擁有和管理 AWS 的 KMS 金鑰。這是預設選項，不需要額外的設定。
   - 選擇不同的金 AWS KMS 鑰 — 在您建立、擁有和管理的帳戶中使用對稱加密 KMS 金鑰。

     若要使用 AWS KMS 主控台建立新金鑰，請選擇 [建立 AWS KMS 金鑰]。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的建立對稱加密 KMS 金鑰。

     若要使用現有的 KMS 金鑰，請從下拉式清單中選擇一個金鑰，或指定 KMS 金鑰 ARN。
5. 選擇確認變更。

更新分類帳 (AWS CLI)

使用可使用預設金鑰 AWS 擁有的金鑰 或客戶管理金鑰 AWS CLI 來更新 QLDB 中現有的分類帳。

Example — 若要使用預設值更新分類帳 AWS 擁有的金鑰

```
aws qldb update-ledger --name my-example-ledger --kms-key AWS_OWNED_KMS_KEY
```

Example — 使用客戶管理的密鑰更新分類帳

```
aws qldb update-ledger \
    --name my-example-ledger \
    --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

監控您的 AWS KMS keys

如果您使用客戶受管金鑰來保護 Amazon QLDB 分類帳，則可以使用AWS CloudTrail或 Amazon CloudWatch Logs 來追蹤 QLDB 代表您傳送的請求。 AWS KMS 如需詳細資訊，請參閱AWS Key Management Service 開發人員指南 AWS KMS keys中的監控。

下列範例是作業CreateGrant、、GenerateDataKeyDecrypt、Encrypt和的 CloudTrail 記錄項目DescribeKey。

CreateGrant

當您指定用來保護分類帳的客戶管理金鑰時，QLDB 會代表您傳送CreateGrant要 AWS KMS 求，以允許存取 KMS 金鑰。此外，當您刪除分類帳時，QLDB 會使用此RetireGrant作業移除授權。

QLDB 建立的授權是分類帳特有的。CreateGrant請求中的主參與者是建立表格的使用者。

記錄 CreateGrant 操作的事件類似於以下範例事件。這些參數包括客戶受管金鑰的 Amazon 資源名稱 (ARN)、受權者主體和淘汰主體 (QLDB 服務)，以及授權涵蓋的作業。

```
{
    "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

當您指定客戶管理的金鑰來保護分類帳時，QLDB 會建立唯一的資料金鑰。它會傳送GenerateDataKey要求 AWS KMS ，以指定分類帳的客戶管理金鑰。

記錄 GenerateDataKey 操作的事件類似於以下範例事件。使用者是 QLDB 服務帳戶。參數包括客戶管理金鑰的 ARN、需要 32 位元組長度的資料金鑰指定字，以及識別內部金鑰階層節點的加密內容。

```
{
    "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"
}
```

解密

存取分類帳時，QLDB 會呼叫`Decrypt`作業以解密分類帳的已儲存資料金鑰，以便存取分類帳中的加密資料。

記錄 `Decrypt` 操作的事件類似於以下範例事件。使用者是 QLDB 服務帳戶。這些參數包括客戶管理金鑰的 ARN，以及識別內部金鑰階層節點的加密內容。

```
{
    "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"
}
```

加密

QLDB 呼叫Encrypt作業，使用您的客戶管理金鑰將明文加密為密文。

記錄 Encrypt 操作的事件類似於以下範例事件。使用者是 QLDB 服務帳戶。參數包括客戶管理金鑰的 ARN，以及指定分類帳內部唯一 ID 的加密內容。

```
{
    "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 會呼叫`DescribeKey`作業，以判斷您指定的 KMS 金鑰是否存在於 AWS 帳戶 和區域中。

記錄 `DescribeKey` 操作的事件類似於以下範例事件。主體是指定 KMS 金鑰 AWS 帳戶 的使用者。這些參數包括客戶管理金鑰的 ARN。

```
{
    "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"
}
```

Amazon QLDB 中的傳輸中加密

Amazon QLDB 僅接受使用 HTTPS 通訊協定的安全連線，此通訊協定會使用安全通訊端層 (SSL) /傳輸層安全性 (TLS) 來保護網路流量。傳輸中的加密功能會在資料往返 QLDB 時加密，藉此提供額外一層的資料保護。組織原則、產業或政府法規以及合規性要求通常需要在傳輸過程中使用加密，以提高應用程式透過網路傳輸資料時的資料安全性。

QLDB 也在選取的區域中提供 FIPS 端點。與標準 AWS 端點不同，FIPS 端點使用符合美國聯邦資訊處理標準 (FIPS) 140-2 的 TLS 軟體程式庫。會與美國政府互動的企業可能需要這些端點。如需詳細資訊，請參閱《》中的 AWS 一般參考FIPS 端點。如需 QLDB 可用區域和端點的完整清單，請參閱 Amazon QLDB 端點和配額。

Amazon QLDB 的 Identity and Access Management

AWS Identity and Access Management (IAM) 可協助管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員控制哪些人可以驗證 (登入) 和授權 (具有權限) 以使用 QLDB 資源。IAM 是您可以使用的 AWS 服務 ，無需額外付費。

主題

- 物件
- 使用身分驗證
- 使用政策管理存取權
- Amazon QLDB 如何與 IAM 合作
- 開始使用 Amazon QLDB 中的標準許可模式
- Amazon QLDB 以身分識別為基礎的政策範例
- 預防跨服務混淆代理人
- AWS Amazon QLDB 的受管政策
- 疑難排解 Amazon QLDB 身分識別和存取

物件

根據您在 QLDB 中執行的工作，使用方式 AWS Identity and Access Management (IAM) 會有所不同。

服務使用者 — 如果您使用 QLDB 服務執行工作，則管理員會為您提供所需的認證和權限。當您使用更多 QLDB 功能來完成工作時，您可能需要額外的權限。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 QLDB 中的功能，請參閱。疑難排解 Amazon QLDB 身分識別和存取

服務管理員 — 如果您負責公司的 QLDB 資源，您可能擁有 QLDB 的完整存取權。決定您的服務使用者應該存取哪些 QLDB 功能和資源是您的工作。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何將 IAM 與 QLDB 搭配使用，請參閱。Amazon QLDB 如何與 IAM 合作

IAM 管理員 — 如果您是 IAM 管理員，可能需要瞭解如何撰寫政策來管理 QLDB 存取權的詳細資訊。若要檢視可在 IAM 中使用的 QLDB 身分型政策範例，請參閱。Amazon QLDB 以身分識別為基礎的政策範例

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證和 IAM 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務 和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的需要根使用者憑證的任務。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時登入資料進行存取 AWS 服務 。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需 IAM Identity Center 的相關資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱 IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法更多相關資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 – 若要向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，

則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。

- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權 – 您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的委託人) 存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取權角色和資源型政策間的差異，請參閱 IAM 使用者指南中的 IAM 角色與資源類型政策的差異。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色來管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需更多資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

若要了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色

工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的更多相關資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。若要了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些委託人 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 和 Amazon VPC 是支援 ACL 的服務範例。 AWS WAF若要進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可範圍的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 實體許可範圍。
- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需組織和 SCP 的更多相關資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需更多資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

Amazon QLDB 如何與 IAM 合作

在您使用 IAM 管理 QLDB 的存取權限之前，請先了解哪些 IAM 功能可搭配 QLDB 使用。

您可以搭配 Amazon QLDB 使用的 IAM 功能

IAM 功能	QLDB 支援
身分型政策	是
資源型政策	否

IAM 功能	QLDB 支援
政策動作	是
政策資源	是
政策條件索引鍵	是
ACL	否
ABAC (政策中的標籤)	是
臨時憑證	是
主體許可	否
服務角色	是
服務連結角色	否

若要深入了解 QLDB 和其他如何使 AWS 服務 用大多數 IAM 功能，請參閱 AWS 服務 IAM 使用者指南中的 IAM。

QLDB 的以身分識別為基礎的原則

支援身分型政策	是

身分型政策是可以連接到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

QLDB 的身分識別原則範例

若要檢視以 QLDB 身分識別為基礎的原則範例，請參閱。Amazon QLDB 以身分識別為基礎的政策範例

QLDB 中以資源為基礎的政策

支援以資源基礎的政策	否

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

若要啟用跨帳戶存取，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源位於不同時 AWS 帳戶，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 角色與資源型政策有何差異。

QLDB 的政策動作

支援政策動作	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼条件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看 QLDB 動作清單，請參閱服務授權參考中由 Amazon QLDB 定義的動作。

QLDB 中的原則動作會在動作之前使用下列前置詞：

```
qldb
```

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "qldb:action1",
      "qldb:action2"
]
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定開頭是 Describe 文字的所有動作，請包含以下動作：

```
"Action": "qldb:Describe*"
```

若要透過在分類帳上執行 PartiQL 陳述式與 QLDB 交易資料 API (QLDB 工作階段) 互動，您必須授與動作的權限，如下所示。SendCommand

```
"Action": "qldb:SendCommand"
```

對於STANDARD權限模式下的分類帳，請參閱以取PartiQL 參考得每個 PartiQL 命令的其他必要權限。

若要檢視以 QLDB 身分識別為基礎的原則範例，請參閱。Amazon QLDB 以身分識別為基礎的政策範例

QLDB 的政策資源

支援政策資源	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

若要查看 QLDB 資源類型及其 ARN 的清單，請參閱服務授權參考資料中的 Amazon QLDB 定義的資源。若要了解可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon QLDB 定義的動作。

在 QLDB 中，主要資源為分類帳。QLDB 也支援其他資源類型：資料表和串流。但是，您只能在現有分類帳的內容中建立表格和串流。

QLDB 表格是分類帳分錄中文件修訂之無序集合的具體化視觀表。在分類帳的STANDARD許可模式中，您必須建立 IAM 政策，以授與權限，才能在此表格資源上執行 PartiQL 陳述式。透過表格資源的權限，您可以執行存取資料表目前狀態的陳述式。您還可以使用內置`history()`函數查詢表格的修訂歷史記錄。如需進一步了解，請參閱開始使用 Amazon QLDB 中的標準許可模式。

> **Note**
>
> 該`CREATE TABLE`語句創建具有唯一 ID 和提供的表名稱的表。所提供的表格名稱在所有使用中的表格中必須是唯一的。不過，QLDB 可讓您停用資料表，因此可能有多個非作用中的表格共用相同的資料表名稱。因此，表格資源 ARN 會參考系統指派的唯一 ID，而非使用者定義的資料表名稱。

每個分類帳還提供系統定義的型錄資源，您可以查詢此資源，以列出分類帳中的所有表格與索引。如需有關 QLDB 資料物件模型的詳細資訊，請參閱〈〉。亞馬遜 QLDB 中的核心概念和術語

這些資源具有與其相關聯的唯一 ARN，如下表所示。

資源類型	ARN
ledger	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}`
table	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}/table/${TableId}`
catalog	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}/information_schema/user_tables`
stream	`arn:${Partition}:qldb:${Region}:${Account}:stream/${LedgerName}/${StreamId}`

例如，若要在陳述式中指定`myExampleLedger`資源，請使用下列 ARN。

```
"Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
```

若要在單一陳述式中指定多項資源，請使用逗號分隔 ARN。

```
"Resource": [
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger1",
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger2"
]
```

若要檢視以 QLDB 身分識別為基礎的原則範例，請參閱。Amazon QLDB 以身分識別為基礎的政策範例

QLDB 的原則條件金鑰

支援服務特定政策條件金鑰	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

若要查看 QLDB 條件金鑰清單，請參閱服務授權參考中的 Amazon QLDB 條件金鑰。若要了解可以使用條件金鑰的動作和資源，請參閱 Amazon QLDB 定義的動作。

PartiQLDropIndex和動PartiQLDropTable作支援qldb:Purge條件索引鍵。這個條件索引鍵會依照 PartiQL DROP 陳述式中指定的purge值來篩選存取。不過，QLDB 目前只purge = true支援陳述式和DROP INDEXpurge = false陳述式。DROP TABLE其他 QLDB 動作支援某些全域條件索引鍵。

若要檢視以 QLDB 身分識別為基礎的原則範例，請參閱。Amazon QLDB 以身分識別為基礎的政策範例

QLDB 中的存取控制清單 (ACL)

支援 ACL	否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

基於屬性的訪問控制（ABAC）與 QLDB

支援 ABAC (政策中的標籤)	是

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

若要根據標籤控制存取，請使用 `aws:ResourceTag/`*`key-name`*、`aws:RequestTag/`*`key-name`* 或 `aws:TagKeys` 條件金鑰，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱《IAM 使用者指南》中的使用屬性型存取控制 (ABAC)。

如需標記 QLDB 資源的詳細資訊，請參閱。標記 Amazon QLDB 資源

若要檢視身分型政策範例，以根據該資源上的標籤來限制存取資源，請參閱根據標籤更新 QLDB 分類帳。

搭配 QLDB 使用臨時登入資料

支援臨時憑證	是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配AWS 服務 使用，請參閱 IAM 使用者指南中的 IAM。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立暫時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的切換至角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

QLDB 的跨服務主體權限

支援轉寄存取工作階段 (FAS)	否

當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

QLDB 的服務角色

支援服務角色	是

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

> **Warning**
>
> 變更服務角色的權限可能會中斷 QLDB 功能。只有在 QLDB 提供指引時才編輯服務角色。

QLDB 支援`ExportJournalToS3`和 `StreamJournalToKinesis` API 作業的服務角色，如下節所述。

在 QLDB 中選擇 IAM 角色

當您匯出或串流 QLDB 中的日誌區塊時，您必須選擇一個角色，以允許 QLDB 代表您將物件寫入指定的目的地。如果您先前已建立服務角色，QLDB 會提供您可供選擇的角色清單。選擇允許存取寫入指定 Amazon S3 儲存貯體以進行匯出的角色，或選擇指定的 Amazon Kinesis 資料串流資源以進行串流的角色。如需詳細資訊，請參閱 QLDB 中的分錄匯出權限 或 QLDB 中的串流權限 。

> **Note**
>
> 若要在請求日誌匯出或串流時將角色傳遞給 QLDB，您必須擁有對 IAM 角色資源執行`iam:PassRole`動作的權限。這是`qldb:ExportJournalToS3`對 QLDB 分類帳資源或 QLDB 串流子資源執行權限`qldb:StreamJournalToKinesis`的附加權限。

QLDB 的服務連結角色

支援服務連結角色。	否

服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理服務連結角色的詳細資訊 AWS 服務 ，請參閱使用 IAM。在表格中尋找服務，其中包含服務連結角色欄中的 `Yes`。選擇是連結，以檢視該服務的服務連結角色文件。

開始使用 Amazon QLDB 中的標準許可模式

您可以使用本節開始使用 Amazon QLDB 中的標準許可模式。本節提供參考資料表，可協助您在 AWS Identity and Access Management (IAM) 中針對 QLDB 中的 PartiQL 動作和表格資源撰寫以身分識別為基礎的政策。其中也包含在 IAM 中建立許可政策的 step-by-step 教學課程，以及如何在 QLDB 中尋找資料表 ARN 和建立資料表標籤的指示。

主題

- 權STANDARD限模式
- PartiQL 參考
- 尋找資料表識別碼和 ARN
- 標籤表格
- 快速入門教學課程：建立權限原則

權STANDARD限模式

QLDB 現在支援分類帳資源的STANDARD權限模式。標準權限模式可針對分類帳、資料表和 PartiQL 命令提供更精細的存取控制。根據預設，此模式會拒絕在分類帳中的任何資料表上執行任何 PartiQL 命令的所有請求。

> **Note**
>
> 以前，分類帳的唯一可用權限模式是ALLOW_ALL。此ALLOW_ALL模式可透過分類帳的 API 層級精細度啟用存取控制，並繼續為 QLDB 分類帳提供支援 (但不建議)。此模式允許具有 SendCommand API 權限的使用者在權限原則指定的分類帳中的任何資料表上執行所有 PartiQL 命令 (因此，「允許所有」PartiQL 命令)。
> 您可以將現有分類帳的權限模式從變更ALLOW_ALL為STANDARD。如需相關資訊，請參閱移轉至標準權限模式。

若要在標準模式下允許命令，您必須在 IAM 中針對特定資料表資源和 PartiQL 動作建立許可政策。這是分類帳 SendCommand API 權限的附加功能。為了促進此模式下的政策，QLDB 針對 PartiQL 命令引入了一組 IAM 動作，以及針對 QLDB 表引入了 Amazon 資源名稱 (ARN)。如需有關 QLDB 資料物件模型的詳細資訊，請參閱〈〉。亞馬遜 QLDB 中的核心概念和術語

PartiQL 參考

下表列出每個 QLDB PartiQL 命令、您必須授予權限才能執行命令的對應 IAM 動作，以及您可以授與權限的 AWS 資源。您在政策的 `Action` 欄位中指定動作，然後在政策的 `Resource` 欄位中指定資源值。

> **Important**
>
> - 授予這些 PartiQL 命令許可的 IAM 政策僅在將STANDARD許可模式分配給分類帳時適用於您的總帳。這些原則不適用於ALLOW_ALL權限模式下的分類帳。
>
> 若要瞭解如何在建立或更新分類帳時指定權限模式Amazon QLDB 分類帳的基本操作，請參閱或步驟 1：建立新的分類帳在開始使用主控台中的。
> - 若要在分類帳上執行任何 PartiQL 命令，您還必須授與分類帳資源之 SendCommand API 動作的權限。這是下表中列出的 PartiQL 動作和表格資源的附加功能。如需詳細資訊，請參閱執行資料交易。

Amazon QLDB 的命 PartiQL 和所需的許可

Command	所需的許可 (IAM 動作)	資源	相依動作
CREATE TABLE	qldb:PartiQLCreateTable	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*	qldb:TagResource (用於建立時標記)
DROP TABLE	qldb:PartiQLDropTable	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
取消刪除資料表	qldb:PartiQLUndropTable	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
CREATE INDEX	qldb:PartiQLCreateIndex	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
DROP INDEX	qldb:PartiQLDropIndex	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
DELETE 來源-移除 (針對整個文件)	qldb:PartiQLDelete	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	qldb:PartiQLSelect
INSERT	qldb:PartiQLInsert	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
UPDATE		arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	

Command	所需的許可 (IAM 動作)	資源	相依動作
從 (插入、移除或設定)	qldb:PartiQLUpdate		qldb:PartiQLSelect
修訂版(預存程序)	qldb:PartiQLRedact	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
從表格名稱中選擇	qldb:PartiQLSelect	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
從資訊架構. 使用者表格中選擇	qldb:PartiQLSelect	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /information_schema/user_tables	
從歷史記錄中選擇（表格名稱）	qldb:PartiQLHistoryFunction	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	

如需授予這些 PartiQL 命令許可的 IAM 政策文件範例，請繼續快速入門教學課程：建立權限原則或參閱Amazon QLDB 以身分識別為基礎的政策範例。

尋找資料表識別碼和 ARN

您可以使用 AWS Management Console 或查詢資料表資訊 _schema.user _tables 來尋找資料表識別碼。若要在主控台上檢視表格詳細資訊，或查詢此系統目錄表格，您必須擁有系統目錄資源的SELECT權限。例如，若要尋找資料表的資Vehicle料表 ID，您可以執行下列陳述式。

```
SELECT * FROM information_schema.user_tables
WHERE name = 'Vehicle'
```

此查詢會以類似下列範例的格式傳回結果。

```
{
    tableId: "Au1EiThbt8s0z9wM26REZN",
    name: "Vehicle",
    indexes: [
        { indexId: "Djg2nt0yIs2GY0T29Kud1z", expr: "[VIN]", status: "ONLINE" },
        { indexId: "4tPW3fUhaVhDinRgKRLhGU", expr: "[LicensePlateNumber]", status:
 "BUILDING" }
    ],
    status: "ACTIVE"
}
```

若要授與在資料表上執行 PartiQL 陳述式的權限，您可以使用下列 ARN 格式指定資料表資源。

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

以下是資料表識Au1EiThbt8s0z9wM26REZN別碼的資料表 ARN 範例。

```
arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/Au1EiThbt8s0z9wM26REZN
```

使用主控台

您也可以使用 QLDB 主控台來尋找資料表 ARN。

要查找表（控制台）的 ARN

1. 登入 AWS Management Console，然後開啟 Amazon QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在瀏覽窗格中，選擇「分類帳」。
3. 在「分類帳」清單中，選擇您要搜尋其表格 ARN 的分類帳名稱。
4. 在分類帳詳細資訊頁面的「表格」標籤下，找出您要尋找其 ARN 的表格名稱。若要複製 ARN，請選擇旁邊的複製圖示 ()。

標籤表格

您可以標記表格資源。若要管理現有資料表的標籤，請使用 AWS Management Console 或 API 作業TagResourceUntagResource、和ListTagsForResource。如需詳細資訊，請參閱 標記 Amazon QLDB 資源。

> **Note**
>
> 表格資源不會繼承其根分類帳資源的標籤。
> 目前僅在STANDARD權限模式下，分類帳支援在建立時標記表格。

您也可以在建立資料表時使用 QLDB 主控台或在 CREATE TABLE PartiQL 陳述式中指定資料表標籤來定義資料表標籤。下列範例會建立以標籤命名Vehicle的資料表environment=production。

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

在建立表格時標記表格需要存取qldb:PartiQLCreateTable和qldb:TagResource動作。

藉由在建立時為資源建立標籤，您可以消除在資源建立後執行自訂標籤指令碼的必要。標記資料表之後，您可以根據這些標籤控制對資料表的存取。例如，您只能將完整存取權授與具有特定標籤的資料表。如需 JSON 政策範例，請參閱根據表格標籤對所有動作的完整存取權。

使用主控台

您也可以在建立資料表時使用 QLDB 主控台來定義資料表標籤。

若要在建立時標記表格 (主控台)

1. 登入 AWS Management Console，然後開啟 Amazon QLDB 主控台，網址為 https://console.aws.amazon.com/qldb。
2. 在瀏覽窗格中，選擇「分類帳」。
3. 在「分類帳」清單中，選擇您要在其中建立表格的分類帳名稱。
4. 在分類帳詳細資訊頁面的「表格」標籤下，選擇「建立表格」。
5. 在「建立表格」頁面上，執行下列動作：

   - 表格名稱 — 輸入表格名稱。
   - 標籤 — 透過附加標籤做為鍵值配對，將中繼資料新增至表格。您可以將標籤新增至表格，以協助組織和識別它們。

選擇 [新增標籤]，然後視需要輸入任何索引鍵值配對。

6. 當您滿意設定後，請選擇 Create table (建立資料表)。

快速入門教學課程：建立權限原則

本教學將引導您完成步驟，在 IAM 中為 Amazon QLDB 分類帳在許可模式下建立STANDARD許可政策。然後，您可以將權限指派給您的使用者、群組或角色。

如需授予 PartiQL 命令和表格資源許可的 IAM 政策文件的更多範例，請參閱Amazon QLDB 以身分識別為基礎的政策範例。

主題

- 必要條件
- 建立唯讀政策
- 建立完整存取原則
- 建立特定資料表的唯讀原則
- 指派權限

必要條件

在開始之前，請確保您執行以下操作：

1. 如果您尚未這麼做訪問 Amazon QLDB，請依照中的 AWS 設定指示進行。這些步驟包括註冊 AWS 和建立管理使用者。
2. 創建一個新的分類帳，並選擇分類帳的STANDARD權限模式。若要瞭解如何進行，請參閱開始使用主控台步驟 1：建立新的分類帳中的或Amazon QLDB 分類帳的基本操作。

建立唯讀政策

若要使用 JSON 政策編輯器在標準權限模式下為分類帳中的所有資料表建立唯讀政策，請執行下列動作：

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側的導覽欄中，選擇 Policies (政策)。

如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。

3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 選擇 JSON 標籤。
5. 複製並貼上下列 JSON 政策文件。此範例政策會授與分類帳中所有表格的唯讀存取權。

   若要使用此原則，請以您自己的資訊取代 *us-east-1*、*123456789012*，並在此範例中。*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. 選擇檢閱政策。

> **Note**
>
> 您可以隨時切換 Visual editor (視覺化編輯器) 與 JSON 標籤。不過，如果您進行更改或在 Visual editor (視覺編輯工具) 索引標籤中選擇 Review policy (檢閱政策)，IAM 可能會調整

您的政策結構以針對視覺編輯工具進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在 Review policy (檢閱政策) 頁面上，為您正在建立的政策輸入選用的 Name (名稱) 與 Description (描述)。檢閱政策 Summary (摘要) 來查看您的政策所授予的許可。然後選擇 Create policy (建立政策) 來儲存您的工作。

建立完整存取原則

若要在標準權限模式下為 QLDB 分類帳中的所有表格建立完整存取原則，請執行下列動作：

- 使用下列原則文件重複上述步驟。此範例原則會使用萬用字元 (*) 來涵蓋分類帳下的所有 PartiQL 動作和所有資源，授予對分類帳中所有資料表的所有 PartiQL 命令的存取權。

  Warning

  這是使用萬用字元 (*) 允許所有 PartiQL 動作的範例，包括 QLDB 分類帳中所有資料表的管理和讀取/寫入作業。相反地，最佳做法是明確指定每個要授與的動作，以及只指定該使用者、角色或群組需要的項目。

  若要使用此原則，請以您自己的資訊取代 *us-east-1*、*123456789012*，並在此範例中。*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"
      ]
    }
  ]
}
```

建立特定資料表的唯讀原則

若要在標準權限模式下為 QLDB 分類帳中的特定表格建立唯讀存取原則，請執行下列動作：

1. 使用 AWS Management Console 或查詢系統目錄表格來尋找表格`information_schema.user_tables`的 ARN。如需說明，請參閱尋找資料表識別碼和 ARN。
2. 您可以使用表格 ARN 來建立允許表格唯讀存取的原則。若要這麼做，請使用下列原則文件重複上述步驟。

   此範例原則只會授與指定資料表的唯讀存取權。在此範例中，資料表 ID 為`Au1EiThbt8s0z9wM26REZN`。若要使用此原則，請在範例中以您自己的資訊取代 *us-east-1*、*123456789012 # AU EiThbt 1 8* S0Z9W9WM26reZn。*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/Au1EiThbt8s0z9wM26REZN"
```

```
        ]
      }
    ]
}
```

指派權限

建立 QLDB 權限原則之後，您可以依照下列方式指派權限。

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 使用者和群組位於 AWS IAM Identity Center：

  建立權限合集。請按照 AWS IAM Identity Center 使用者指南 中的 建立權限合集 說明進行操作。
- 透過身分提供者在 IAM 中管理的使用者：

  建立聯合身分的角色。請按照 IAM 使用者指南 的 為第三方身分提供者 (聯合) 建立角色 中的指示進行操作。
- IAM 使用者：
  - 建立您的使用者可擔任的角色。請按照 IAM 使用者指南 的 為 IAM 使用者建立角色 中的指示進行操作。
  - (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循 IAM 使用者指南的新增許可到使用者 (主控台)中的指示。

Amazon QLDB 以身分識別為基礎的政策範例

根據預設，使用者和角色沒有建立或修改 QLDB 資源的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行工作。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

如需有關 QLDB 定義的動作和資源類型的詳細資訊，包括每種資源類型的 ARN 格式，請參閱服務授權參考中的 Amazon QLDB 的動作、資源和條件金鑰。

內容

政策最佳實務

以身分識別為基礎的原則會決定使用者是否可以建立、存取或刪除您帳戶中的 QLDB 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們可用在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授與對服務動

作的存取權 (如透過特定 AWS 服務) 使用 AWS CloudFormation。如需更多資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。

- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。若要在呼叫 API 作業時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

使用 QLDB 主控台

若要存取 Amazon QLDB 主控台，您必須擁有最少一組許可。這些權限必須允許您列出和檢視有關. AWS 帳戶如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

您不需要為僅對 AWS CLI 或 AWS API 進行呼叫的使用者允許最低主控台權限。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

若要確保使用者和角色具有 QLDB 主控台及其所有功能的完整存取權，請將下列 AWS 受管理的原則附加至實體。如需詳細資訊AWS Amazon QLDB 的受管政策，請參閱 IAM 使用者指南中的和向使用者新增許可。

```
AmazonQLDBConsoleFullAccess
```

查詢記錄權限

除了 QLDB 權限之外，某些主控台功能還需要資料庫查詢中繼資料服務 (服務前置詞:dbqms) 的權限。這是一項僅限內部使用的服務，可在 QLDB 和其他主控台查詢編輯器上管理您最近和儲存的查詢。AWS 服務如需 DBQMS API 動作的完整清單，請參閱服務授權參考中的資料庫查詢中繼資料服務。

要允許查詢歷史記錄權限，您可以使用 AWS 管理策略 Amazon ConsoleFullAccess QLDB。此原則會使用萬用字元 (dbqms:*) 來允許所有資源的所有 DBQMS 動作。

或者，您可以建立自訂 IAM 政策，並包含下列 DBQMS 動作。QLDB 主控台上的 PartiQL 查詢編輯器需要使用這些動作進行查詢歷程記錄功能的權限。

```
dbqms:CreateFavoriteQuery
dbqms:CreateQueryHistory
dbqms:DeleteFavoriteQueries
dbqms:DeleteQueryHistory
dbqms:DescribeFavoriteQueries
dbqms:DescribeQueryHistory
dbqms:UpdateFavoriteQuery
```

無查詢歷史記錄的完整訪問控制台

若要允許在沒有任何查詢歷程記錄許可的情況下完整存取 QLDB 主控台，您可以建立排除所有 DB QMS 動作的自訂 IAM 政策。例如，下列原則文件允許 AWS 受管理原則 AmazonQldb 所授與的相同權限ConsoleFullAccess，但以服務前置詞開頭的動作除外。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"
        }
      }
    }
  ]
}
```

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此原則包含在主控台上或以程式設計方式使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
    "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": "*"
        }
    ]
}
```

執行資料交易

若要透過在分類帳上執行 PartiQL 陳述式與 QLDB 交易資料 API (QLDB 工作階段) 互動，您必須授與 API 動作的權限。SendCommand下列 JSON 文件是政策範例，該政策僅授與分類帳上 SendCommand API 動作的權限myExampleLedger。

若要使用此原則，請以您自己的資訊取代 *us-east-1*、*123456789012*，並在此範例中。*myExampleLedger*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        }
    ]
}
```

如果myExampleLedger使用ALLOW_ALL權限模式，則此原則會授與在分類帳中的任何資料表上執行所有 PartiQL 命令的權限。

您也可以使用 AWS 受管理的政策來授與所有 QLDB 資源的完整存取權。如需詳細資訊，請參閱 AWS Amazon QLDB 的受管政策。

PartiQL 動作和資料表資源的標準權限

對於STANDARD許可模式下的分類帳，您可以參考以下 IAM 政策文件，作為授予適當 PartiQL 許可的範例。如需每個 PartiQL 命令所需權限的清單，請參閱. PartiQL 參考

主題

- 完全存取所有動作
- 根據表格標籤對所有動作的完整存取權
- 讀/寫存取
- 唯讀存取
- 特定資料表的唯讀存取權
- 允許存取建立資料表
- 允許根據要求標記建立資料表的存取權

完全存取所有動作

下列 JSON 政策文件授與在*myExampleLedger*中的所有資料表上使用所有 PartiQL 命令的完整存取權。此原則產生的效果與使用分類帳的ALLOW_ALL權限模式相同。

若要使用此原則，請以您自己的資訊取代 *us-east-1*、*123456789012*，並在此範例中。*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"
            ]
        }
    ]
}
```

根據表格標籤對所有動作的完整存取權

下列 JSON 政策文件使用以表格資源標籤為基礎的條件，以授與對中*myExampleLedger*的所有表格使用所有 PartiQL 命令的完整存取權。只有當資料表標籤environment具有值時，才會授與權限development。

> **Warning**
>
> 這是使用萬用字元 (*) 允許所有 PartiQL 動作的範例，包括 QLDB 分類帳中所有資料表的管理和讀取/寫入作業。相反地，最佳做法是明確指定每個要授與的動作，以及只指定該使用者、角色或群組需要的項目。

若要使用此原則，請以您自己的資訊取代 *us-east-1*、*123456789012*，並在此範例中。*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" }
            }
        }
    ]
}
```

讀/寫存取

下列 JSON 政策文件授與選取、插入、更新和刪除中所有資料表資料的權限myExampleLedger。此原則不會授與編輯資料或變更結構描述的權限，例如建立和刪除資料表和索引。

> **Note**
>
> UPDATE陳述式需要對正在修改之表格上的qldb:PartiQLUpdate和qldb:PartiQLSelect動作的權限。當您執行UPDATE陳述式時，除了更新作業之外，它還會執行讀取作業。需要這兩種動作可確保只有被允許讀取表格內容的使用者才能獲得UPDATE權限。

同樣地，DELETE陳述式需要qldb:PartiQLDelete和qldb:PartiQLSelect動作的權限。

若要使用此原則，請以您自己的資訊取代 *us-east-1*、*123456789012*，並在此範例中。*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"
            ]
        }
    ]
}
```

唯讀存取

下列 JSON 政策文件會授與中所有資料表的唯讀權限myExampleLedger。若要使用此原則，請以您自己的資訊取代 *us-east-1*、*123456789012*，並在此範例中。*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"
            ]
        }
    ]
}
```

特定資料表的唯讀存取權

下列 JSON 政策文件授與中特定資料表的唯讀權限myExampleLedger。在此範例中，資料表 ID 為Au1EiThbt8s0z9wM26REZN。

若要使用此原則，請在範例中以您自己的資訊取代 *us-east-1*、*123456789012 # AU EiThbt 1 8* S0Z9W9WM26reZn。*myExampleLedger*

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

允許存取建立資料表

下列 JSON 政策文件授與在中建立表格的權限myExampleLedger。此動qldb:PartiQLCreateTable作需要表格資源類型的權限。不過，當您執行CREATE TABLE陳述式時，新資料表的資料表識別碼並不知道。因此，授與qldb:PartiQLCreateTable權限的政策必須在資料表 ARN 中使用萬用字元 (*) 來指定資源。

若要使用此原則，請以您自己的資訊取代 *us-east-1*、*123456789012*，並在此範例中。*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/*"
            ]
        }
    ]
}
```

允許根據要求標記建立資料表的存取權

下列 JSON 政策文件使用根據aws:RequestTag內容索引鍵的條件來授與在中建立表格的權限myExampleLedger。只有在要求標記environment具有值時，才會授與權限development。在建立表格時標記表格需要存取qldb:PartiQLCreateTable和qldb:TagResource動作。若要瞭解如何在建立表格時標記表格，請參閱標籤表格。

若要使用此原則，請以您自己的資訊取代 *us-east-1*、*123456789012*，並在此範例中。*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" }
            }
        }
    ]
}
```

將日誌匯出到 Amazon S3 儲存貯體

步驟 1：QLDB 日誌匯出權限

在下列範例中，您授與使用者對 QLDB 分類帳資源執行`qldb:ExportJournalToS3`動作的 AWS 帳戶 權限。您也授與許可，以便`iam:PassRole`對要傳遞至 QLDB 服務的 IAM 角色資源執行動作。所有分錄匯出請求都需要此功能。

`#################### us-east-1#123456789012 ######myExampleLedger`

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

步驟 2：Amazon S3 存儲桶許可

在下列範例中，您可以使用 IAM 角色授與 QLDB 存取權，以寫入其中一個 Amazon S3 儲存貯體。`DOC-EXAMPLE-BUCKET`所有 QLDB 分錄匯出也必須執行此動作。

除了授與`s3:PutObject`權限之外，原則還會`s3:PutObjectAcl`授與設定物件存取控制清單 (ACL) 權限的權限。

若要使用此政策，請將範例中的文件範例儲存貯體取代為您的 Amazon S3 儲存貯體名稱。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "QLDBJournalExportS3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

然後，您可以將此許可政策附加到 QLDB 可假設存取 Amazon S3 儲存貯體的 IAM 角色。下列 JSON 文件是允許 QLDB 僅針對帳戶中任何 QLDB 資源採用 IAM 角色的信任政策範例。123456789012

若要使用此政策，請以您自己的資訊取代範例中的 *us-east-1* 和 *1234567890* 12。

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

將日誌串流至 Kinesis Data Streams

步驟 1：QLDB 日誌串流權限

在下列範例中，您授與使用者對分類帳中所有 QLDB 串流子資源執行`qldb:StreamJournalToKinesis`動作的 AWS 帳戶 權限。您也授與許可，以

便iam:PassRole對要傳遞至 QLDB 服務的 IAM 角色資源執行動作。所有日誌串流請求都需有此許可。

若要使用此原則，請以您自己的資訊取代 *us-east-1*、*123456789012* *myExampleLedger*，並在此範例中。*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"
               }
            }
        }
    ]
}
```

步驟 2：Kinesis Data Streams 權限

在下列範例中，您可以使用 IAM 角色授與 QLDB 存取權，將資料記錄寫入 Amazon Kinesis 資料串流。*stream-for-qldb*所有 QLDB 日誌串流也需要這個動作。

若要使用此原則，請以您自己的資訊取代 *us-east-1*、*123456789012*，並在此範例中。*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"
        }
    ]
}
```

然後，您可以將此許可政策附加到 QLDB 可假設存取 Kinesis 資料串流的 IAM 角色。下列 JSON 文件是信任政策範例，該政策允許 QLDB 僅針對總帳帳戶123456789012中的任何 QLDB 串流假設 IAM 角色。myExampleLedger

若要使用此原則，請以您自己的資訊取代 *us-east-1*、*123456789012*，並在此範例中。*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"
                }
            }
        }
    ]
}
```

根據標籤更新 QLDB 分類帳

您可以使用身分型原則中的條件，根據標籤來控制 QLDB 資源的存取。此範例顯示如何建立允許更新分類帳的策略。不過，只有當分類帳標記Owner具有該使用者的使用者名稱值時，才會授與權限。此政策也會授予在主控台完成此動作的必要許可。

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

您可以將此政策連接到您帳戶中的 使用者。如果名為的使用者`richard-roe`嘗試更新 QLDB 分類帳，則必須將分類帳加上標記`Owner=richard-roe`或。`owner=richard-roe`否則，他便會被拒絕存取。條件標籤金鑰 `Owner` 符合 `Owner` 和 `owner`，因為條件金鑰名稱不區分大小寫。如需更多資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。

預防跨服務混淆代理人

混淆代理人問題屬於安全性問題，其中沒有執行動作許可的實體可以強制具有更多許可的實體執行該動作。在中 AWS，跨服務模擬可能會導致混淆的副問題。

在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。為了避免混淆的副問題，AWS 提供的工具可協助您保護所有服務的資料，其中包含已授予您帳戶資源存取權的服務主體。

我們建議在資源政策中使用`aws:SourceArn`和`aws:SourceAccount`全域條件內容金鑰，以限制 Amazon QLDB 為資源提供其他服務的許可。如果您同時使用全域條件內容索引鍵，則`aws:SourceAccount`值和帳戶在相同政策陳述式中使用時，`aws:SourceArn`值和帳戶必須使用相同的帳戶 ID。

下表列出ExportJournalToS3和 StreamsJournalToKinesisQLDB API aws:SourceArn 作業的可能值。這些操作在此安全問題的範圍內，因為它們會呼叫 AWS Security Token Service (AWS STS) 來承擔您指定的 IAM 角色。

API 操作	呼叫的服務	AWS：SourceArn
ExportJournalToS3	AWS STS (AssumeRole)	允許 QLDB 擔任帳戶中任何 QLDB 資源的角色： `arn:aws:qldb: us-east-1 :123456789012 :*` 目前，QLDB 僅支援此萬用字元 ARN 進行分錄匯出。

StreamsJournalToKinesis	AWS STS (AssumeRole)	允許 QLDB 擔任特定 QLDB 串流的角色：
		`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /IiPT4brpZCqCq3f4MTHbYy`
		注意：您只能在串流資源建立之後在 ARN 中指定串流 ID。使用此 ARN，您可以允許角色僅用於單一 QLDB 串流。
		允許 QLDB 擔任分類帳之任何 QLDB 串流的角色：
		`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /*`
		允許 QLDB 擔任帳戶中任何 QLDB 串流的角色：
		`arn:aws:qldb: us-east-1 :123456789012 :stream/*`
		允許 QLDB 擔任帳戶中任何 QLDB 資源的角色：
		`arn:aws:qldb: us-east-1 :123456789012 :*`

防範混淆代理人問題的最有效方法是使用 aws:SourceArn 全域條件內容索引鍵，以及資源的完整 ARN。如果您不知道資源的完整 ARN，或者您要指定多個資源，請針對 ARN 的未知部分使用萬用字元 (*) 的aws:SourceArn全域內容條件索引鍵，例如。arn:aws:qldb:us-east-1:*123456789012*:*

IAM 角色的下列信任政策範例說明如何使用aws:SourceArn和aws:SourceAccount全域條件內容金鑰來避免混淆的副問題。透過此信任原則，QLDB 只能擔任分類帳帳戶123456789012中任何 QLDB 串流的角色。myExampleLedger

若要使用此原則，請以您自己的資訊取代 *us-east-1*、*123456789012*，並在此範例中。*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 Amazon QLDB 的受管政策

受 AWS 管理的策略是由建立和管理的獨立策略 AWS。 AWS 受管理的策略旨在為許多常見使用案例提供權限，以便您可以開始將權限指派給使用者、群組和角色。

請記住， AWS 受管理的政策可能不會為您的特定使用案例授與最低權限權限，因為這些權限可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法變更受 AWS 管理策略中定義的權限。如果 AWS 更新 AWS 受管理原則中定義的權限，則此更新會影響附加原則的所有主體識別 (使用者、群組和角色)。 AWS 當新的啟動或新 AWS 服務 的 API 操作可用於現有服務時，最有可能更新 AWS 受管理策略。

如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

如需這些 AWS 受管理原則中 QLDB API 作業的詳細資訊，請參閱. Amazon QLDB API 參考參考

主題

- AWS 管理策略:亞馬遜 QLDB ReadOnly
- AWS 管理策略:亞馬遜 QLDB FullAccess
- AWS 管理策略:亞馬遜 QLDB ConsoleFullAccess
- 受管理政策的 QLDB 更新 AWS

AWS 管理策略:亞馬遜 QLDB ReadOnly

使用 AmazonQLDB ReadOnly 政策將唯讀權限授與所有 QLDB 資源。您可以將此政策附加到 IAM 身分。

許可詳細資訊

此原則包含qldb服務的下列權限。

- 允許主參與者描述和列出所有 QLDB 資源及其標籤。這些資源包括分類帳、Amazon S3 匯出任務，以及到 Kinesis Data Streams。
- 允許主參與者從任何分類帳中的日誌取得區塊、摘要或修訂，以密碼方式驗證資料。
- 不允許主參與者在任何分類帳中的任何表格上執行任何 PartiQL 命令。

AWS 管理策略:亞馬遜 QLDB FullAccess

使用 AmazonQLDB FullAccess 原則，透過 QLDB API 或授與所有 QLDB 資源的完整管理權限。AWS CLI您可以將此政策附加到 IAM 身分。

許可詳細資訊

此政策包含以下許可。

- qldb
  - 允許主參與者建立、描述、列出及管理所有 QLDB 資源及其標籤。這些資源包括分類帳、Amazon S3 匯出任務，以及到 Kinesis Data Streams。
  - 允許主體使用 QLDB 驅動程式或 QLDB 殼層，在任何總帳中的所有表格上執行所有 PartiQL 命令。

  - 允許主參與者從任何分類帳中的日誌取得區塊、摘要或修訂，以密碼方式驗證資料。
- `iam`— 允許主體將您帳戶中的任何 IAM 角色資源傳遞至 QLDB 服務。所有日誌匯出和串流請求都需要此功能。

AWS 管理策略:亞馬遜 QLDB ConsoleFullAccess

使用 AmazonQLDB ConsoleFullAccess 原則 AWS Management Console，透過 QLDB API 或. AWS CLI您可以將此政策附加到 IAM 身分。

許可詳細資訊

此政策包含以下許可。

- `qldb`
  - 允許主參與者建立、描述、列出及管理所有 QLDB 資源及其標籤。這些資源包括分類帳、Amazon S3 匯出任務，以及到 Kinesis Data Streams。
  - 允許主體使用 QLDB 主控台、QLDB 驅動程式或 QLDB 殼層，在任何總帳中的所有表格上執行所有 PartiQL 命令。
  - 允許主參與者使用 QLDB 主控台，在任何分類帳中插入範例應用程式資料。
  - 允許主參與者從任何分類帳中的日誌取得區塊、摘要或修訂，以密碼方式驗證資料。
- `dbqms`— 允許主參與者使用「資料庫查詢中繼資料服務」中的所有動作。這是一項僅供內部使用的服務，QLDB 主控台需要為 PartiQL 查詢編輯器建立、描述和管理最近和儲存的查詢。
- `kinesis`— 允許主體描述和列出 Amazon Kinesis Data Streams 資源。這些資源是 QLDB 串流資源可以寫入資料的目標目的地。
- `iam`— 允許主體將您帳戶中的任何 IAM 角色資源傳遞至 QLDB 服務。所有日誌匯出和串流請求都需要此功能。

受管理政策的 QLDB 更新 AWS

檢視有關 QLDB AWS 受管理原則的更新詳細資訊，因為此服務開始追蹤這些變更。如需有關此頁面變更的自動警示，請訂閱 QLDB 版本歷史記錄頁面上的 RSS 摘要。

變更	描述	日期
亞馬遜 QLDBFullAccess, 亞馬遜 QLDB — 更新Console FullAccess到現有政策	QLDB 增加了新的權限，允許主參與者在權限模式下編輯所有分類帳中的文件修訂版本。STANDARD	2022 年 11 月 4 日
亞馬遜 QLDBFullAccess, 亞馬遜 QLDB — 更新Console FullAccess到現有政策	QLDB 新增了許可，允許主體將您帳戶中的任何 IAM 角色資源傳遞至 QLDB 服務。所有日誌匯出和串流請求都需要此功能。	2021 年 9 月 2 日
亞馬遜 QLDB ReadOnly — 更新到現有策略	QLDB 移除先前列出兩次的重複`qldb:GetBlock` 動作，並重新排序欄位，使其出現在`"Effect"`欄位之前。`"Action"`	2021 年 7 月 1 日
亞馬遜 QLDBFullAccess, 亞馬遜 QLDB — 更新Console FullAccess到現有政策	QLDB 新增了新權限，允許主體更新所有分類帳中的權限模式，並在新權限模式下執行所有分類帳中的所有 PartiQL 命令。STANDARD STANDARD權限模式支援 PartiQL 命令的資料表層級存取控制和精細度。為了促進新的許可模式，QLDB 針對 PartiQL 命令類型引入了一組 IAM 動作，以及針對 QLDB 表格資源引入了 Amazon 資源名稱 (ARN)。這兩個原則已更新，以包含新的 PartiQL 動作，以授與STANDARD分類帳的完整存取權。	2021 年 5 月 27 日

變更	描述	日期
QLDB 開始追蹤變更	QLDB 開始追蹤其 AWS 受管理原則的變更。	2021 年 3 月 1 日

疑難排解 Amazon QLDB 身分識別和存取

使用下列資訊可協助您診斷和修正使用 QLDB 和 IAM 時可能遇到的常見問題。

主題

- 我沒有在 QLDB 中執行動作的授權
- 我沒有授權執行 iam：PassRole
- 我想允許我以外的人訪問我 AWS 帳戶 的 QLDB 資源

我沒有在 QLDB 中執行動作的授權

如果 AWS Management Console 告訴您您沒有執行動作的授權，則您必須聯絡您的管理員以尋求協助。您的管理員是為您提供簽署憑證的人員。

下列範例錯誤會在 `mateojackson` 使用者嘗試使用主控台檢視一個虛構 *`myExampleLedger`* 資源的詳細資訊，但卻無虛構 `qldb:`*`DescribeLedger`* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 qldb:DescribeLedger on resource: myExampleLedger
```

在此情況下，Mateo 會請求管理員更新他的政策，允許他使用 *`myExampleLedger`* 動作存取 `qldb:`*`DescribeLedger`* 資源。

我沒有授權執行 iam：PassRole

如果您收到無權執行`iam:PassRole`動作的錯誤訊息，您必須更新原則，才能將角色傳遞給 QLDB。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為的 IAM 使用者`marymajor`嘗試使用主控台在 QLDB 中執行動作時，就會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 `iam:PassRole` 動作。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

如需有關日誌匯出或串流作業特定錯誤的疑難排解指引，請參閱Amazon QLDB 故障診斷。

我想允許我以外的人訪問我 AWS 帳戶 的 QLDB 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要瞭解 QLDB 是否支援這些功能，請參閱。Amazon QLDB 如何與 IAM 合作
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱《IAM 使用者指南》中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何將資源存取權提供給第三方 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。
- 若要了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱 IAM 使用者指南中的 IAM 角色與資源型政策的差異。

在 Amazon QLDB 中進行記錄和監控

監控是維護 Amazon QLDB 和解決方案的可靠性、可用性和效能的重要組成部分 AWS 。您應該從 AWS 解決方案的所有部分收集監視資料，以便在發生多點失敗時更輕鬆地偵錯。不過，在開始監視 QLDB 之前，您應該建立一個監視計劃，其中包含下列問題的答案：

- 監控目標是什麼？
- 要監控哪些資源？
- 監控這些資源的頻率為何？

- 要使用哪些監控工具？
- 誰將執行監控任務？
- 發生問題時應該通知誰？

下一個步驟是透過測量不同時間和不同負載條件下的效能，為環境中的正常 QLDB 效能建立基準。監控 QLDB 時，請儲存歷史監視資料，以便您可以將其與目前的效能資料進行比較，識別正常的效能模式和效能異常，並設計解決問題的方法。

若要建立基準，您至少必須監控下列項目：

- 讀取和寫入 I/O 和儲存空間，以便您可以追蹤分類帳的使用模式以進行計費。
- 命令延遲，以便您可以在運行數據操作時跟踪分類帳的性能。
- 例外狀況，以便您可以判斷是否有任何要求導致錯誤。

主題

監控工具

AWS 提供各種可用來監控 Amazon QLDB 的工具。您可以設定其中一些工具來進行監控，但有些工具需要手動介入。建議您盡可能自動化監控任務。

主題

自動化監控工具

您可以使用以下自動監控工具來觀看 QLDB 並在出現問題時報告：

- Amazon CloudWatch 警示 — 觀看您指定期間內的單一指標，並根據指定臨界值在多個時段內相對於指定閾值的指標值執行一或多個動作。動作是傳送至亞馬遜簡單通知服務 (Amazon SNS) 主題或

Amazon EC2 Auto Scaling 政策的通知。 CloudWatch 警示不會僅因為處於特定狀態而叫用動作；狀態必須已變更並維持指定數目的期間。如需詳細資訊，請參閱 使用 Amazon 監控 CloudWatch。

- Amazon CloudWatch 日誌 — 監控、存放和存取來自 AWS CloudTrail 或其他來源的日誌檔。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的監控日誌檔。
- Amazon E CloudWatch vents — 匹配事件並將其路由到一個或多個目標函數或串流，以進行變更、擷取狀態資訊並採取糾正措施。有關更多信息，請參閱 Amazon 用 CloudWatch 戶指南中的 Amazon CloudWatch 事件是什麼。
- AWS CloudTrail 記錄監控 — 在帳戶之間共用記錄檔、即時監控記 CloudTrail 錄檔案，方法是將記錄檔傳送至 CloudWatch 記錄檔、使用 Java 撰寫記錄處理應用程式，以及驗證您的記錄檔在傳送之後未變更 CloudTrail。若要取得更多資訊，請參閱《使用指南》中的〈AWS CloudTrail 使用 CloudTrail 記錄檔〉。

手動監控工具

監視 QLDB 的另一個重要部分是手動監視 CloudWatch 警報未涵蓋的項目。QLDB、 CloudWatch Trusted Advisor、和其他 AWS Management Console 儀表板可提供您環境狀態的 at-a-glance AWS 檢視。我們建議您也檢查 Amazon QLDB 上的日誌文件。

- QLDB 儀表板顯示以下內容：
  - 讀取和寫入I/Os
  - 日誌和索引儲存
  - 命令延遲
  - 例外狀況
- CloudWatch 首頁會顯示下列內容：
  - 目前警示與狀態
  - 警示與資源的圖表
  - 服務運作狀態

  此外，您可以使用執行 CloudWatch 以下操作：
  - 建立自定儀表板來監控您注重的服務
  - 用於疑難排解問題以及探索驅勢的圖形指標資料。
  - 搜尋並瀏覽您的所有資 AWS 源指標
  - 建立與編輯要通知發生問題的警示

使用 Amazon 監控 CloudWatch

您可以使用 Amazon QLDB 來監控 Amazon QLDB CloudWatch，它會收集來自 Amazon QLDB 的原始資料，並將其處理成可讀的指標。 near-real-time 它會記錄這些統計數據兩週，以便您可以訪問歷史信息，並更好地了解 Web 應用程序或服務的執行情況。依預設，QLDB 指標資料會 CloudWatch 在 1 或 15 分鐘內自動傳送至。如需詳細資訊，請參閱什麼是 Amazon CloudWatch、Amazon CloudWatch 活動和 Amazon CloudWatch 日誌？ 在 Amazon 用 CloudWatch 戶指南。

主題

- 如何使用 QLDB 指標？
- Amazon QLDB 指標和維度
- 創建 CloudWatch 警報以監控 Amazon QLDB

如何使用 QLDB 指標？

QLDB 報告的指標提供您可以用不同方式分析的資訊。下列清單顯示一些常見的指標用途。這些是協助您開始的建議，而不是完整清單。

- 您可以監視`JournalStorage`並`IndexedStorage`在指定的時間段內跟踪分類帳消耗的磁盤空間。
- 您可以監視`ReadIOs`並`WriteIOs`在指定的期間內追蹤分類帳正在處理的請求數目。
- 您可以監視`CommandLatency`以跟踪分類帳的數據操作性能，並分析導致最大延遲的命令類型。

Amazon QLDB 指標和維度

當您與 Amazon QLDB 互動時，它會將下列指標和維度傳送到。 CloudWatch儲存指標每 15 分鐘報告一次，而所有其他指標則會每分鐘彙總並報告一次。您可以使用下列程序來檢視 QLDB 的測量結果。

使用 CloudWatch 主控台檢視指標

指標會先依服務命名空間分組，再依各命名空間內不同的維度組合分類。

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 如有必要請變更區域。在導覽列上，選擇資 AWS 源所在的地區。如需更多詳細資訊，請參閱區域與端點。
3. 在導覽窗格中，選擇 指標。
4. 在「所有測量結果」標籤下，選擇 QLDB。

若要使用 AWS CLI

- 在命令提示中，使用下列命令。

```
aws cloudwatch list-metrics --namespace "AWS/QLDB"
```

CloudWatch 顯示 QLDB 的下列測量結果。

Amazon QLDB 維度和指標

這裡列出了 Amazon QLDB 傳送給 Amazon CloudWatch 的指標和維度。

QLDB 測量結果

指標	描述
JournalStorage	分類帳日誌使用的磁碟空間總量，每隔 15 分鐘報告一次。該日誌包含對數據的所有更改的完整，不可變和可驗證的歷史記錄。 單位：Bytes 維度：LedgerName
IndexedStorage	分類帳表格、索引和索引歷史記錄所使用的磁碟空間總量，每隔 15 分鐘報告一次。索引儲存包含針對高效能查詢最佳化的總帳資料。 單位：Bytes 維度：LedgerName
ReadIOs	讀取 I/O 要求的數目，以一分鐘的間隔報告。這會擷取所有類型的讀取作業，包括資料交易、驗證請求、分錄匯出和分錄串流。 單位：Count 維度：LedgerName
WriteIOs	以一分鐘間隔報告的寫入 I/O 要求數目。

指標	描述
	單位：Count 維度：LedgerName
CommandLatency	資料作業所使用的時間量，以一分鐘的間隔報告。 單位：Milliseconds 維度：CommandType, LedgerName
IsImpaired	指出 Kinesis 資料串流的日誌串流是否受損的旗標，並以一分鐘的間隔報告。的值1表示資料流處於受損狀態，並0指出其他狀態。 單位:Boolean(0 或 1) 維度：LedgerName, StreamId
OccConflictExceptions	對 QLDB 產生的要求數目。OccConflictException 如需樂觀並行控制 (OCC) 的相關資訊，請參閱。Amazon QLLDB 並行模型 單位：Count
Session4xxExceptions	對 QLDB 產生 HTTP 4xx 錯誤的要求數目。 單位：Count
Session5xxExceptions	對 QLDB 產生 HTTP 5xx 錯誤的要求數目。 單位：Count
SessionRateExceededExceptions	對 QLDB 產生的要求數目。SessionRateExceededException 單位：Count

QLDB 測量結果的維度

QLDB 的量度由帳戶、分類帳名稱、串流 ID 或命令類型的值限定。您可以使用 CloudWatch 主控台來擷取下表中任何維度的 QLDB 資料。

維度	描述
LedgerName	此維度會將資料限制在特定分類帳。此值可以是目前 AWS 區域與目前中的任何分類帳名稱 AWS 帳戶。
StreamId	此維度會將資料限制在特定的分錄串流。此值可以是目前 AWS 區域 和目前分類帳的任何串流 ID AWS 帳戶。
CommandType	此維度會將資料限制為下列其中一個 QLDB 資料 API 命令： • AbortTransaction • CommitTransaction • EndSession • ExecuteStatement • FetchPage • StartSession • StartTransaction 若要瞭解 QLDB 如何使用這些指令來管理資料作業，請參閱。驅動程式的工作階段管理

創建 CloudWatch 警報以監控 Amazon QLDB

您可以建立 Amazon CloudWatch 警示，在警示狀態變更時傳送 Amazon 簡單通知服務 (Amazon SNS) 訊息。警示會在您指定的期間監看單一指標。警示會根據在數個期間與指定閾值相關的指標值，來執行一個或多個動作。此動作是傳送到 Amazon SNS 主題或 Auto Scaling 政策的通知。

警示只會呼叫持續狀態變更的動作。 CloudWatch 警報不會僅僅因為它們處於特定狀態而叫用動作。狀態必須已變更，且在指定的期間數內維持此狀態。

如需建立 CloudWatch 警示的詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用 Amazon CloudWatch 警示。

使用事件自動化 Amazon QLDB CloudWatch

Amazon E CloudWatch vents 可讓您自動執行 AWS 服務 並自動回應系統事件，例如應用程式可用性問題或資源變更。來自的事件 AWS 服務 會以近乎即時的方式傳送至「 CloudWatch 事件」。您可編寫簡單的規則，來指示您在意的事件，以及當事件符合規則時所要自動執行的動作。可以自動觸發的動作如下：

- 調用函數 AWS Lambda
- 調用 Amazon EC2 執行命令
- 將事件轉傳至 Amazon Kinesis Data Streams
- 啟動 AWS Step Functions 狀態機
- 通知 Amazon SNS 主題或 Amazon SQS 佇列

每當您 AWS 帳戶 變更總帳資源狀態時，Amazon QLDB 就會向 CloudWatch 事件報告事件。目前只針對 QLDB 分類帳資源發出事件的保證。 at-least-once

以下是 QLDB 報告的事件範例，其中分類帳的狀態變更為。`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"
    }
}
```

搭配 QLDB 使用 CloudWatch 事件的一些範例可能包括但不限於下列項目：

- 每當最初在`CREATING`狀態中創建新分類帳並最終變成時，激活 Lambda 函數`ACTIVE`。
- 當分類帳狀態變更為，然後變為`DELETING`時，通知 Amazon SNS 主題。`DELETED`

如需詳細資訊，請參閱 Amazon CloudWatch 事件使用者指南。

使用記錄 Amazon QLDB API 呼叫 AWS CloudTrail

Amazon QLDB 與服務整合 AWS 服務 在一起 AWS CloudTrail，可提供使用者、角色或 QLDB 中所採取的動作記錄的服務。 CloudTrail 擷取 QLDB 的所有資源管理 API 呼叫做為事件。擷取的呼叫包括來自 QLDB 主控台的呼叫，以及對 QLDB API 作業的程式碼呼叫。如果您建立追蹤，您可以啟用連續交付 CloudTrail 事件到 Amazon Simple Storage Service (Amazon S3) 儲存貯體，包括 QLDB 的事件。如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台上的最新事件。使用收集的資訊 CloudTrail，您可以判斷向 QLDB 發出的要求、提出要求的 IP 位址、提出要求的人員、提出要求的時間以及其他詳細資訊。

若要進一步了解 CloudTrail，包括如何設定和啟用它，請參閱AWS CloudTrail 使用者指南。

QLDB 資訊於 CloudTrail

CloudTrail 在您創建帳戶 AWS 帳戶 時啟用。當 QLDB 中發生受支援的活動時，該活動會與事件歷史記錄中的其他 CloudTrail 事件一起記錄在 AWS 服務 事件中。您可以查看，搜索和下載最近的事件 AWS 帳戶。如需詳細資訊，請參閱使用 CloudTrail 事件歷程記錄檢視事件。

若要持續記錄您 AWS 帳戶的事件 (包括 QLDB 的事件)，請建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。依預設，當您在主控台中建立追蹤時，該追蹤會套用至所有的 AWS 區域。追蹤記錄來自 AWS 分區中所有區域的事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。此外，您可以設定其他， AWS 服務 以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。

如需詳細資訊，請參閱 AWS CloudTrail 使用者指南中的以下主題：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 記錄檔
- 從多個帳戶接收 CloudTrail 日誌文件

所有 QLDB 資源管理和非交易資料 API 動作都會記錄下來， CloudTrail 並記錄在 Amazon Q LDB API 參考資料中。例如，呼叫`CreateLedger`、`DescribeLedger`和`DeleteLedger`動作會在 CloudTrail 記錄檔中產生項目。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否使用根或 使用者登入資料提出
- 提出該請求時，是否使用了特定角色或聯合身分使用者的臨時安全憑證
- 請求是否由另一個人提出 AWS 服務

如需詳細資訊，請參閱CloudTrail 使 userIdentity 元素。

瞭解 QLDB 記錄檔項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 Amazon S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示示範這些動作的 CloudTrail 記錄項目：

- `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"
    }
  ]
}
```


Amazon QLDB 的合規驗證

第三方稽核員會評估 Amazon QLDB 的安全性和合規性，做為多個 AWS 合規計劃的一部分，包括但不限於下列各項：

- 系統和組織控制 (SOC)
- 支付卡產業 (PCI)
- 國際標準組織 (ISO)

- 資訊系統安全管理與評估計劃
- 美國健康保險流通與責任法案 (HIPAA)

> **Note**
>
> 這不是 Amazon QLDB 認證的完整列表。

若要瞭解 AWS 服務 是否屬於特定規範遵循方案的範圍內，請參閱AWS 服務 遵循規範計劃方案中的，並選擇您感興趣的合規方案。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用時的合規責任取決 AWS 服務 於資料的敏感性、公司的合規目標以及適用的法律和法規。AWS 提供下列資源以協助遵循法規：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供部署以安全性和合規性 AWS 為重點的基準環境的步驟。
- 在 Amazon Web Services 上架構 HIPAA 安全性與合規性 — 本白皮書說明公司如何使用建立符合 HIPAA 資格的應 AWS 用程式。

  > **Note**
  >
  > 並非所有人 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS 客戶合規指南 — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 中，保 AWS 服務 護指引並對應至安全控制的最佳實務。
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub— 這 AWS 服務 提供了內部安全狀態的全面視圖 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱 Security Hub controls reference。

- Amazon GuardDuty — 透過監控環境中的 AWS 帳戶可疑和惡意活動，藉此 AWS 服務 偵測您的工作負載、容器和資料的潛在威脅。 GuardDuty 可協助您滿足特定合規性架構所要求的入侵偵測需求，如 PCI DSS 等各種合規性需求。
- AWS Audit Manager— 這 AWS 服務 有助於您持續稽核您的 AWS 使用情況，以簡化您管理風險的方式，以及遵守法規和業界標準的方式。

Amazon QLDB 的韌性

AWS 全球基礎架構是圍繞 AWS 區域 和可用區域建立的。 AWS 區域 提供多個實體分離和隔離的可用區域，這些區域與低延遲、高輸送量和高冗餘網路相連。透過可用區域，您所設計與操作的應用程式和資料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域 和可用區域的詳細資訊，請參閱AWS 全域基礎結構。

儲存耐用性

QLDB 日誌儲存功能可在交易認可時同步複寫至多個可用區域。這可確保即使是日誌儲存的完整可用區域失敗，也不會影響資料完整性或維護作用中服務的能力。此外，QLDB 日誌還具有非同步歸檔功能，可用於容錯儲存。此功能可在多個可用區域同時發生儲存故障的可能性極低的情況下支援災難復原。

QLDB 索引儲存由複寫至多個可用區域提供支援。這可確保即使是索引儲存體的完整可用區域失敗，也不會影響資料完整性或維護作用中服務的能力。

資料耐久性功能

除了 AWS 全球基礎架構外，QLDB 還提供下列功能，協助您支援資料復原和備份需求。

QLDB 服務特色

隨需分錄匯出

QLDB 提供隨需分錄匯出功能。透過將分類帳中的日誌區塊匯出至 Amazon S3 儲存貯體，以存取日誌的內容。您可以將此資料用於各種目的，例如資料保留、分析和稽核。如需詳細資訊，請參閱從 Amazon QLDB 匯出日誌資料。

備份和還原

目前不支援匯出的自動還原。匯出可提供基本功能，以您定義的頻率建立額外的資料備援。但是，還原案例取決於應用程式，其中匯出的記錄大概會利用相同或類似的擷取方法，將匯出的記錄寫回新的分類帳。

QLDB 目前不提供專用的備份與還原功能。

日誌串流

QLDB 也提供連續的日誌串流功能。您可以將 QLDB 日誌串流與 Amazon Kinesis 串流平台整合，以處理即時日誌資料。如需詳細資訊，請參閱 從 Amazon QLDB 串流日誌資料。

QLDB 設計特色

QLDB 的設計是為了抵禦邏輯損毀而設計。QLDB 日誌是不可變的，確保所有已認可的交易都保留在日誌中。此外，系統會記錄每個已提交的文件變更，因為這point-in-time可讓您查看分類帳資料的任何非預期變更。

QLDB 目前不提供邏輯損毀案例的自動復原功能。

Amazon QLDB 中的基礎設施安全

作為受管服務，Amazon QLDB 受到 Amazon Web Services：安 AWS 全流程概觀白皮書中所述的全球網路安全程序的保護。

您可以使用 AWS 已發佈的 API 呼叫，透過網路存取 QLDB。用戶端必須支援 Transport Layer Security (TLS) 1.0 或更新版本。建議使用 TLS 1.2 或更新版本。用戶端也必須支援具備完美轉送私密 (PFS) 的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。

此外，請求必須使用與 IAM 主體相關聯的程式設計登入資料來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

您也可以針對 QLDB 使用虛擬私有雲端 (VPC) 端點。介面虛擬私人雲端端點可讓您的 Amazon VPC 資源使用其私有 IP 地址存取 QLDB，而不會暴露於公用網際網路。如需詳細資訊，請參閱 使用界面端點存取 Amazon QLDB ()AWS PrivateLink。

使用界面端點存取 Amazon QLDB ()AWS PrivateLink

您可以使 AWS PrivateLink 用在 VPC 和 Amazon QLDB 之間建立私有連接。您可以像在 VPC 中一樣存取 QLDB，而無需使用網際網路閘道、NAT 裝置、VPN 連線或連線。 AWS Direct Connect VPC 中的執行個體不需要公用 IP 位址即可存取 QLDB。

您可以建立由 AWS PrivateLink提供支援的介面端點來建立此私有連線。我們會在您為介面端點啟用的每個子網中建立端點網路介面。這些是由要求者管理的網路介面，可做為目的地 QLDB 流量的進入點。

如需詳細資訊，請參閱《AWS PrivateLink 指南》中的透過 AWS PrivateLink存取 AWS 服務。

主題

QLDB 的考量

在設定 QLDB 的介面端點之前，請先檢閱指AWS PrivateLink 南中的考量事項。

> **Note**
>
> QLDB 僅支援透過介面端點呼叫 QLDB 工作階段交易資料 API。此 API 僅包含SendCommand操作。在分類帳的STANDARD權限模式下，您可以控制此 API 中特定 PartiQL 動作的權限。

建立 QLDB 的介面端點

您可以使用 Amazon VPC 主控台或 () 建立 QLDB 的介面端點。 AWS Command Line Interface AWS CLI如需詳細資訊，請參閱《AWS PrivateLink 指南》中的建立介面端點。

使用下列服務名稱建立 QLDB 的介面端點：

```
com.amazonaws.region.qldb.session
```

如果您為介面端點啟用私有 DNS，您可以使用其預設的區域 DNS 名稱向 QLDB 發出 API 要求。例如 `session.qldb.us-east-1.amazonaws.com`。

為您的介面端點建立端點政策

端點政策為 IAM 資源，您可將其連接至介面端點。預設端點策略允許透過介面端點完整存取 QLDB。若要控制允許從 VPC 存取 QLDB，請將自訂端點原則附加到介面端點。

端點政策會指定以下資訊：

- 可以執行動作 (AWS 帳戶、使用者和角色) 的主參與者。
- 可執行的動作。
- 可供執行動作的資源。

如需詳細資訊，請參閱《AWS PrivateLink 指南》中的使用端點政策控制對服務的存取。

您也可以使用附加至使用者、群組或角色之策略中的`Condition`欄位，僅允許從指定的介面端點存取。一起使用時，端點政策和 IAM 政策可以將指定分類帳上特定 QLDB 動作的存取限制到指定的介面端點。

端點策略範例：限制對特定 QLDB 分類帳的存取

以下是 QLDB 的自訂端點原則範例。當您將此原則附加至介面端點時，它會授與指定分類帳資源上所有主體之`SendCommand`動作和 PartiQL 唯讀動作的存取權。在此範例中，分類帳必須處於`STANDARD`權限模式。

若要使用此原則，請以您自己的資訊取代 *`us-east-1`*、*`123456789012`*，並在此範例中。*`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"
         ]
      }
   ]
}
```

IAM 政策範例：僅限從特定介面端點存取 QLDB 分類帳

以下是 QLDB 的 IAM 身分型政策範例。將此原則附加至使用者、角色或群組時，僅允許從指定的介面端點SendCommand存取總帳資源。

###################### us-east-1#123456789012 # vpce-1a2b3c4d#myExampleLedger

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

QLDB 的介面端點的可用性

Amazon QLDB 支援介面端點，其中包含所有可用 QLDB AWS 區域 的政策。如需可用區域的完整清單，請參閱中的 Amazon QLDB 端點和配額。AWS 一般參考

Amazon QLDB 故障診斷

以下各節提供使用 Amazon QLDB 時可能遇到的常見錯誤彙總清單，以及如何進行疑難排解的指導。

如需 IAM 存取特定的疑難排解指引，請參閱疑難排解 Amazon QLDB 身分識別和存取。

如需調整 PartiQL 陳述式的最佳作法，請參閱最佳化查詢效能。

主題

使用 QLDB 驅動程式執行交易

本節列出 Amazon QLDB 驅動程式在分類帳上執行 PartiQL 交易時，可能傳回的常見例外狀況。如需使用此功能的詳細資訊，請參閱「開始使用驅動程式」。如需設定和使用驅動程式的最佳作法，請參閱驅動程式建議。

每個例外都包含特定的錯誤訊息，後面接著簡短的描述和可能解決方案的建議。

CapacityExceededException

訊息：容量超過

Amazon QLDB 拒絕了該請求，因為它超出了分類帳的處理能力。QLDB 會針對每個分類帳強制執行內部調整規模限制，以維持服務的健康狀態和效能。此限制會根據每個個別要求的工作負載大小而有所不同。例如，如果要求執行效率低下的資料交易 (例如，非索引限定查詢所產生的資料表掃描)，工作負載可能會增加。

我們建議您等待，然後再重試請求。如果您的應用模組持續遇到此例外，請最佳化對帳單，並降低傳送至分類帳之請求的匯率與數量。陳述式最佳化的範例包括每個交易執行較少的陳述式以及調整資料表索引。若要瞭解如何最佳化陳述式並避免表格掃描，請參閱最佳化查詢效能。

我們也建議使用最新版本的 QLDB 驅動程式。驅動程式具有預設的重試原則，該原則會使用指數輪詢和抖動來自動重試例外狀況 (例如此類)。指數輪詢的概念是在連續錯誤響應的重試之間使用逐漸更長的等待時間。

InvalidSessionException

訊息：交易*## ID* 已過期

交易超過其最大生命週期。在認可之前，交易最多可以執行 30 秒。在此逾時限制之後，任何在交易上完成的工作都會遭到拒絕，而 QLDB 會捨棄工作階段。此限制可透過啟動交易而不提交或取消工作階段來保護用戶端不會洩漏工作階段。

如果這是應用程式中常見的例外狀況，則可能是交易執行時間太長。如果交易執行階段的時間超過 30 秒，請最佳化陳述式以加速交易。陳述式最佳化的範例包括每個交易執行較少的陳述式以及調整資料表索引。如需詳細資訊，請參閱 最佳化查詢效能。

InvalidSessionException

訊息：*#### ID* 已過期

QLDB 捨棄工作階段，因為它超過其最大總存留期。QLDB 會在 13—17 分鐘後捨棄工作階段，不論作用中的交易為何。工作階段可能會因為多種原因而遺失或受損，例如硬體故障、網路故障或應用程式重新啟動。因此，QLDB 會強制執行工作階段的最大生命週期，以確保用戶端軟體能夠抵禦工作階段失敗。

如果您遇到此例外狀況，建議您取得新的階段作業，然後重試該交易。我們也建議您使用最新版本的 QLDB 驅動程式，該驅動程式會代表應用程式管理工作階段集區及其健全狀況。

InvalidSessionException

訊息：沒有此類工作階段

用戶端嘗試使用不存在的工作階段與 QLDB 進行交易。假設用戶端使用先前存在的工作階段，工作階段可能因為下列其中一種情況而不再存在：

- 如果工作階段涉及內部伺服器故障 (也就是 HTTP 回應碼為 500 的錯誤)，QLDB 可能會選擇完全捨棄工作階段，而不是允許客戶在不確定狀態的工作階段進行交易。然後，該會話上的任何重試嘗試都會失敗並顯示此錯誤。
- 過期的工作階段最終會被 QLDB 遺忘。然後，任何嘗試繼續使用會話都會導致此錯誤，而不是初始錯誤`InvalidSessionException`。

如果您遇到此例外狀況，建議您取得新的階段作業，然後重試該交易。我們也建議您使用最新版本的 QLDB 驅動程式，該驅動程式會代表應用程式管理工作階段集區及其健全狀況。

RateExceededException

訊息：超過比率

QLDB 限制了基於呼叫者的身份的客戶端。QLDB 會使用權杖儲存貯體節流演算法，針對每個區域、每個帳戶強制執行節流。QLDB 這樣做是為了協助服務效能，並確保所有 QLDB 客戶的公平使用。例如，嘗試使用該作業取得大量並行工`StartSessionRequest`作階段可能會導致節流。

若要維持應用程式健康狀態並減輕進一步的節流，您可以使用指數輪詢和抖動在此例外狀況上重試。指數輪詢的概念是在連續錯誤響應的重試之間使用逐漸更長的等待時間。我們建議使用最新版本的 QLDB 驅動程式。驅動程式具有預設的重試原則，該原則會使用指數輪詢和抖動來自動重試例外狀況 (例如此類)。

如果您的應用程式持續受到 QLDB 限制`StartSessionRequest`呼叫，最新版本的 QLDB 驅動程式也會有所幫助。驅動程式會維護跨交易重複使用的工作階段集區，這有助於減少應用程式進行的`StartSessionRequest`呼叫次數。若要請求提高 API 調節限制，請與中AWS Support心聯絡。

LimitExceededException

訊息：超過工作階段限制

分類帳超過其作用中階段作業數目的配額 (也稱為限制)。此配額定義於中亞馬遜 QLDB 中的配額和限制。分類帳的有效階段作業計數最終是一致的，且持續在配額附近執行的分類帳可能會定期看到此例外。

為了維持應用程式的健康狀態，我們建議您重試此例外狀況。若要避免此例外狀況，請確定您尚未針對所有用戶端的單一分類帳設定超過 1,500 個並行階段作業。例如，您可以使用適用於 Java 的 Amazon QLDB 驅動程式maxConcurrentTransactions方法來設定驅動程式執行個體中可用工作階段的數目上限。

QldbClientException

訊息：只有在上階交易開啟時，串流的結果才有效

交易已關閉，無法用於從 QLDB 擷取結果。交易在提交或取消時關閉。

當用戶端直接使用`Transaction`物件，並嘗試提交或取消交易之後從 QLDB 擷取結果時，就會發生此例外狀況。為了緩解此問題，客戶端必須在關閉交易之前讀取數據。

匯出分錄資料

本節列出當您將日誌資料從分類帳匯出到 Amazon S3 儲存貯體時，QLDB 可能傳回的常見例外狀況。如需使用此功能的詳細資訊，請參閱「從 Amazon QLDB 匯出日誌資料」。

每個例外都包含特定的錯誤訊息，後面接著簡短的描述和可能解決方案的建議。

AccessDeniedException

訊息：使用*#############IAM:PassRole #####roleARN*

您沒有將 IAM 角色傳遞至 QLDB 服務的許可。QLDB 需要所有日誌匯出要求的角色，而且您必須擁有將此角色傳遞給 QLDB 的權限。此角色為 QLDB 提供指定 Amazon S3 儲存貯體中的寫入許可。

確認您定義的 IAM 政策授與對 QLDB 服務 (`qldb.amazonaws.com`) 的指定 IAM 角色資源執行`PassRole` API 作業的權限。如需政策範例，請參閱「Amazon QLDB 以身分識別為基礎的政策範例」。

IllegalArgumentException

訊息：QLDB 驗證 S3 組態時發生錯誤：*errorCode error Message*

造成此錯誤的可能原因是 Amazon S3 儲存貯體中不存在於 Amazon S3 儲存貯體。或者，QLDB 沒有足夠的許可將物件寫入指定的 Amazon S3 儲存貯體。

確認您在匯出任務請求中提供的 S3 儲存貯體名稱正確無誤。如需有關命名儲存貯體的詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的儲存貯體限制

此外，請確認您是否為指定儲存貯體定義政策，以`PutObjectAcl`授`PutObject`與 QLDB 服務 (`qldb.amazonaws.com`)。若要進一步了解，請參閱匯出權限。

IllegalArgumentException

訊息：驗證 S3 組態時，來自 Amazon S3 的意外回應。 S3 回應：*errorCode error Message*

嘗試將日誌匯出資料寫入提供的 S3 儲存貯體失敗，並顯示提供的 Amazon S3 錯誤回應。如需可能原因的詳細資訊，請參閱《Amazon S imple Storage Service 使用者指南》中的 Amazon S3 故障診斷。

IllegalArgumentException

訊息：Amazon S3 儲存貯體前綴不得超過 128 個字元

分錄匯出要求中提供的字首超過 128 個字元。

IllegalArgumentException

訊息：開始日期不得大於結束日期

`InclusiveStartTime`和`ExclusiveEndTime`必須採用 ISO 8601 日期和時間格式，並以國際標準時間 (UTC) 表示。

IllegalArgumentException

訊息：結束日期不能在 future

`InclusiveStartTime`和`ExclusiveEndTime`必須採用`ISO 8601`日期和時間格式顯示，並以 UTC 表示。

IllegalArgumentException

訊息：提供的物件加密設定 (S3EncryptionConfiguration) 與AWS Key Management Service (AWS KMS) 金鑰不相容

您提`KMSKeyArn`供了一個`ObjectEncryptionType`NO_ENCRYPTION或`SSE_S3`。您只能提供 AWS KMS key針對物件加密類型所管理的客戶`SSE_KMS`。若要進一步了解 Amazon S3 中的伺服器端加密選項，請參閱《Amazon S3 開發人員指南》中的使用伺服器端加密保護資料。

LimitExceededException

訊息：已超過 2 個同時執行的「分錄」匯出工作的限制

QLDB 會強制執行兩個並行分錄匯出工作的預設限制。

串流日誌資料

本節列出當您將日誌資料從分類帳串流到 Amazon Kinesis 資料串流時，QLDB 可能傳回的常見例外狀況。如需使用此功能的詳細資訊，請參閱「從 Amazon QLDB 串流日誌資料」。

每個例外都包含特定的錯誤訊息，後面接著簡短的描述和可能解決方案的建議。

AccessDeniedException

訊息：使用*##############IAM:PassRole #####roleARN*

您沒有將 IAM 角色傳遞至 QLDB 服務的許可。QLDB 需要所有日誌串流要求的角色，而且您必須擁有將此角色傳遞給 QLDB 的權限。此角色為 QLDB 提供您指定的 Amazon Kinesis Data Streams 源中的寫入許可。

確認您定義的 IAM 政策授與對 QLDB 服務 (`qldb.amazonaws.com`) 的指定 IAM 角色資源執行`PassRole` API 作業的權限。如需政策範例，請參閱「Amazon QLDB 以身分識別為基礎的政策範例」。

IllegalArgumentException

訊息：QLDB 在驗證 Kinesis Data Streams 時遇到錯誤：來自 Kinesis 的回應：*errorCode error Message*

此錯誤的可能原因是提供的 Kinesis Data Streams 資源不存在。或者，QLDB 沒有足夠的權限將資料記錄寫入指定的 Kinesis 資料串流。

確認您在串流請求中提供的 Kinesis 資料串流是否正確。如需詳細資訊，請參閱《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料。

此外，請確認您是否為指定的 Kinesis 資料串流定義原則，以授與下列動作的 QLDB 服務 (`qldb.amazonaws.com`) 權限。如需詳細資訊，請參閱 串流權限。

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

訊息：在驗證 Kinesis 組態時，Kinesis 資料串流產生非預期的回應。 來自 Kinesis 的回應：*errorCode errorMessage*

嘗試將資料記錄寫入提供的 Kinesis 資料串流失敗，並顯示提供的 Kinesis 錯誤回應。如需有關可能原因的詳細資訊，請參閱《Amazon Kinesis Data Streams 開發人員指南》中的 Amazon Kinesis Data Streams 故障診斷。

IllegalArgumentException

訊息：開始日期不得大於結束日期。

`InclusiveStartTime`和`ExclusiveEndTime`必須採用 ISO 8601 日期和時間格式，並以國際標準時間 (UTC) 表示。

IllegalArgumentException

訊息：開始日期不能在 future。

`InclusiveStartTime`和`ExclusiveEndTime`必須採用`ISO 8601`日期和時間格式顯示，並以 UTC 表示。

LimitExceededException

訊息：已超過 5 個同時執行 Kinesis 資料串流的日誌串流上限

QLDB 會強制執行五個並行日誌串流的預設限制。

驗證分錄資料

本節列出當您驗證分類帳中的分錄資料時，QLDB 可以傳回的常見例外。如需使用此功能的詳細資訊，請參閱「Amazon QLDB 中的數據驗證」。

每個異常都包含特定的錯誤消息，然後是可以拋出它的 API 操作，簡短描述以及可能的解決方案的建議。

IllegalArgumentException

訊息：提供的離子值無效且無法剖析。

API 操作：`GetDigest, GetBlock, GetRevision`

在重試請求之前，請確保您提供了有效的 Amazon Ion 值。

IllegalArgumentException

訊息：提供的區塊位址無效。

API 操作：`GetDigest, GetBlock, GetRevision`

在重試請求之前，請確保您提供了有效的阻止地址。區塊位址是具有兩個欄位的 Amazon Ion 結構：`strandId`和`sequenceNo`。

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

IllegalArgumentException

消息：提供的摘要提示地址的序列號超出了鏈的最新提交記錄。

API 操作：`GetDigest, GetBlock, GetRevision`

您提供的摘要提示位址的序號必須小於或等於日誌串的最新認可記錄的序號。在重試您的請求之前，請確定您提供的摘要提示位址具有有效的序號。

IllegalArgumentException

訊息：所提供區塊位址的串 ID 無效。

API 操作：`GetDigest, GetBlock, GetRevision`

您提供的區塊位址必須具有與分錄串 ID 相符的串 ID。在重試請求之前，請確保您提供了一個帶有效鏈 ID 的塊地址。

IllegalArgumentException

訊息：所提供區塊位址的序號超出線條的最新提交記錄。

API 操作：`GetBlock, GetRevision`

您提供的塊地址的序列號必須小於或等於鏈的最新提交記錄的序列號。在重試請求之前，請確保您提供了一個包含有效序號的塊地址。

IllegalArgumentException

消息：提供的塊地址的鏈 ID 必須與提供的摘要提示地址的 Strand ID 匹配。

API 操作：`GetBlock, GetRevision`

如果文件修訂或區塊存在於與您提供的摘要相同的日誌鏈中，您才能驗證該文件修訂或區塊。

IllegalArgumentException

訊息：所提供區塊位址的序號不得大於所提供摘要提示位址的序號。

API 操作：`GetBlock, GetRevision`

只有在您提供的摘要涵蓋的文件修訂或區塊時，才能驗證該文件修訂或區塊。這意味著它是在摘要提示地址之前提交給日記的。

IllegalArgumentException

訊息：在指定區塊位址的區塊中找不到提供的文件 ID。

API 操作：`GetRevision`

您提供的文件 ID 必須存在於您提供的區塊位址中。在重試您的請求之前，請確保這兩個參數是一致的。

Amazon QLDB 參考

亞馬遜 QLDB 支援 P artiQL 查詢語言的一個子集。下列主題描述 PartiQL 的 QLDB 實作。

> **Note**
>
> - QLDB 不支援所有 PartiQL 作業。
> - QLDB 中的所有 PartiQL 陳述式都受到交易限制的限制，如中所定義亞馬遜 QLDB 中的配額和限制。
> - 此參考提供您在 QLDB 主控台或 QLDB 命令介面上手動執行的 PartiQL 陳述式基本語法和使用範例。如需示範如何使用 QLDB 驅動程式以程式設計方式執行類似陳述式的程式碼範例，請參閱開始使用驅動程式.

主題

什麼是 PartiQL？

PartiQL 在包含結構化資料、半結構化資料和巢狀資料的多個資料存放區提供與 SQL 相容的查詢存取。此語言在 Amazon 內廣泛使用AWS 服務，現在作為許多人 (包括 QLDB) 的一部分提供。

如需 PartiQL 規範和核心查詢語言的教學課程，請參閱 PartiQL 文件。

PartiQL 擴展了 SQL-92 以支持亞馬遜離子數據格式的文檔。若要進一步了解 Amazon Ion 的資訊，請參閱亞馬遜 QLDB 中的亞馬遜離子數據格式參考。

Amazon QLDB 中的 PartiQL

若要在 QLDB 中執行 PartiQL 查詢，您可以使用下列其中一項：

- AWS Management Console適用於 QLDB 的 PartiQL 編輯器
- 命令行 QLDB 外殼
- AWS提供的 QLDB 驅動程式以程式設計方式執行查詢

如需使用這些方法存取 QLDB 的詳細資訊，請參閱訪問 Amazon QLDB。

若要瞭解如何控制在特定資料表上執行每個 PartiQL 命令的存取權，請參閱開始使用 Amazon QLDB 中的標準許可模式。

QLDB 中的快速提示

以下是在 QLDB 中使用 PartiQL 的秘訣和最佳作法的簡短摘要：

- 瞭解並行與交易限制 — 所有陳述式 (包括SELECT查詢) 都受到樂觀的並行控制 (OCC) 衝突和交易限制，包括 30 秒的交易逾時。
- 使用索引 — 使用高基數索引並執行目標查詢，以最佳化陳述式並避免完整表格掃描。如需進一步了解，請參閱 最佳化查詢效能。
- 使用相等述詞 — 索引查詢需要相等運算子 (=或IN)。不等式運算子 (<、>LIKE、BETWEEN) 不符合索引查詢的資格，因此會產生完整的資料表掃描。
- 僅使用內部聯結 — QLDB 僅支援內部聯結。最佳做法是聯結您要加入的每個資料表索引的欄位。為聯結準則和相等述詞選擇高基數索引。

亞馬遜 QLDB PartiQL 考慣例

本節說明用來為 Amazon QLDB PartiQL 參考資料中描述的 PartiQL 命令、函數和運算式撰寫語法的慣例。這些慣例不要與 PartiQL 查詢語言本身的語法和語意混淆。

字元	描述
CAPS (大寫字母)	大寫字詞為關鍵字。
[ ]	括號表示可選的參數或子句。方括號中的多個引數，代表您可以選擇任何數目的引數。此外，不同行中括號中的引數表示 QLDB 剖析器預期引數的順序與語法中列出的順序相同。
|	直立線符號會將您可選擇的引數隔開。
####	紅色斜體的字表示預留位置。插入適當的值來取代紅色斜體字詞。
. . .	省略符號表示您可以重複前一個元素。
'	單引號中的值表示您必須輸入單引號。在 PartiQL 中，單引號表示字符串值或亞馬遜離子結構中的字段名稱。
"	雙引號中的值表示您必須輸入雙引號。在 PartiQL 中，雙引號表示帶引號的標識符。
`	反引號中的值表示您必須輸入反引號。在 PartiQL 中，反引號表示離子常值。

Amazon QLDB 中的資料類型

亞馬遜 QLDB 以亞馬遜離子格式存儲文檔。Amazon Ion 是一種數據序列化格式（以文本形式和二進制編碼形式），是 JSON 的超集。下表列出您可以在 QLDB 文件中使用。

資料類型	描述
`null`	一個通用的空值
`bool`	布尔值
`int`	任意大小的有符號整數
`decimal`	任意精度的十進制編碼實數
`float`	二進位編碼浮點數 (64 位元 IEEE)

資料類型	描述
`timestamp`	日期/時間/時區刻任意精度
`string`	Unicode 文字
`symbol`	符号原子 (标识符)
`blob`	使用者定義編碼的二進位
`clob`	用戶定義編碼的文本數據
`struct`	名稱/值組的無序集合
`list`	值的有序異構集合

請參閱 Amazon GitHub 網站上的 Ion 規範文件，以取得 Ion 核心資料類型的完整清單，以及完整的說明和值格式詳細資訊。

Amazon QLDB 文件

亞馬遜 QLDB 將數據記錄存儲為文檔，這些記錄只是插入到表中的亞馬遜離子`struct`對象。有關離子規格，請參閱亞馬遜離子 GitHub網站。

主題

- Ion 文件結構
- 分部离子类型映射
- 文件識別碼

Ion 文件結構

像 JSON 一樣，QLDB 文檔由以下結構中的名稱-值對組成。

```
{
  name1: value1,
  name2: value2,
  name3: value3,
  ...
```

```
  nameN: valueN
}
```

名稱是符號標記，值不受限制。每個名稱-值對稱為一個字段。字段的值可以是任何 Ion資料類型，包括容器類型：嵌套結構，列表和結構列表。

也像 JSON 一樣，astruct 由大括號（{...}）表示，並且 alist 由方括號（[...]）表示。下列範例是來自範例資料的文件，其中包Amazon QLDB 主控台含各種類型的值。

```
{
    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**
>
> 在 Ion 中，雙引號表示字串值，未加引號的符號代表欄位名稱。但是在 PartiQL 中，單引號表示字符串和字段名稱。
> 這種語法差異可讓 PartiQL 查詢語言維持 SQL 相容性，而 Amazon Ion 資料格式則可維持 JSON 相容性。如需有關 QLDB 中 PartiQL 語法和語意的詳細資訊，請參閱使用 PartiQL 查詢離子。

分部离子类型映射

在 QLDB 中，PartiQL 擴展了 SQL 的類型系統，以涵蓋離子數據模型。此映射說明如下：

- SQL 標量類型由它們的離子同行覆蓋。例如：
  - CHAR並且VARCHAR是映射到離子string類型的 Unicode 序列。
  - NUMBER映射到離子decimal類型。

- Ion 的`struct`類型相當於 SQL 元組，它傳統上表示一個表行。
  - 不過，如果是開放式內容且沒有結構描述，則不支援依賴 SQL 元組排序本質的查詢 (例如的輸出順序`SELECT *`)。
- 除此之外`NULL`，PartiQL 還有一個`MISSING`類型。這是一個專業化，`NULL`並表示缺乏一個領域。此類型是必要的，因為 Ion`struct` 欄位可能很稀疏。

文件識別碼

QLDB 會將文件 ID 指派給您插入到表格中的每個文件。所有系統指派的 ID 都是通用唯一識別碼 (UUID)，每個識別碼都以 Base62 編碼的字串表示 (例如`3Qv67yjXEwB9SjmvkuG6Cp`)。如需詳細資訊，請參閱亞馬遜 QLDB 中的唯一 ID。

每個文件版本修訂都是由文件 ID 和從零開始的版本編號的組合來唯一識別。

文件 ID 和版本欄位包含在文件的中繼資料中，您可以在認可的檢視 (表格的系統定義檢視) 中進行查詢。如需 QLDB 中視圖的詳細資訊，請參閱核心概念。若要進一步了解中繼資料，請參閱查詢文件元資料。

在亞馬遜 QLDB 中使用 PartiQL 查詢離子

當您在亞馬遜 QLDB 中查詢資料時，您會以 PartiQL 格式撰寫陳述式，但 QLDB 會以亞馬遜離子格式傳回結果。PartiQL 的目的是與 SQL 兼容，而離子是 JSON 的擴展。這會導致您在查詢陳述式中註記資料的方式與查詢結果顯示方式之間的語法差異。

本節說明使用 QLDB 主控台或 QLDB 殼層手動執行 PartiQL 陳述式的基本語法和語意。

> **Tip**
>
> 當您以程式設計方式執行 PartiQL 查詢時，最佳做法是使用參數化陳述式。您可以在陳述式中使用問號 (?) 做為繫結變數預留位置，以避免這些語法規則。這也更加安全和高效。
> 若要進一步了解，請參閱開始使用驅動程式中的下列教學課程：
>
> - 爪哇：快速入門教學|食譜參考
> - 淨Quick 教學課程網站:|食譜參考
> - 前往：快速入門教學|食譜參考書參考
> - Node.js:快速入門教學 |食譜參考
> - 蟒蛇：速入門教學課程|食譜參考

主題

語法與語義

當使用 QLDB 主控台或 QLDB 外殼查詢離子資料時，以下是 PartiQL 的基本語法和語意：

區分大小寫

所有 QLDB 系統物件名稱 (包括欄位名稱、資料表名稱和分類帳名稱) 都區分大小寫。

字串值

在 Ion 中，雙引號 (`"..."`) 表示一個字符串。

在 PartiQL 中，單引號 (`'...'`) 表示一個字符串。

符號和標識符

在 Ion 中，單引號 (`'...'`) 表示符號。Ion 中的符號的子集稱為標識符由未加引號的文本表示。

在 PartiQL 中，雙引號 (`"..."`) 表示帶引號的 PartiQL 識別碼，例如用來做為資料表名稱的保留字。未加引號的文字代表一般的 PartiQL 識別碼，例如不是保留字的資料表名稱。

Ion 常值

任何離子文字都可以在 PartiQL 語句中用反引號 (`` `...` ``) 表示。

欄位名稱

Ion 欄位名稱區分大小寫的符號。PartiQL 可讓您在 DML 陳述式中使用單引號來表示欄位名稱。這是使用 PartiQL`cast` 函數定義符號的速記替代方法。它也比使用反引號來表示文字離子符號更直觀。

文字

PartiQL 查詢語言的常值對應於離子資料類型，如下所示：

标量

如果適用，請遵循 SQL 語法，如分部离子类型映射章節中所述。例如：

- 5
- 'foo'
- null

结构

也被稱為元組或許多格式和其他數據模型的對象。

由花括號（{...}）表示，用逗號分隔的struct元素。

- { 'id' : 3, 'arr': [1, 2] }

清單

也稱為陣列。

由方括號（[...]）表示，列表元素用逗號分隔。

- [1, 'foo']

包裝袋

在 PartiQL 中無序集合。

由雙角括號（<<...>>）表示，包元素用逗號分隔。QLDB 中的資料表可視為袋子。但是，包包不能嵌套在表格中的文件中。

- << 1, 'foo' >>

範例

下列是具有各種 Ion 類型的INSERT陳述式。

```
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`
}
```

反引号

PartiQL 完全涵蓋了所有 Ion 數據類型，因此您可以在不使用反引號的情況下編寫任何語句。但是在某些情況下，這種 Ion 字面語法可以使您的語句更清晰，更簡潔。

例如，要插入帶有 Ion 時間戳記和符號值的文檔，您可以僅使用純粹的 PartiQL 語法編寫以下語句。

```
INSERT INTO myTable VALUE
{
    'myTimestamp': to_timestamp('2019-09-04T'),
    'mySymbol': cast('foo' as symbol)
}
```

這是相當詳細的，所以相反，你可以使用反引號來簡化你的語句。

```
INSERT INTO myTable VALUE
{
    'myTimestamp': `2019-09-04T`,
    'mySymbol': `foo`
}
```

您也可以將整個結構包含在反引號中，以節省更多的按鍵動作。

```
INSERT INTO myTable VALUE
`{
    myTimestamp: 2019-09-04T,
    mySymbol: foo
}`
```

> **Important**
>
> 字符串和符號是在 PartiQL 不同的類。這意味著即使它們具有相同的文本，它們也不相等。例如，下列 PartiQL 運算式會評估為不同的離子值。
>
> ```
> 'foo'
> ```
>
> ```
> `foo`
> ```

路徑導覽

撰寫資料操作語言 (DML) 或查詢陳述式時，您可以使用路徑步驟存取巢狀結構中的欄位。PartiQL 支援點符號來存取父結構的欄位名稱。下列範例會存取父項的`Model`欄位`Vehicle`。

```
Vehicle.Model
```

要訪問列表的特定元素，可以使用方括號運算符來表示從零開始的序數。下列範例會`SecondaryOwners`使用的序號存取的元素`2`。換句話說，這是列表的第三個元素。

```
SecondaryOwners[2]
```

鋸齒

QLDB 支持開放的內容和模式。因此，當您存取陳述式中的特定欄位時，確保獲得預期結果的最佳方式就是使用別名。例如，如果您沒有指定明確的別名，則系統會為您的`FROM`來源產生隱含的別名。

```
SELECT VIN FROM Vehicle
--is rewritten to
SELECT Vehicle.VIN FROM Vehicle AS Vehicle
```

但是，對於字段名稱衝突，結果是不可預測的。如果文件內的巢狀結構中`VIN`存在另一個名為的欄位，則此查詢傳回的`VIN`值可能會讓您感到驚訝。最佳做法是改寫下列陳述式。此查詢宣告`v`為範圍在`Vehicle`資料表上的別名。`AS`關鍵字是可選的。

```
SELECT v.VIN FROM Vehicle [ AS ] v
```

在文件內的巢狀集合中進行路徑分析時，別名特別有用。例如，下列陳述式會宣告o為範圍超過集合的別名VehicleRegistration.Owners。

```
SELECT o.SecondaryOwners
FROM VehicleRegistration AS r, @r.Owners AS o
```

在這裡，@字符在技術上是可選的。但它明確表明你想要結Owners構VehicleRegistration，而不是名為的不同集合Owners（如果存在的話）。

PartiQL 規格

如需 PartiQL 查詢語言的詳細資訊，請參閱 PartiQL 規格。

亞馬遜 QLDB 中的指令

PartiQL 擴展了 SQL-92 以支持亞馬遜離子數據格式的文檔。Amazon QLDB 支援下列 PartiQL 命令。

若要瞭解如何控制在特定資料表上執行每個 PartiQL 命令的存取權，請參閱開始使用 Amazon QLDB 中的標準許可模式。

> **Note**
>
> - QLDB 不支援所有 PartiQL 指令。
> - QLDB 中的所有 PartiQL 陳述式都受到交易限制的限制，如中所定義亞馬遜 QLDB 中的配額和限制。
> - 此參考提供您在 QLDB 主控台或 QLDB 命令介面上手動執行的 PartiQL 陳述式基本語法和使用範例。如需示範如何使用支援的程式設計語言執行類似陳述式的程式碼範例，請參閱開始使用驅動程式.

DDL 陳述式 (資料定義語言)

資料定義語言 (DDL) 是一組用來管理資料表和索引。您可以使用 DDL 來建立和卸除這些物件。

- CREATE INDEX
- CREATE TABLE
- DROP INDEX

- DROP TABLE
- 取消刪除資料表

DML 陳述式 (資料操作語言)

資料操作語言 (DML) 是一組用來管理 QLDB 資料表中資料的 PartiQL 陳述式。您可使用 DML 陳述式新增、修改或刪除資料表中的資料。

支援下列 DML 和查詢語言陳述式：

- DELETE
- 從 (插入、移除或設定)
- INSERT
- SELECT
- UPDATE

在亞馬遜 QLDB 中創建索引命令

在 Amazon QLDB 中，使用命`CREATE INDEX`令為表格上的文件欄位建立索引。

若要瞭解如何控制在特定資料表上執行此 PartiQL 命令的存取權，請參閱開始使用 Amazon QLDB 中的標準許可模式。

> **Important**
>
> QLDB 需要一個索引來有效地查找文檔。如果沒有索引，QLDB 需要在讀取文檔時進行全表掃描。這可能會造成大型資料表的效能問題，包括並行衝突和交易逾時。
> 若要避免資料表掃描，您必須在索引欄位或文件 ID 上使用相等運算子 (=或`IN`) 執行具有述`WHERE`詞子句的陳述式。如需詳細資訊，請參閱最佳化查詢效能。

建立索引時，請注意下列條件約束：

- 索引只能在單一頂層欄位上建立。不支援複合、巢狀、唯一和以函數為基礎的索引。
- 您可以在任何 Ion 資料類型上建立索引，包括`list`和`struct`。但是，無論 Ion 類型如何，您都只能通過整個 Ion 值的相等性來進行索引查找。例如，當使用`list`類型作為索引時，您不能通過列表中的一個項目進行索引查找。

- 只有在您使用相等述詞時才會改善查詢效能；例如，`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`。

  QLDB 不尊重查詢謂詞中的不等式。因此，範圍篩選的掃描不會實作。
- 索引欄位的名稱區分大小寫，且最大長度可為 128 個字元。
- QLDB 中的索引建立是非同步的。完成建立非空資料表所需的時間量。如需詳細資訊，請參閱 管理索引。

主題

- 語法
- 參數
- 傳回值
- 範例
- 以編程方式使用驅動程序

語法

```
CREATE INDEX ON table_name (field)
```

參數

table_name

要在其中建立索引的資料表的名稱。索資料表必須已存在。

資料表名稱區分大小寫。

##

要為其建立索引的文件欄位名稱。欄位必須是頂層屬性。

索引欄位的名稱區分大小寫，且最大長度可為 128 個字元。

您可以在任何 Amazon Ion 資料類型上建立索引，包括`list`和`struct`。但是，無論 Ion 類型如何，您都只能通過整個 Ion 值的相等性來進行索引查找。例如，當使用`list`類型作為索引時，您不能通過列表中的一個項目進行索引查找。

傳回值

`tableId`— 您建立索引之資料表的唯一 ID。

範例

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

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

以編程方式使用驅動程序

若要了解如何以程式設計方式使用 QLDB 驅動程式執行此陳述式，請參閱開始使用驅動程式中的下列教學課程：

- 爪哇：快速入門教學|食譜參考
- 淨Quick 教學課程網站:|食譜參考
- 前往：快速入門教學|食譜參考書參考
- Node.js:快速入門教學 |食譜參考
- 蟒蛇：速入門教學課程|食譜參考

在亞馬遜 QLDB 中創建表命令

在 Amazon QLDB 中，使用命`CREATE TABLE`令建立新資料表。

表具有沒有命名空間的簡單名稱。QLDB 支援開放式內容，不會強制執行結構定義，因此您不會在建立資料表時定義屬性或資料類型。

> **Note**
>
> 若要瞭解如何控制在分類帳中執行此 PartiQL 命令的存取權，請參閱開始使用 Amazon QLDB 中的標準許可模式。

主題

- 語法
- 參數

語法

```
CREATE TABLE table_name [ WITH (aws_tags = `{'key': 'value'}`) ]
```

參數

table_name

要建立的資料表的唯一名稱。具有相同名稱的活動表必須不存在。以下是命名條件約束：

- 只能包含 1—128 個英數字元或底線。
- 第一個字元必須為字母或底線。
- 其餘字元可以包含英數字元和底線的任意組合。
- 區分大小寫。
- 不得為 QLDB PartiQL 留字。

'*#*' : '*##*'

(選用) 建立期間要連接到資源。每個標籤都被定義為一個鍵值對，其中的鍵和值分別用單引號表示。每個鍵值對都在 Amazon Ion 結構內定義，該結構由反引號表示。

目前僅在*STANDARD*權限模式下，分類帳支援在建立時標記表格。

傳回值

`tableId`— 您建立之資料表的唯一 ID。

建立時為資料表標籤

> **Note**
>
> 目前僅在STANDARD權限模式下，分類帳支援在建立時標記表格。

或者，您可以在CREATE TABLE陳述式中指定標籤來標記資料表資源。如需標籤的詳細資訊，請參閱標記 Amazon QLDB 資源。下列範例會建立以標籤命名Vehicle的資料表environment=production。

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

在建立表格時標記表格需要存取qldb:PartiQLCreateTable和qldb:TagResource動作。若要進一步了解 QLDB 資源的詳細資訊，請參閱Amazon QLDB 如何與 IAM 合作。

藉由在建立時為資源建立標籤，您可以消除在資源建立後執行自訂標籤指令碼的必要。標記資料表之後，您可以根據這些標籤來控制資料表的存取。例如，您只能將完整存取權授與具有特定標籤的資料表。如需 JSON 政策範例，請參閱根據表格標籤對所有動作的完整存取權。

範例

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'development'}`)
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'key1': 'value1', 'key2': 'value2'}`)
```

以編程方式使用驅動程序

若要了解如何以程式設計方式使用 QLDB 驅動程式執行此陳述式，請參閱開始使用驅動程式中的下列教學課程：

- 爪哇：快速入門教學|食譜參考
- 淨Quick 教學課程網站:|食譜參考
- 前往：快速入門教學|食譜參考書參考
- Node.js:快速入門教學 |食譜參考
- 蟒蛇：速入門教學課程|食譜參考

亞馬遜 QLDB 中的刪除命令

在 Amazon QLDB 中，透過建立文件的新最終修訂版，使用此DELETE命令將作用中文件標記為已刪除。此最終修訂版表示文件已刪除。此作業會結束文件的生命週期，這意味著無法再建立具有相同文件 ID 的文件修訂版本。

此項操作無法復原。您仍然可以使用查詢已刪除文件的修訂版本記錄歷史功能。

> **Note**
>
> 若要瞭解如何控制在特定資料表上執行此 PartiQL 命令的存取權，請參閱開始使用 Amazon QLDB 中的標準許可模式。

主題

語法

```
DELETE FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
```

參數

table_name

包含要刪除之資料的使用者資料表名稱。只有預設使用者檢視才支援 DML 陳述式。每個陳述式只能在單一資料表上執行。

如*####*

(選擇性) 使用者定義的別名，其範圍涵蓋要從中刪除的資料表。AS關鍵字是可選的。

按 ***ID ##***

(選擇性) 使用者定義別名，繫結至結果集中每個文件的中id繼資料欄位。別名必須使用BY關鍵字在FROM子句中聲明。當您要在查詢預設使用者檢視時篩選文件 ID 時，此功能非常有用。如需詳細資訊，請參閱使用 BY 子句來查詢文件 ID。

WHERE ***condition***

要刪除的文件的選取條件。

> **Note**
>
> 如果您省略子WHERE句，則會刪除資料表中的所有文件。

傳回值

documentId— 您刪除的每個文件的唯一 ID。

範例

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

以編程方式使用驅動程序

若要了解如何以程式設計方式使用 QLDB 驅動程式執行此陳述式，請參閱開始使用驅動程式中的下列教學課程：

- 爪哇：快速入門教學|食譜參考
- 淨Quick 教學課程網站:|食譜參考
- 前往：快速入門教學|食譜參考書參考
- Node.js:快速入門教學 |食譜參考
- 蟒蛇：速入門教學課程|食譜參考

在亞馬遜 QLDB 刪除索引命令

在 Amazon QLDB 中，使用命DROP INDEX令刪除資料表上的索引。

> **Note**
>
> 若要瞭解如何控制在特定資料表上執行此 PartiQL 命令的存取權，請參閱開始使用 Amazon QLDB 中的標準許可模式。

主題

- 語法

- 參數
- 傳回值
- 範例

語法

```
DROP INDEX "indexId" ON table_name WITH (purge = true)
```

> **Note**
>
> 所有DROP INDEX陳述式WITH (purge = true)都需要此子句，而且true是目前唯一支援的值。
> 關鍵字區purge分大小寫，且必須完全使用小寫。

參數

「*## ID*」

要刪除之索引的唯一識別碼，以雙引號表示。

在*####*上

要刪除其索引的資料表的名稱。

傳回值

`tableId`— 您卸除其索引之表格的唯一 ID。

範例

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

在亞馬遜 QLDB 刪除表命令

在 Amazon QLDB 中，使用命DROP TABLE令停用現有資料表。您可以使用取消刪除資料表陳述式來重新啟用它。停用或重新啟動表格不會影響其文件或索引。

> **Note**
>
> 若要瞭解如何控制在特定資料表上執行此 PartiQL 命令的存取權，請參閱開始使用 Amazon QLDB 中的標準許可模式。

主題

語法

```
DROP TABLE table_name
```

參數

table_name

要停用的資料表名稱。資料表必須已存在，且其狀態必須為ACTIVE。

傳回值

`tableId`— 您停用的資料表的唯一 ID。

範例

```
DROP TABLE VehicleRegistration
```

從亞馬遜 QLDB 中的（插入，刪除或設置）命令

在 Amazon QLDB 中，以開頭的陳述式FROM是 PartiQL 延伸模組，可讓您插入和移除文件中的特定元素。您也可以使用這個陳述式來更新文件中的現有元素，類似於UPDATE指令。

> **Note**
>
> 若要瞭解如何控制在特定資料表上執行此 PartiQL 命令的存取權，請參閱開始使用 Amazon QLDB 中的標準許可模式。

主題

語法

來源-插入

在現有文件中插入新元素。若要將新的頂層文件插入表格中，您必須使用INSERT。

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
INSERT INTO element VALUE data [ AT key_name ]
```

從移除

移除文件中的現有元素，或移除整個頂層文件。後者在語義上與傳統DELETE語法相同。

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
REMOVE element
```

從集合

更新文件中的一或多個元素。如果一個元素不存在，它被插入。這在語義上與傳統UPDATE語法相同。

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
SET element = data [, element = data, ... ]
```

參數

table_name

包含要修改之資料的使用者資料表名稱。只有預設使用者檢視才支援 DML 陳述式。每個陳述式只能在單一資料表上執行。

在這個子句中，您也可以包含嵌套在指定資料表中的一或多個集合。如需詳細資訊，請參閱 巢狀集合。

如*####*

(選擇性) 使用者定義別名，其範圍覆蓋要修改的資料表。在SET、REMOVE、或WHERE子句中使用的所有資料表別名都必須在FROM子句中宣告。INSERT INTOAS關鍵字是可選的。

按 *ID ##*

(選擇性) 使用者定義別名，繫結至結果集中每個文件的中id繼資料欄位。別名必須使用BY關鍵字在FROM子句中聲明。當您要在查詢預設使用者檢視時篩選文件 ID 時，此功能非常有用。如需詳細資訊，請參閱使用 BY 子句來查詢文件 ID。

WHERE *condition*

要修改之文件的選取條件。

> **Note**
>
> 如果您省略子WHERE句，則會修改資料表中的所有文件。

##

要建立或修改的文件元素。

data

元素的新值。

在####

要在要修改的文件中新增的索引鍵名稱。您必須指定VALUE相應的金鑰名稱。這對於在文檔中的特定位置插入新值AT是必需的。

巢狀集合

雖然您只能在單一資料表上執行 DML 陳述式，但您可以將該表格中文件內的巢狀集合指定為其他來源。您為巢狀集合宣告的每個別名都可以在WHERE子句和SETINSERT INTO、或REMOVE子句中使用。

例如，下列陳述式的FROM來源包含資料VehicleRegistration表和巢狀Owners.SecondaryOwners結構。

```
FROM VehicleRegistration r, @r.Owners.SecondaryOwners o
WHERE r.VIN = '1N4AL11D75C109151' AND o.PersonId = 'abc123'
SET o.PersonId = 'def456'
```

此範例會更新SecondaryOwners清單中具有的特定元素，該元素'abc123'在VehicleRegistration文件中具有VIN的'1N4AL11D75C109151'。PersonId這個表達式可以讓你通過它的值，而不是它的索引來指定列表的元素。

傳回值

documentId— 您更新或刪除的每個文件的唯一 ID。

範例

修改文件中的元素。若該元素不存在，則會加以插入。

```
FROM Vehicle AS v
WHERE v.VIN = '1N4AL11D75C109151' AND v.Color = 'Silver'
SET v.Color = 'Shiny Gray'
```

在系統指派的文件id中繼資料欄位上修改或插入元素和篩選器。

```
FROM Vehicle AS v BY v_id
WHERE v_id = 'documentId'
SET v.Color = 'Shiny Gray'
```

修改文件中Owners.SecondaryOwners清單中第一個元素的PersonId欄位。

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
```

移除文件中的現有元素。

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

從表格中移除整個文件。

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

移除表VehicleRegistration格中文件內Owners.SecondaryOwners清單的第一個元素。

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
REMOVE r.Owners.SecondaryOwners[0]
```

在Vehicle表格中的文件中{'Mileage':26500}作為頂層名稱-值配對插入。

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

在VehicleRegistration表中的文檔的Owners.SecondaryOwners字段中{'PersonId':'abc123'}作為名稱-值對追加。請注意，Owners.SecondaryOwners必須已經存在，且必須是清單資料類型，此陳述式才能有效。否則，子INSERT INTO句中需要關鍵字AT。

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
```

插入{'PersonId':'abc123'}為文件內現有Owners.SecondaryOwners清單中的第一個元素。

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
```

將多個名稱-值對附加到文檔中的現有Owners.SecondaryOwners列表。

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
```

以編程方式使用驅動程序

若要了解如何以程式設計方式使用 QLDB 驅動程式執行此陳述式，請參閱開始使用驅動程式中的下列教學課程：

- 爪哇：快速入門教學|食譜參考
- 淨Quick 教學課程網站:|食譜參考
- 前往：快速入門教學|食譜參考書參考
- Node.js:快速入門教學 |食譜參考
- 蟒蛇：速入門教學課程|食譜參考

在亞馬遜 QLDB 插入命令

在亞馬遜 QLDB 中，使用INSERT命令將一個或多個 Amazon 離子文檔添加到表中。

> **Note**
>
> 若要瞭解如何控制在特定資料表上執行此 PartiQL 命令的存取權，請參閱開始使用 Amazon QLDB 中的標準許可模式。

主題

- 語法
- 參數

- 傳回值
- 範例
- 以編程方式使用驅動程序

語法

插入單一文件。

```
INSERT INTO table_name VALUE document
```

插入多個文件。

```
INSERT INTO table_name << document, document, ... >>
```

參數

table_name

您希望在其中插入資料的使用者資料表名稱。索資料表必須已存在。只有預設使用者檢視才支援 DML 陳述式。

##

有效的 QLDB 文件。您必須指定至少 1 個文件。多個文件必須以英文逗號分隔。

該文檔必須用大括號 (`{...}`) 來表示。

文件中的每個欄位名稱都是區分大小寫的 Ion 符號，可在 PartiQL 中以單引號 (`'...'`) 表示。

PartiQL 中的字串值也使用單引號 (`'...'`) 表示。

任何離子文字都可以用反引號 (`` `...` ``) 表示。

> **Note**
>
> 雙角括號 (`<<...>>`) 表示無序集合 (在 PartiQL 中稱為袋) ，並且只有在您要插入多個文檔時才需要使用。

傳回值

documentId— 您插入的每個文件的唯一 ID。

範例

插入單一文件。

```
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`
}
```

這個陳述式會傳回您插入之文件的唯一 ID，如下所示。

```
{
    documentId: "2kKuOPNB07D2iTPBrUTWGl"
}
```

插入多個文件。

```
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'
}
>>
```

這個陳述式會傳回您插入的每個文件的唯一 ID，如下所示。

```
{
    documentId: "6WXzLscsJ3bDWW97Dy8nyp"
},
{
    documentId: "35e0ToZyTGJ7LGvcwrkX65"
},
{
    documentId: "BVHPcH612o7JROQ4yP8jiH"
}
```

以編程方式使用驅動程序

若要了解如何以程式設計方式使用 QLDB 驅動程式執行此陳述式，請參閱開始使用驅動程式中的下列教學課程：

- 爪哇：快速入門教學|食譜參考
- 淨Quick 教學課程網站:|食譜參考
- 前往：快速入門教學|食譜參考書參考
- Node.js:快速入門教學 |食譜參考

- 蟒蛇：速入門教學課程|食譜參考

亞馬遜 QLDB 中的選擇命令

在 Amazon QLDB 中，使用命SELECT令從一個或多個表格擷取資料。QLDB 中的每個SELECT查詢都會在交易中處理，並受到交易逾時限制的限制。

結果的順序不是特定的，並且每個SELECT查詢可能會有所不同。您不應該依賴 QLDB 中任何查詢的結果順序。

若要瞭解如何控制在特定資料表上執行此 PartiQL 命令的存取權，請參閱開始使用 Amazon QLDB 中的標準許可模式。

> **Warning**
>
> 當您在沒有索引查閱的情況下在 QLDB 中執行查詢時，它會叫用完整資料表掃描。PartiQL 支持這樣的查詢，因為它是 SQL 兼容。不過，請勿在 QLDB 中針對生產使用案例執行資料表掃描。資料表掃描可能會造成大型資料表的效能問題，包括並行衝突和交易逾時。
> 若要避免資料表掃描，您必須在索引欄位或文件 ID 上使用相等運算子來執行具有述WHERE詞子句的陳述式；例如，`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`。如需詳細資訊，請參閱 最佳化查詢效能。

主題

- 語法
- 參數
- 聯結
- 巢狀查詢限制
- 範例
- 以編程方式使用驅動程序

語法

```
SELECT [ VALUE ] expression [ AS field_alias ] [, expression, ... ]
FROM source [ AS source_alias ] [ AT idx_alias ] [ BY id_alias ] [, source, ... ]
```

```
[ WHERE condition ]
```

參數

VALUE

運算式的限定詞，可讓查詢傳回原始資料類型值，而不是封裝在元組結構中的值。

###

從結果集中*一或多個文件欄位的投影清單。一個表達式可以包含對 PartiQL 函數 的呼叫或由 PartiQL 運算子 修改的欄位。

AS ***####***

(選擇性) 在最終結果集中使用之欄位的暫時性、使用者定義別名。AS關鍵字是可選的。

如果您沒有為非簡單欄位名稱的運算式指定別名，則結果集會將預設名稱套用至該欄位。

來***#***

要查詢的來源。目前支援的唯一來源是資料表名稱、資料表之間的內部聯結、巢狀SELECT查詢 (受限於巢狀查詢限制)，以及資料表的歷程函數呼叫。

您必須指定至少 1 個來源。多個來源必須以英文逗號分隔。

AS ***####***

(選擇性) 使用者定義的別名，其範圍超過要查詢的來源。SELECTORWHERE 子句中使用的所有來源別名都必須在FROM子句中宣告。AS關鍵字是可選的。

在 ***ID ##***

(選擇性) 使用者定義別名，繫結至來源清單中每個元素的索引 (序數) 編號。別名必須使用AT關鍵字在FROM子句中聲明。

按 ***ID ##***

(選擇性) 使用者定義別名，繫結至結果集中每個文件的中id繼資料欄位。別名必須使用BY關鍵字在FROM子句中聲明。當您想要在查詢預設使用者檢視時對文件 ID 進行投影或篩選時，此功能非常有用。如需詳細資訊，請參閱使用 BY 子句來查詢文件 ID。

WHERE ***condition***

查詢的選取條件和聯結準則 (如果適用)。

> **Note**
>
> 如果您省略子WHERE句，則會檢索資料表中的所有文件。

聯結

目前僅支援內部聯結。您可以使用 explicitINNER JOIN 子句編寫內部聯接查詢，如下所示。在此語法中，JOIN必須與配對ON，並且INNER關鍵字是可選的。

```
SELECT expression
FROM table1 AS t1 [ INNER ] JOIN table2 AS t2
ON t1.element = t2.element
```

或者，您可以使用隱含語法編寫內部聯結，如下所示。

```
SELECT expression
FROM table1 AS t1, table2 AS t2
WHERE t1.element = t2.element
```

巢狀查詢限制

您可以在SELECT運算式內和FROM來源中撰寫巢狀查詢 (子查詢)。主要限制是只有最外層的查詢才能訪問全局數據庫環境。例如，假設您有一個包含資料表VehicleRegistration和Person。下列巢狀查詢無效，因為內部SELECT會嘗試存取Person。

```
SELECT r.VIN,
    (SELECT p.PersonId FROM Person AS p WHERE p.PersonId =
 r.Owners.PrimaryOwner.PersonId) AS PrimaryOwner
FROM VehicleRegistration AS r
```

而下面的嵌套查詢是有效的。

```
SELECT r.VIN, (SELECT o.PrimaryOwner.PersonId FROM @r.Owners AS o) AS PrimaryOwner
FROM VehicleRegistration AS r
```

範例

下列查詢會顯示具SELECT有使用IN運算子之標準WHERE述詞子句的全部萬用字元的基本全部萬用字元。

```
SELECT * FROM Vehicle
WHERE VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

以下顯SELECT示具有字串篩選的投影。

```
SELECT FirstName, LastName, Address
FROM Person
WHERE Address LIKE '%Seattle%'
AND GovId = 'LEWISR261LL'
```

下面顯示了一個扁平化嵌套數據的相關子查詢。請注意，字@符在技術上是可選的。但它明確表明您想要嵌套在其中的Owners結構VehicleRegistration，而不是名為的不同集合Owners（如果存在的話）。如需更多前後關聯，請參閱〈使用資料和記錄〉一章巢狀資料中的。

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

以下顯示了SELECT列表中投影嵌套數據的子查詢以及隱含的內部關連。

```
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')
```

下面顯示了一個明確的內部聯接。

```
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')
```

下面顯示了使用BY子句的文檔id元數據字段的投影。

```
SELECT
    r_id,
    r.VIN
FROM
    VehicleRegistration AS r BY r_id
WHERE
    r_id = 'documentId'
```

以下使用子BY句分別聯結其欄位DriversLicense和文件id欄位上的PersonId和Person表格。

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
ON d.PersonId = pid
WHERE pid = 'documentId'
```

下列使用分別聯結其欄位DriversLicense和文件id欄位上的PersonId和Person表格。已提交檢視

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS cp
ON d.PersonId = cp.metadata.id
WHERE cp.metadata.id = 'documentId'
```

以下內容會針對表格中的文件傳回Owners.SecondaryOwners清單中每個人員的PersonId與索引 (序數) 編號VehicleRegistration。

```
SELECT s.PersonId, owner_idx
FROM VehicleRegistration AS r, @r.Owners.SecondaryOwners AS s AT owner_idx
WHERE r.VIN = 'KM8SRDHF6EU074761'
```

以編程方式使用驅動程序

若要了解如何以程式設計方式使用 QLDB 驅動程式執行此陳述式，請參閱開始使用驅動程式中的下列教學課程：

- 爪哇：快速入門教學|食譜參考

- 淨Quick 教學課程網站:|食譜參考
- 前往：快速入門教學|食譜參考書參考
- Node.js:快速入門教學 |食譜參考
- 蟒蛇：速入門教學課程|食譜參考

亞馬遜 QLDB 中的更新命令

在 Amazon QLDB 中，使用UPDATE命令修改文件內一或多個元素的值。如果一個元素不存在，它被插入。

您也可以使用這個命令，在文件中明確插入和移除特定元素，類似從 (插入、移除或設定)陳述式。

> **Note**
>
> 若要瞭解如何控制在特定資料表上執行此 PartiQL 命令的存取權，請參閱開始使用 Amazon QLDB 中的標準許可模式。

主題

- 語法
- 參數
- 傳回值
- 範例
- 以編程方式使用驅動程序

語法

更新集

更新文件中的一或多個元素。如果一個元素不存在，它被插入。這在語義上與 FROM-SET 語句相同。

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
SET element = data [, element = data, ... ]
[ WHERE condition ]
```

更新插入

在現有文件中插入新元素。若要將新的頂層文件插入表格中，您必須使用INSERT。

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
INSERT INTO element VALUE data [ AT key_name ]
[ WHERE condition ]
```

更新刪除

移除文件中的現有元素，或移除整個頂層文件。後者在語義上與傳統DELETE語法相同。

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
REMOVE element
[ WHERE condition ]
```

參數

table_name

要修改之資料表的名稱。只有預設使用者檢視才支援 DML 陳述式。每個陳述式只能在單一資料表上執行。

如####

(選擇性) 使用者定義別名，其範圍覆蓋要更新的資料表。AS關鍵字是可選的。

按 *ID ##*

(選擇性) 使用者定義別名，繫結至結果集中每個文件的中id繼資料欄位。別名必須使用BY關鍵字在UPDATE子句中聲明。當您要在查詢預設使用者檢視時篩選文件 ID 時，此功能非常有用。如需詳細資訊，請參閱使用 BY 子句來查詢文件 ID。

##

要建立或修改的文件元素。

data

元素的新值。

在####

要在要修改的文件中新增的索引鍵名稱。您必須指定VALUE相應的金鑰名稱。這對於在文檔中的特定位置插入新值AT是必需的。

WHERE *condition*

要修改之文件的選取條件。

> **Note**
>
> 如果您省略子WHERE句，則會修改資料表中的所有文件。

傳回值

documentId— 您更新的每個文件的唯一 ID。

範例

更新文件中的欄位。若該欄位會加以插入。

```
UPDATE Person AS p
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE p.GovId = '111-22-3333'
```

篩選系統指派的文件id中繼資料欄位。

```
UPDATE Person AS p BY pid
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE pid = 'documentId'
```

覆寫整個文件。

```
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'
```

修改文件中Owners.SecondaryOwners清單中第一個元素的PersonId欄位。

```
UPDATE VehicleRegistration AS r
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
WHERE r.VIN = '1N4AL11D75C109151'
```

在Vehicle表格中的文件中{'Mileage':26500}作為頂層名稱-值配對插入。

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

在VehicleRegistration表中的文檔的Owners.SecondaryOwners字段中{'PersonId':'abc123'}作為名稱-值對追加。請注意，Owners.SecondaryOwners必須已經存在，且必須是清單資料類型，此陳述式才能有效。否則，子INSERT INTO句中需要關鍵字AT。

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
WHERE r.VIN = '1N4AL11D75C109151'
```

插入{'PersonId':'abc123'}為文件內現有Owners.SecondaryOwners清單中的第一個元素。

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
WHERE r.VIN = '1N4AL11D75C109151'
```

將多個名稱-值對附加到文檔中的現有Owners.SecondaryOwners列表。

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
WHERE r.VIN = '1N4AL11D75C109151'
```

移除文件中的現有元素。

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

從表格中移除整個文件。

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

移除表VehicleRegistration格中文件內Owners.SecondaryOwners清單的第一個元素。

```
UPDATE VehicleRegistration AS r
REMOVE r.Owners.SecondaryOwners[0]
WHERE r.VIN = '1N4AL11D75C109151'
```

以編程方式使用驅動程序

若要了解如何以程式設計方式使用 QLDB 驅動程式執行此陳述式，請參閱開始使用驅動程式中的下列教學課程：

- 爪哇：快速入門教學|食譜參考
- 淨Quick 教學課程網站:|食譜參考
- 前往：快速入門教學|食譜參考書參考
- Node.js:快速入門教學 |食譜參考
- 蟒蛇：速入門教學課程|食譜參考

在亞馬遜 QLDB 中取消刪除表命令

在 Amazon QLDB 中，使用命UNDROP TABLE令重新啟用先前刪除的資料表 (也就是停用)。停用或重新啟動表格不會影響其文件或索引。

> **Note**
>
> 若要瞭解如何控制在特定資料表上執行此 PartiQL 命令的存取權，請參閱開始使用 Amazon QLDB 中的標準許可模式。

主題

- 語法
- 參數
- 傳回值

- 範例

語法

```
UNDROP TABLE "tableId"
```

參數

「*## ID*」

要重新啟動之表格的唯一識別碼，以雙引號表示。

表格必須先前已被刪除，表示它存在於系統目錄表格中，information_schema.user_tables且狀態為INACTIVE。也必須沒有具有相同名稱的活動現有表。

傳回值

tableId— 您重新啟動之資料表的唯一 ID。

範例

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

亞馬遜 QLDB 中的 PartiQL 函數

Amazon QLDB 中的 PartiQL 支援下列 SQL 標準函數的內建變體。

> **Note**
>
> QLDB 目前不支援任何未包含在此清單中的 SQL 函數。
> 這個函數引用是基於 PartiQL 文檔的內置函數。

未知的類型 (空值和遺失) 傳播

除非另有說明，否則所有這些函數都會傳播 null 和缺少的參數值。傳播NULL或MISSING被定義為返回，NULL如果任何函數參數是NULL或MISSING。以下是此傳播的範例。

```
CHAR_LENGTH(null)    -- null
CHAR_LENGTH(missing) -- null (also returns null)
```

彙總函數

- AVG
- COUNT
- MAX
- MIN
- SIZE
- SUM

條件函數

- COALESCE
- EXISTS
- NULLIF

日期和時間函數

- DATE_ADD
- DATE_DIFF
- EXTRACT
- UTCNOW

標量函數

- TXID

字串函數

- CHAR_LENGTH
- CHARACTER_LENGTH

- LOWER
- SUBSTRING
- TRIM
- UPPER

資料類型格式化函數

- CAST
- TO_STRING
- TO_TIMESTAMP

亞馬遜 QLDB 中的 AVG 功能

在 Amazon QLDB 中，使用AVG函數傳回輸入運算式值的平均值 (算術平均值)。此函數適用於數值，並忽略 null 或缺少值。

語法

```
AVG ( expression )
```

引數

###

該函數對其運作的數值資料類型的欄位名稱或運算式。

資料類型

支持的參數類型：

- `int`
- `decimal`
- `float`

返回類型：`decimal`

範例

```
SELECT AVG(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 147.19
SELECT AVG(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 2.
```

相關函數

- COUNT
- MAX
- MIN
- SIZE
- SUM

亞馬遜 QLDB 中的 CAST 函數

在 Amazon QLDB 中，使用CAST函數將指定運算式評估為值，並將該值轉換為指定的目標資料類型。如果無法進行轉換，函數會傳回錯誤。

語法

```
CAST ( expression AS type )
```

引數

###

評估為函數轉換之值的欄位名稱或運算式。轉換 Null 值會傳回 Null。此參數可以是任何支援的參數資料類型。

##

要轉換的目標資料類型名稱。此參數可以是受支援的參數之一資料類型。

傳回類型

由類型引數指定的資料*##*。

範例

下列範例顯示未知類型 (NULL或MISSING) 的傳播。

```
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)
```

任何不是未知類型的值都無法轉換為NULL或MISSING。

```
CAST(true AS null)    -- error
CAST(true AS missing) -- error
CAST(1    AS null)    -- error
CAST(1    AS missing) -- error
```

下列範例顯示鑄造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'`)
```

下列範例顯示鑄造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
```

下列範例顯示鑄造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
```

下列範例顯示鑄造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
```

下列範例顯示鑄造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
```

下列範例顯示鑄造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'
```

下列範例顯示鑄造AS 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"
```

下列範例顯示鑄造AS struct。

```
CAST(`{ a: 1 }` AS struct) -- {a:1}
CAST(true       AS struct) -- err
```

下列範例顯示鑄造AS 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
```

下列範例是可執行的陳述式，其中包含一些先前範例。

```
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"
```

相關函數

- TO_STRING
- TO_TIMESTAMP

亞馬遜 QLDB 中的字符長度函數

在 Amazon QLDB 中，使用CHAR_LENGTH函數傳回指定字串中的字元數，其中字元定義為單一 unicode 程式碼點。

語法

```
CHAR_LENGTH ( string )
```

CHAR_LENGTH是的同義詞亞馬遜 QLDB 中的字符長度函數。

引數

string

函數評估的欄位名稱或資料類型string的運算式。

傳回類型

int

範例

```
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)
```

相關函數

- LOWER
- SUBSTRING
- TRIM
- UPPER

亞馬遜 QLDB 中的字符長度函數

CHAR_LENGTH 函數的同義詞。

請參閱 亞馬遜 QLDB 中的字符長度函數。

亞馬遜 QLDB 中的合併功能

在 Amazon QLDB 中，指定一個或多個引數的清單，請使用COALESCE函數從左到右的順序評估引數，並傳回不是未知類型 (NULL或MISSING) 的第一個值。如果所有參數類型都未知，則結果為NULL。

該COALESCE函數不會傳播NULL和MISSING。

語法

```
COALESCE ( expression [, expression, ... ] )
```

引數

###

函數評估的一個或多個欄位名稱或運算式的清單。每個引數都可以是任何支援的引數資料類型。

傳回類型

任何支援的資料類型。傳回類型與評估為非空值和非遺漏值的第一個運算式類型相同NULL或相同。

範例

```
SELECT COALESCE(1, null) FROM << 0 >>          -- 1
SELECT COALESCE(null, null, 1) FROM << 0 >>    -- 1
SELECT COALESCE(null, 'string') FROM << 0 >>  -- "string"
```

相關函數

- EXISTS
- NULLIF

亞馬遜 QLDB 中的計數功能

在 Amazon QLDB 中，使用COUNT函數傳回指定運算式所定義的文件數目。此功能有兩種變化：

- `COUNT(*)`— 計算目標資料表中的所有文件，無論它們是否包含 null 值或缺少值。
- `COUNT(expression)`— 計算特定現有欄位或運算式中具有非空值的文件數目。

> **Warning**
>
> 該`COUNT`函數未經過優化，因此我們不建議在沒有索引查找的情況下使用它。當您在沒有索引查閱的情況下在 QLDB 中執行查詢時，它會叫用完整資料表掃描。這可能會造成大型資料表的效能問題，包括並行衝突和交易逾時。
>
> 若要避免資料表掃描，您必須在索引欄位或文件 ID 上使用相等運算子 (`=`或`IN`) 執行具有述`WHERE`詞子句的陳述式。如需詳細資訊，請參閱 最佳化查詢效能。

語法

```
COUNT ( * | expression )
```

引數

###

該函數對其運作的欄位名稱或運算式。此參數可以是任何支援的參數資料類型。

傳回類型

`int`

範例

```
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
```

相關函數

- AVG
- MAX
- MIN

- SIZE
- SUM

亞馬遜 QLDB 中的日期添加功能

在 Amazon QLDB 中，使用DATE_ADD函數按指定的間隔遞增指定的時間戳記值。

語法

```
DATE_ADD( datetimepart, interval, timestamp )
```

引數

######

該函數對其運作的日期或時間部分。此參數可為下列其中一個值：

- year
- month
- day
- hour
- minute
- second

##

整數；指定要新增至指定時間*##*的間隔。負整數會減去間隔。

timestamp

功能遞增的欄位名稱或資料類型timestamp的運算式。

離子時間戳文字值可以用反引號（`...`）表示。如需格式化時間戳記值的詳細資訊和範例，請參閱 Amazon Ion 規格文件中的時間戳記。

傳回類型

timestamp

範例

```
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
```

相關函數

- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

亞馬遜 QLDB 中的日期 _ 差異函數

在 Amazon QLDB 中，使用DATE_DIFF函數傳回兩個給定時間戳記的指定日期部分之間的差異。

語法

```
DATE_DIFF( datetimepart, timestamp1, timestamp2 )
```

引數

######

該函數對其運作的日期或時間部分。此參數可為下列其中一個值：

- year

- month
- day
- hour
- minute
- second

1, *#### 2*

函數比較的兩個欄位名稱或資料類型timestamp的運算式。如果*#### 2* 晚於*#### 1*，則結果為陽性。如果*#### 2* 早於*#### 1*，則結果為負值。

離子時間戳文字值可以用反引號（`...`）表示。如需格式化時間戳記值的詳細資訊和範例，請參閱 Amazon Ion 規格文件中的時間戳記。

傳回類型

int

範例

```
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
```

相關函數

- DATE_ADD
- EXTRACT

- TO_STRING
- TO_TIMESTAMP
- UTCNOW

亞馬遜 QLDB 中的存在功能

在 Amazon QLDB 中，給定一個 PartiQL 值，TRUE如果該值是一個非空集合，則使用該EXISTS函數來傳回。否則，此函數返回FALSE。如果輸入不EXISTS是容器，則結果為FALSE。

該EXISTS函數不會傳播NULL和MISSING。

語法

```
EXISTS ( value )
```

引數

#

該函數評估的欄位名稱或運算式。此參數可以是任何支援的參數資料類型。

傳回類型

bool

範例

```
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
```

相關函數

- COALESCE
- NULLIF

在亞馬遜 QLDB 提取功能

在 Amazon QLDB 中，使用EXTRACT函數從指定時間戳記傳回指定日期或時間部分的整數值。

語法

```
EXTRACT ( datetimepart FROM timestamp )
```

引數

######

函數擷取的日期或時間部分。此參數可為下列其中一個值：

- year
- month
- day
- hour
- minute
- second
- timezone_hour
- timezone_minute

timestamp

函數從中擷取的資料類型timestamp的欄位名稱或運算式。如果此參數是未知類型 (NULL或MISSING)，則函數會傳回NULL。

離子時間戳文字值可以用反引號 (`...`) 表示。如需格式化時間戳記值的詳細資訊和範例，請參閱 Amazon Ion 規格文件中的時間戳記。

傳回類型

`int`

範例

```
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
```

相關函數

- DATE_ADD
- DATE_DIFF
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

在亞馬遜 QLDB 下功能

在 Amazon QLDB 中，使用LOWER函數將指定字串中的所有大寫字元轉換為小寫字元。

語法

```
LOWER ( string )
```

引數

string

該函數轉換的數據類型string的字段名稱或表達式。

傳回類型

`string`

範例

```
SELECT LOWER('AbCdEfG!@#$') FROM << 0 >> -- 'abcdefg!@#$'
```

相關函數

- CHAR_LENGTH
- SUBSTRING
- TRIM
- UPPER

亞馬遜 QLDB 中的 MAX 功能

在 Amazon QLDB 中，使用MAX函數傳回一組數值中的最大值。

語法

```
MAX ( expression )
```

引數

###

該函數對其運作的數值資料類型的欄位名稱或運算式。

資料類型

支持的參數類型：

- `int`
- `decimal`
- `float`

支持的返回類型：

- `int`
- `decimal`
- `float`

範例

```
SELECT MAX(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 442.30
SELECT MAX(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 3
```

相關函數

- AVG
- COUNT
- MIN
- SIZE
- SUM

亞馬遜 QLDB 中的最小功能

在 Amazon QLDB 中，使用`MIN`函數傳回一組數值中的最小值。

語法

```
MIN ( expression )
```

引數

###

該函數對其運作的數值資料類型的欄位名稱或運算式。

資料類型

支持的參數類型：

- int
- decimal
- float

支持的返回類型：

- int
- decimal
- float

範例

```
SELECT MIN(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 30.45
SELECT MIN(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 1
```

相關函數

- AVG
- COUNT
- MAX
- SIZE
- SUM

亞馬遜 QLDB 中的 NULLIF 函數

在 Amazon QLDB 中，給定兩個運算式，NULL如果兩個運算式評估為相同的值，則會使用該NULLIF函數來傳回。否則，此函數會傳回第一個運算式評估的結果。

該NULLIF函數不會傳播NULL和MISSING。

語法

```
NULLIF ( expression1, expression2 )
```

引數

1, *### 2*

函數比較的兩個欄位名稱或運算式。這些參數可以是任何支援的參數資料類型。

傳回類型

任何支援的資料類型。返回類型與第一個表達式的類型相同。NULL

範例

```
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
```

相關函數

- COALESCE
- EXISTS

亞馬遜 QLDB 中的尺寸功能

在 Amazon QLDB 中，使用SIZE函數傳回指定容器資料類型 (清單、結構或包裝) 中的元素數目。

語法

```
SIZE ( container )
```

引數

##

該函數對其運作的容器欄位名稱或運算式。

資料類型

支持的參數類型：

- list
- 結構
- 袋

返回類型：`int`

如果輸入不SIZE是容器，函數會擲回錯誤。

範例

```
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
```

相關函數

- AVG
- COUNT
- MAX
- MIN

- SUM

亞馬遜 QLDB 中的子字符串函數

在 Amazon QLDB 中，使用SUBSTRING函數從指定字串傳回子字串。子字符串從指定的開始索引開始，並在字符串的最後一個字符結束，或者在指定的長度結束。

語法

```
SUBSTRING ( string, start-index [, length ] )
```

引數

string

要從中擷取子字串的資料類型string的欄位名稱或運算式。

####

*##*內開始擷取的起始位置。這可以是負數。

*##*的第一個字元的索引為 1。

##

（可選）字符數（代碼點）從字*##*中提取，從開始*##*開始和結束於（*####* + *##*）-1。換言之，子字串的長度。這個數字不能是負數。

如果未提供此參數，函數會繼續執行，直到*##*結尾為止。

傳回類型

string

範例

```
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"
```

相關函數

- CHAR_LENGTH
- LOWER
- TRIM
- UPPER

亞馬遜 QLDB 中的 SUM 函數

在 Amazon QLDB 中，使用SUM函數傳回輸入欄位或運算式值的總和。此函數適用於數值，並忽略 null 或缺少值。

語法

```
SUM ( expression )
```

引數

###

該函數對其運作的數值資料類型的欄位名稱或運算式。

資料類型

支持的參數類型：

- int
- decimal
- float

支持的返回類型：

- int-對於整數引數
- decimal— 適用於小數或浮點引數

範例

```
SELECT SUM(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 735.95
SELECT SUM(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 6
```

相關函數

- AVG
- COUNT
- MAX
- MIN
- SIZE

亞馬遜 QLDB 中的字符串函數

在 Amazon QLDB 中，使用TO_STRING函數以指定的格式模式傳回指定時間戳記的字串表示。

語法

```
TO_STRING ( timestamp, 'format' )
```

引數

timestamp

該函數轉換為字符串的數據類型timestamp的字段名稱或表達式。

離子時間戳文字值可以用反引號 (`...`) 表示。如需格式化時間戳記值的詳細資訊和範例，請參閱 Amazon Ion 規格文件中的時間戳記。

format

字串常值，指定結果的格式模式，以其日期部分計算。如需有效格式，請參閱時間戳格式字串。

傳回類型

`string`

範例

```
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"
```

相關函數

- CAST
- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_TIMESTAMP

- UTCNOW

亞馬遜 QLDB 中的時間戳記函數

在 Amazon QLDB 中，指定代表時間戳記的字串，請使用TO_TIMESTAMP函數將字串轉換為timestamp資料類型。這是的反轉操作TO_STRING。

語法

```
TO_TIMESTAMP ( string [, 'format' ] )
```

引數

string

該函數轉換為時間戳記的數據類型string的字段名稱或表達式。

format

(選擇性) 以日期部分定義輸入字#格式模式的字串常值。如需有效格式，請參閱時間戳格式字串。

如果省略這個引數，函數會假設字#是標準 Ion 時間戳記的格式。這是使用此函數解析 Ion 時間戳的推薦方法。

使用單一字元格式符號 (例如y、、、、、、、、s) 時MdH，填補零是選擇性的hm，但對於填補零的變體 (例如yyyy、、、、、、MM、ddHHhhmm、ss) 則需要填補零。

特殊處理給予兩位數年（格式符號yy）。1900 被添加到大於或等於 70 的值，並將 2000 添加到小於 70 的值中。

月份名稱和 AM 或 PM 指標不區分大小寫。

傳回類型

timestamp

範例

```
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
```

相關函數

- CAST
- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- UTCNOW

亞馬遜 QLDB 中的修剪功能

在 Amazon QLDB 中，透過移除前導和尾端空格或一組指定的字元，使用TRIM函數修剪指定的字串。

語法

```
TRIM ( [ LEADING | TRAILING | BOTH [ characters ] FROM ] string)
```

引數

LEADING

(選擇性) 表示要從字串開頭移除空格或指定#元。如果未指定，則預設行為為BOTH。

TRAILING

(選擇性) 表示要從字串結尾移除空格或指定#元。如果未指定，則預設行為為BOTH。

BOTH

(選擇性) 表示要從字串的開頭和結尾移除開頭和結尾的空格或指定#元。

characters

(選擇性) 要移除的字元集，指定為string.

如果未提供此參數，則會移除空格。

string

功能修剪的數據類型string的字段名稱或表達式。

傳回類型

string

範例

```
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         "
```

相關函數

- CHAR_LENGTH
- LOWER
- SUBSTRING
- UPPER

亞馬遜 QLDB 中的 TXID 功能

在 Amazon QLDB 中，使用TXID函數傳回您正在執行之目前陳述式的唯一交易 ID。這是當目前交易確認至分錄時，指派給文件txId中繼資料欄位的值。

語法

```
TXID()
```

引數

無

傳回類型

`string`

範例

```
SELECT TXID() FROM << 0 >>  -- "L7S9iJqcn9W2M4qOEn27ay"
SELECT TXID() FROM Person WHERE GovId = 'LEWISR261LL'  -- "BKeMb48PNyvHWJGZHkaodG"
```

亞馬遜 QLDB 中的上層功能

在 Amazon QLDB 中，使用UPPER函數將指定字串中的所有小寫字元轉換為大寫字元。

語法

```
UPPER ( string )
```

引數

`string`

該函數轉換的數據類型`string`的字段名稱或表達式。

傳回類型

`string`

範例

```
SELECT UPPER('AbCdEfG!@#$') FROM << 0 >> -- 'ABCDEFG!@#$'
```

相關函數

- CHAR_LENGTH
- LOWER
- SUBSTRING
- TRIM

亞馬遜 QLDB 中的轉換功能

在 Amazon QLDB 中，使用UTCNOW函數將目前的時間 (以國際標準時間 (UTC) 傳回為timestamp.

語法

```
UTCNOW()
```

此函數會要求您在SELECT查詢中指定FROM子句。

引數

無

傳回類型

timestamp

範例

```
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'
```

相關函數

- DATE_ADD
- DATE_DIFF

- EXTRACT
- TO_STRING
- TO_TIMESTAMP

時間戳格式字串

本節提供時間戳記格式字串的參考資訊。

時間戳記格式字串會套用至TO_STRING和TO_TIMESTAMP函數。這些字串可以包含日期部分分隔符號 (例如- '/', '', 或: '') 和下列格式符號。

格式	範例	描述
yy	70	兩位數年份
y	1970	四位數年份
yyyy	1970	填補零的四位數年份
M	1	月整數
MM	01	填補零的某月整數
MMM	1 月	精簡的月名稱
MMMM	一月	完整的月名稱
d	2	一個月中的某一天 (1-31)
dd	02	填補零的某月某日 (1-31)
a	AM 或 PM	正午指標 (用於 12 小時制)
h	3	小時（12 小時制，1-12 小時制）
hh	03	零填充小時（12 小時制時鐘，01—12）
H	3	小時（二十四小時制，0-23）

格式	範例	描述
HH	03	零填充時數（24 小時制，00—23）
m	4	分鐘 (0 至 59 秒)
mm	04	零填充分鐘數 (00—59)
s	5	秒 (0 至 59)
ss	05	零填充秒數 (00—59)
S	0	幾分之幾秒 (精確度：0.1，範圍：0.0—0.9)
SS	06	幾分之幾秒 (精確度：0.01，範圍：0.0—0.99)
SSS	060	幾分之幾秒 (精確度：0.001，範圍：0.—0.999)
X	+07 或 Z 軸	如果位移為 0，則在小時或「Z」內位移
XX	+0700 or Z	如果位移為 0，則在小時和分鐘或「Z」內位移
XXX	晚上 7 點或 Z	如果位移為 0，則在小時和分鐘或「Z」內位移
x	+07	以小時為單位，從 UTC 偏移
xx	+0700	在小時和分鐘內位移
xxx	+ 7：00	在小時和分鐘內位移

亞馬遜 QLDB 中的存儲過程

在 Amazon QLDB 中，您可以使用命EXEC令，以下列語法執行 PartiQL 預存程序。

```
EXEC stored_procedure_name argument [, ... ]
```

QLDB 僅支援以下系統預存程序：

主題

- 亞馬遜 QLDB 中的修訂存儲過程

亞馬遜 QLDB 中的修訂存儲過程

> **Note**
>
> 在 2021 年 7 月 22 日之前建立的任何分類帳目前都不符合編輯資格。您可以在 Amazon QLDB 主控台上檢視分類帳的建立時間。

在 Amazon QLDB 中，使用REDACT_REVISION預存程序永久刪除索引儲存和日誌儲存中的個別非作用中文件修訂版。這個預存程序會刪除指定修訂版本中的所有使用者資料。不過，它會保持日誌序列和文件中繼資料 (包括文件 ID 和雜湊) 不變。此操作是不可逆轉的。

指定的文件修訂版本必須是記錄中的非使用中修訂版本。文件的最新使用中修訂版本不符合密文的資格。

在您執行此預存程序來提交密文要求之後，QLDB 會以非同步方式處理資料的密文。完成密文之後，指定修訂版本 (以data結構表示) 的使用者資料會被新dataHash欄位取代。此欄位的值是已移除data結構的 Amazon Ion 雜湊值。因此，分類帳會維護其整體資料完整性，並透過現有的驗證 API 作業維持密碼編譯驗證。

有關具有範例資料的密文操作範例，請參閱密文範例中的編輯文件修訂版本。

> **Note**
>
> 若要瞭解如何控制在特定資料表上執行此 PartiQL 命令的存取權，請參閱開始使用 Amazon QLDB 中的標準許可模式。

主題

- 密文考量與限制

- 語法
- 引數
- 傳回值
- 範例

密文考量與限制

在 Amazon QLDB 中開始進行資料編輯之前，請務必檢閱下列考量和限制：

- REDACT_REVISION預存程序會將您的使用者資料鎖定在個別的非使用中文件修訂版本中。若要編輯多個修訂版本，您必須針對每個修訂執行一次預存程序。您可以為每個異動編輯一個修訂。
- 若要編輯文件修訂版本中的特定欄位，您必須先使用個別的資料處理語言 (DML) 陳述式來修改修訂版本。如需詳細資訊，請參閱標記版本中的特定欄位。
- QLDB 收到密文請求後，您無法取消或更改請求。若要確認密文是否完成，您可以檢查修訂版本的data結構是否已由dataHash欄位取代。如需進一步了解，請參閱 檢查密文是否完成。
- 編修不會影響任何在 QLDB 服務外部複寫的 QLDB 資料。這包括任何對 Amazon S3 的匯出以及到 Amazon Kinesis Data Streams。您必須使用其他資料保留方法來管理儲存在 QLDB 以外的任何資料。
- 編修不會影響日誌中記錄的 PartiQL 陳述式中的常值。最佳實務是，您應使用變數的預留位置，而不是常值執行參數化的陳述式。預留位置會以問號 (?) 的形式寫入日誌，而不是任何可能需要密文的敏感資訊。

  若要了解如何使用 QLDB 驅動程式以程式設計方式執行 PartiQL 陳述式，請參閱開始使用驅動程式.

語法

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

引數

`####`

要編輯之文件修訂版本的分錄區塊位置。地址是具有兩個字段的 Amazon 離子結構：strandId和sequenceNo。

這是由反引號表示的離子文字值。例如：

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

若要瞭解如何尋找區塊位址，請參閱查詢文件元資料。

'*#####*'

您要編輯其文件修訂版本之表格的唯一 ID，以單引號表示。

若要瞭解如何尋找資料表 ID，請參閱查詢系統目錄。

「*#####*」

要編輯之修訂的唯一文件 ID，以單引號表示。

若要瞭解如何尋找文件 ID，請參閱查詢文件元資料。

傳回值

Amazon Ion 結構，代表要編輯的文件修訂版，格式如下。

```
{
  blockAddress: {
    strandId: String,
    sequenceNo: Int
  },
  tableId: String,
  documentId: String,
  version: Int
}
```

傳回結構欄位

- `blockAddress`— 要編輯之修訂的日誌區塊位置。一個地址有下列兩個欄位。
  - `strandId`— 包含區塊之日誌鏈的唯一 ID。
  - `sequenceNo`— 指定圖塊在鏈中的位置的索引號碼。
- `tableId`— 您正在編輯其修訂版本的表格的唯一 ID。
- `documentId`— 要編輯的修訂版本的唯一文件 ID。
- `version`— 要編輯的文件修訂版本的版本號碼。

以下是帶有示例數據的傳回結構。

```
{
  blockAddress: {
    strandId: "CsRnx0RDoNK6ANEEePa1ov",
    sequenceNo: 134
  },
  tableId: "6GZumdHggklLdMGyQq9DNX",
  documentId: "IXlQPSbfyKMIIsygePeKrZ",
  version: 0
}
```

範例

```
EXEC REDACT_REVISION `{strandId:"7z2P0AyQKWD8oFYmGNhi8D", sequenceNo:7}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

亞馬遜 QLDB 中的運營商

在亞馬遜 QLDB 的 PartiQL 支持以下 SQL 標準運算符。

> **Note**
>
> QLDB 目前不支援任何未包含在此清單中的 SQL 運算子。

算術運算子

運算子	描述
+	加
-	減
*	Multiply
/	Divide
%	模

比較運算子

運算子	描述
=	等於
>	大於
<	小於
>=	大於或等於
<=	小於或等於
<>	不等於

邏輯運算子

運算子	描述
AND	如果以 AND 分隔的所有的條件為 TRUE，則為 TRUE
BETWEEN	如果運算元在比較範圍內，則為 TRUE
IN	如果運算元等於表達式清單中的一個表達式，則為 TRUE
IS	TRUE如果操作數是給定的數據類型，包括NULL或MISSING
LIKE	TRUE如果操作數匹配的模式
NOT	反轉指定布林表達式的值
OR	TRUE如果任何分隔的條件OR是TRUE

字串運算子

運算子	描述
||	在||運算符的任一側連接兩個字符串，並返回連接的字符串。如果有一個或兩個字符串NULL，則連接的結果為 null。

亞馬遜 QLDB 中的保留關鍵字

以下是在亞馬遜 QLDB 中保留的 PartiQL 關鍵字列表。您可以使用保留關鍵字做為帶雙引號的帶引號的識別碼 (例如，`"user"`)。如需有關 QLDB 中的 PartiQL 引用慣例的詳細資訊，請參閱使用 PartiQL 查詢離子。

> **Important**
>
> 此列表中的關鍵字都被視為保留，因為 PartiQL 與 SQL-92 向後兼容。但是，QLDB 僅支援這些保留字的子集。如需 QLDB 目前支援的 SQL 關鍵字清單，請參閱下列主題：
>
> - PartiQL 函數
> - PartiQL 運算子
> - PartiQL 命令

```
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
```

亞馬遜 QLDB 中的亞馬遜離子數據格式參考

亞馬遜 QLDB 使用資料表示法模型，將亞馬遜離子與一個 PartiQL 類型的子集合在一起。本節提供 Ion 文件資料格式的參考概觀，與其與 PartiQL 整合不同。

在亞馬遜 QLDB 中使用 PartiQL 查詢離子

如需使用 QLDB 中的 PartiQL 查詢離子資料的語法和語意，請參閱亞馬遜 QLDB PartiQL 參考資料使用 PartiQL 查詢離子中的。

如需在 QLDB 分類帳中查詢和處理 Ion 資料的程式碼範例，請參閱Amazon Ion 程式碼範例和使用 Amazon Amazon Amazon ION。

主題

- 什麼是 Amazon Ion？
- Ion 規格
- JSON 相容
- JSON 中的擴充套件
- Ion 文本範例
- API 參考
- QLDB 中的亞馬遜離子代碼示例

什麼是 Amazon Ion？

Ion 是一種開源的，豐富的類型，自我描述的分層數據序列化格式，最初是在 Amazon 內部開發的。它基於抽象數據模型，可讓您同時存儲結構化和非結構化數據。這是 JSON 的超集合，這意味著任何有效的 JSON 文檔也是一個有效的離子文檔。本指南假設有 JSON 的基準工作知識。如果您還不熟悉 JSON，請參閱簡介 JSON 以取得詳細資訊。

您可以以人類可讀的文本形式或二進制編碼形式互換表示 Ion 文檔。與 JSON 一樣，文本表單易於閱讀和寫入，支持快速原型製作。二進位編碼對於持續、傳輸和剖析而言更加緊湊且有效率。離子處理器可以在兩種格式之間進行轉碼，以表示完全相同的一組數據結構，而不會丟失任何數據。此功能可讓應用程式針對不同使用案例最佳化處理資料的方式。

> **Note**
>
> Ion 數據模型是嚴格基於價值的，不支持引用。因此，數據模型可以表示可以嵌套到任意深度，但不是有向圖的數據層次結構。

Ion 規格

有關 Ion 核心數據類型的完整列表，其中包含完整描述和值格式詳細信息，請參閱 Amazon GitHub 網站上的 Ion 規範文檔。

為了簡化應用程式開發，Amazon Ion 提供可為您處理 Ion 資料的用戶端程式庫。如需處理 Ion 資料的常見使用案例的程式碼範例，請參閱上的 Amazon Ion 食譜 GitHub。

JSON 相容

與 JSON 類似，您可以使用一組原始資料類型和一組遞迴定義的容器類型來撰寫 Amazon Ion 文件。Ion 包括以下傳統的 JSON 數據類型：

- `null`：一個通用的，不具類型的空（空）值。此外，如以下部分所述，Ion 支持每個基本類型的不同 null 類型。
- `bool`：布林值。
- `string`: 文字常值。
- `list`：值的排序異構集合。
- `struct`：名稱/值組的無序集合。與 JSON 一樣，每個名稱`struct`允許多個值，但通常不鼓勵這樣做。

JSON 中的擴充套件

數字類型

Amazon Ion 不是模糊的 JSON 類number型，而是嚴格將數字定義為以下類型之一：

- `int`：任意大小的有符號整數。
- `decimal`: 任意精度的十進制編碼實數。
- `float`: 二進位編碼的浮點數 (64 位元 IEEE)。

解析文件時，Ion 處理器會依照下列方式指定編號類型：

- `int`: 沒有指數或小數點的數字 (例如，`100200`)。
- `decimal`: 具有小數點且沒有指數的數字 (例如，`0.00001`,`200.0`)。
- `float`：具有指數的數字，例如科學記數法或 E 表示法（例如`2e0`，，`3.1e-4`）。

新資料類型

亞馬遜離子添加了以下數據類型：

- `timestamp`：任意精度的日期/時間/時區時刻。
- `symbol`：Unicode 符號原子（如標識符）。
- `blob`: 使用者定義編碼的二進位資料。
- `clob`: 使用者定義編碼的文字資料。
- `sexp`：具有應用程序定義語義的值的有序集合。

空類型

除了 JSON 定義的泛型空類型之外，Amazon Ion 還支援每個原始類型的不同空類型。這表示缺乏價值，同時保持嚴格的數據類型。

```
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
```

Ion 文本範例

```
// 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
    ],
}
```

API 參考

- 離子去
- 離子爪哇
- 離子-JS
- 離子蟒蛇

QLDB 中的亞馬遜離子代碼示例

本節提供程式碼範例，這些範例透過讀取和寫入 Amazon QLDB 分類帳中的文件值來處理 Amazon Ion 資料。程式碼範例使用 QLDB 驅動程式，在分類帳上執行 PartiQL 陳述式。這些範例是中範例應用程式的一部分使用範例應用程式教學課程開始使用 Amazon QLDB，是AWS範例 GitHub 網站上的開放原始碼。

如需顯示處理 Ion 資料常見使用案例的一般程式碼範例，請參閱上的 Amazon Ion 食譜 GitHub。

執行程式碼

每種程式設計語言的教學課程程式碼會執行下列步驟：

1. Connect 至範`vehicle-registration`例分類帳。
2. 建立名稱為的資料表`IonTypes`。
3. 使用單一欄位將文件插入到表格`Name`中。
4. 對於每種支持的 Ion 數據類型：
   a. 使用資料類型的常值來更新文件的欄`Name`位。
   b. 查詢資料表以取得文件的最新修訂版本。
   c. 透過檢查是否符合預期的類型，驗證的值是否`Name`保留其原始資料類型性質。
5. 放下`IonTypes`桌子。

> **Note**
>
> 執行此教學課程代碼之前，您必須建立名稱為的分類帳`vehicle-registration`。

Java

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
```

```
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonBlob;
import com.amazon.ion.IonBool;
import com.amazon.ion.IonClob;
import com.amazon.ion.IonDecimal;
import com.amazon.ion.IonFloat;
import com.amazon.ion.IonInt;
import com.amazon.ion.IonList;
import com.amazon.ion.IonNull;
import com.amazon.ion.IonSexp;
import com.amazon.ion.IonString;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonSymbol;
import com.amazon.ion.IonTimestamp;
import com.amazon.ion.IonValue;
import com.amazon.ion.Timestamp;
import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import java.util.Collections;
import java.util.List;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;

/**
```

```
 * Insert all the supported Ion types into a ledger and verify that they are stored
 and can be retrieved properly, retaining
 * their original properties.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-credentials.html
 */
public class InsertIonTypes {
    public static final Logger log = LoggerFactory.getLogger(InsertIonTypes.class);
    public static final String TABLE_NAME = "IonTypes";

    private InsertIonTypes() {}

    /**
     * Update a document's Name value in the database. Then, query the value of the
 Name key and verify the expected Ion type was
     * saved.
     *
     * @param txn
     *              The {@link TransactionExecutor} for statement execution.
     * @param ionValue
     *              The {@link IonValue} to set the document's Name value to.
     *
     * @throws AssertionError when no value is returned for the Name key or if the
 value does not match the expected type.
     */
    public static void updateRecordAndVerifyType(final TransactionExecutor txn,
 final IonValue ionValue) {
        final String updateStatement = String.format("UPDATE %s SET Name = ?",
 TABLE_NAME);
        final List<IonValue> parameters = Collections.singletonList(ionValue);
        txn.execute(updateStatement, parameters);
        log.info("Updated document.");

        final String searchQuery = String.format("SELECT VALUE Name FROM %s",
 TABLE_NAME);
        final Result result = txn.execute(searchQuery);

        if (result.isEmpty()) {
            throw new AssertionError("Did not find any values for the Name key.");
        }
        for (IonValue value : result) {
            if (!ionValue.getClass().isInstance(value)) {
```

```
                throw new AssertionError(String.format("The queried value, %s, is
not an instance of %s.",
                        value.getClass().toString(),
ionValue.getClass().toString()));
            }
            if (!value.getType().equals(ionValue.getType())) {
                throw new AssertionError(String.format("The queried value type, %s,
does not match %s.",
                        value.getType().toString(), ionValue.getType().toString()));
            }
        }

        log.info("Successfully verified value is instance of {} with type {}.",
ionValue.getClass().toString(),
                ionValue.getType().toString());
    }

    /**
     * Delete a table.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              The name of the table to delete.
     */
    public static void deleteTable(final TransactionExecutor txn, final String
tableName) {
        log.info("Deleting {} table...", tableName);
        final String statement = String.format("DROP TABLE %s", tableName);
        txn.execute(statement);
        log.info("{} table successfully deleted.", tableName);
    }

    public static void main(final String... args) {
        final IonBlob ionBlob = Constants.SYSTEM.newBlob("hello".getBytes());
        final IonBool ionBool = Constants.SYSTEM.newBool(true);
        final IonClob ionClob = Constants.SYSTEM.newClob("{{'This is a CLOB of
text.'}}".getBytes());
        final IonDecimal ionDecimal = Constants.SYSTEM.newDecimal(0.1);
        final IonFloat ionFloat = Constants.SYSTEM.newFloat(0.2);
        final IonInt ionInt = Constants.SYSTEM.newInt(1);
        final IonList ionList = Constants.SYSTEM.newList(new int[]{1, 2});
        final IonNull ionNull = Constants.SYSTEM.newNull();
        final IonSexp ionSexp = Constants.SYSTEM.newSexp(new int[]{2, 3});
```

```
        final IonString ionString = Constants.SYSTEM.newString("string");
        final IonStruct ionStruct = Constants.SYSTEM.newEmptyStruct();
        ionStruct.put("brand", Constants.SYSTEM.newString("ford"));
        final IonSymbol ionSymbol = Constants.SYSTEM.newSymbol("abc");
        final IonTimestamp ionTimestamp =
Constants.SYSTEM.newTimestamp(Timestamp.now());

        final IonBlob ionNullBlob = Constants.SYSTEM.newNullBlob();
        final IonBool ionNullBool = Constants.SYSTEM.newNullBool();
        final IonClob ionNullClob = Constants.SYSTEM.newNullClob();
        final IonDecimal ionNullDecimal = Constants.SYSTEM.newNullDecimal();
        final IonFloat ionNullFloat = Constants.SYSTEM.newNullFloat();
        final IonInt ionNullInt = Constants.SYSTEM.newNullInt();
        final IonList ionNullList = Constants.SYSTEM.newNullList();
        final IonSexp ionNullSexp = Constants.SYSTEM.newNullSexp();
        final IonString ionNullString = Constants.SYSTEM.newNullString();
        final IonStruct ionNullStruct = Constants.SYSTEM.newNullStruct();
        final IonSymbol ionNullSymbol = Constants.SYSTEM.newNullSymbol();
        final IonTimestamp ionNullTimestamp = Constants.SYSTEM.newNullTimestamp();


        ConnectToLedger.getDriver().execute(txn -> {
            CreateTable.createTable(txn, TABLE_NAME);
            final Document document = new
Document(Constants.SYSTEM.newString("val"));
            InsertDocument.insertDocuments(txn, TABLE_NAME,
Collections.singletonList(document));

            updateRecordAndVerifyType(txn, ionBlob);
            updateRecordAndVerifyType(txn, ionBool);
            updateRecordAndVerifyType(txn, ionClob);
            updateRecordAndVerifyType(txn, ionDecimal);
            updateRecordAndVerifyType(txn, ionFloat);
            updateRecordAndVerifyType(txn, ionInt);
            updateRecordAndVerifyType(txn, ionList);
            updateRecordAndVerifyType(txn, ionNull);
            updateRecordAndVerifyType(txn, ionSexp);
            updateRecordAndVerifyType(txn, ionString);
            updateRecordAndVerifyType(txn, ionStruct);
            updateRecordAndVerifyType(txn, ionSymbol);
            updateRecordAndVerifyType(txn, ionTimestamp);

            updateRecordAndVerifyType(txn, ionNullBlob);
            updateRecordAndVerifyType(txn, ionNullBool);
```

```
            updateRecordAndVerifyType(txn, ionNullClob);
            updateRecordAndVerifyType(txn, ionNullDecimal);
            updateRecordAndVerifyType(txn, ionNullFloat);
            updateRecordAndVerifyType(txn, ionNullInt);
            updateRecordAndVerifyType(txn, ionNullList);
            updateRecordAndVerifyType(txn, ionNullSexp);
            updateRecordAndVerifyType(txn, ionNullString);
            updateRecordAndVerifyType(txn, ionNullStruct);
            updateRecordAndVerifyType(txn, ionNullSymbol);
            updateRecordAndVerifyType(txn, ionNullTimestamp);

            deleteTable(txn, TABLE_NAME);
        });
    }

    /**
     * This class represents a simple document with a single key, Name, to use for
 the IonTypes table.
     */
    private static class Document {
        private final IonValue name;

        @JsonCreator
        private Document(@JsonProperty("Name") final IonValue name) {
            this.name = name;
        }

        @JsonProperty("Name")
        private IonValue getName() {
            return name;
        }
    }
}
```

Node.js

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
```

```
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { AssertionError } from "assert";
import { dom, IonType, IonTypes } from "ion-js";

import { insertDocument } from "./InsertDocument";
import { getQldbDriver } from "./ConnectToLedger";
import { createTable } from "./CreateTable";
import { error, log } from "./qldb/LogUtil";

const TABLE_NAME: string = "IonTypes";

/**
 * Delete a table.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName Name of the table to delete.
 * @returns Promise which fulfills with void.
 */
export async function deleteTable(txn: TransactionExecutor, tableName: string):
 Promise<void> {
    log(`Deleting ${tableName} table...`);
    const statement: string = `DROP TABLE ${tableName}`;
    await txn.execute(statement);
    log(`${tableName} table successfully deleted.`);
}

/**
```

```
 * Update a document's Name value in QLDB. Then, query the value of the Name key and
 verify the expected Ion type was
 * saved.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param parameter The IonValue to set the document's Name value to.
 * @param ionType The Ion type that the Name value should be.
 * @returns Promise which fulfills with void.
 */
async function updateRecordAndVerifyType(
    txn: TransactionExecutor,
    parameter: any,
    ionType: IonType
): Promise<void> {
    const updateStatement: string = `UPDATE ${TABLE_NAME} SET Name = ?`;
    await txn.execute(updateStatement, parameter);
    log("Updated record.");

    const searchStatement: string = `SELECT VALUE Name FROM ${TABLE_NAME}`;
    const result: Result = await txn.execute(searchStatement);

    const results: dom.Value[] = result.getResultList();

    if (0 === results.length) {
        throw new AssertionError({
            message: "Did not find any values for the Name key."
        });
    }

    results.forEach((value: dom.Value) => {
        if (value.getType().binaryTypeId !== ionType.binaryTypeId) {
            throw new AssertionError({
                message: `The queried value type, ${value.getType().name}, does not
 match expected type, ${ionType.name}.`
            });
        }
    });

    log(`Successfully verified value is of type ${ionType.name}.`);
}

/**
 * Insert all the supported Ion types into a table and verify that they are stored
 and can be retrieved properly,
 * retaining their original properties.
```

```
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await createTable(txn, TABLE_NAME);
            await insertDocument(txn, TABLE_NAME, [{ "Name": "val" }]);
            await updateRecordAndVerifyType(txn, dom.load("null"), IonTypes.NULL);
            await updateRecordAndVerifyType(txn, true, IonTypes.BOOL);
            await updateRecordAndVerifyType(txn, 1, IonTypes.INT);
            await updateRecordAndVerifyType(txn, 3.2, IonTypes.FLOAT);
            await updateRecordAndVerifyType(txn, dom.load("5.5"), IonTypes.DECIMAL);
            await updateRecordAndVerifyType(txn, dom.load("2020-02-02"),
 IonTypes.TIMESTAMP);
            await updateRecordAndVerifyType(txn, dom.load("abc123"),
 IonTypes.SYMBOL);
            await updateRecordAndVerifyType(txn, dom.load("\"string\""),
 IonTypes.STRING);
            await updateRecordAndVerifyType(txn, dom.load("{{ \"clob\" }}"),
 IonTypes.CLOB);
            await updateRecordAndVerifyType(txn, dom.load("{{ blob }}"),
 IonTypes.BLOB);
            await updateRecordAndVerifyType(txn, dom.load("(1 2 3)"),
 IonTypes.SEXP);
            await updateRecordAndVerifyType(txn, dom.load("[1, 2, 3]"),
 IonTypes.LIST);
            await updateRecordAndVerifyType(txn, dom.load("{brand: ford}"),
 IonTypes.STRUCT);
            await deleteTable(txn, TABLE_NAME);
        });
    } catch (e) {
        error(`Error updating and validating Ion types: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

Python

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy, modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from datetime import datetime
from decimal import Decimal
from logging import basicConfig, getLogger, INFO

from amazon.ion.simple_types import IonPyBool, IonPyBytes, IonPyDecimal, IonPyDict,
 IonPyFloat, IonPyInt, IonPyList, \
    IonPyNull, IonPySymbol, IonPyText, IonPyTimestamp
from amazon.ion.simpleion import loads
from amazon.ion.symbols import SymbolToken
from amazon.ion.core import IonType

from pyqldbsamples.create_table import create_table
from pyqldbsamples.constants import Constants
from pyqldbsamples.insert_document import insert_documents
from pyqldbsamples.model.sample_data import convert_object_to_ion
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)

TABLE_NAME = 'IonTypes'


def update_record_and_verify_type(driver, parameter, ion_object, ion_type):
```

```
    """
    Update a record in the database table. Then query the value of the record and
verify correct ion type saved.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type parameter: :py:class:`amazon.ion.simple_types.IonPyValue`
    :param parameter: The Ion value or Python native type that is convertible to Ion
for filling in parameters of the
                      statement.

    :type
ion_object: :py:obj:`IonPyBool`/:py:obj:`IonPyBytes`/:py:obj:`IonPyDecimal`/:py:obj:`IonPyD

/:py:obj:`IonPyFloat`/:py:obj:`IonPyInt`/:py:obj:`IonPyList`/:py:obj:`IonPyNull`

/:py:obj:`IonPySymbol`/:py:obj:`IonPyText`/:py:obj:`IonPyTimestamp`
    :param ion_object: The Ion object to verify against.

    :type ion_type: :py:class:`amazon.ion.core.IonType`
    :param ion_type: The Ion type to verify against.

    :raises TypeError: When queried value is not an instance of Ion type.
    """
    update_query = 'UPDATE {} SET Name = ?'.format(TABLE_NAME)
    driver.execute_lambda(lambda executor: executor.execute_statement(update_query,
parameter))
    logger.info('Updated record.')

    search_query = 'SELECT VALUE Name FROM {}'.format(TABLE_NAME)
    cursor = driver.execute_lambda(lambda executor:
executor.execute_statement(search_query))

    for c in cursor:
        if not isinstance(c, ion_object):
            raise TypeError('The queried value is not an instance of
{}'.format(ion_object.__name__))

        if c.ion_type is not ion_type:
            raise TypeError('The queried value type does not match
{}'.format(ion_type))
```

```
    logger.info("Successfully verified value is instance of '{}' with type
 '{}'.".format(ion_object.__name__, ion_type))
    return cursor


def delete_table(driver, table_name):
    """
    Delete a table.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type table_name: str
    :param table_name: Name of the table to delete.

    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Deleting '{}' table...".format(table_name))
    cursor = driver.execute_lambda(lambda executor: executor.execute_statement('DROP
 TABLE {}'.format(table_name)))
    logger.info("'{}' table successfully deleted.".format(table_name))
    return len(list(cursor))


def insert_and_verify_ion_types(driver):
    """
    Insert all the supported Ion types and Python values that are convertible to Ion
 into a ledger and verify that they
    are stored and can be retrieved properly, retaining their original properties.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: A QLDB Driver object.
    """
    python_bytes = str.encode('hello')
    python_bool = True
    python_float = float('0.2')
    python_decimal = Decimal('0.1')
    python_string = "string"
    python_int = 1
    python_null = None
    python_datetime = datetime(2016, 12, 20, 5, 23, 43)
    python_list = [1, 2]
    python_dict = {"brand": "Ford"}
```

```
    ion_clob = convert_object_to_ion(loads('{{"This is a CLOB of text."}}'))
    ion_blob = convert_object_to_ion(python_bytes)
    ion_bool = convert_object_to_ion(python_bool)
    ion_decimal = convert_object_to_ion(python_decimal)
    ion_float = convert_object_to_ion(python_float)
    ion_int = convert_object_to_ion(python_int)
    ion_list = convert_object_to_ion(python_list)
    ion_null = convert_object_to_ion(python_null)
    ion_sexp = convert_object_to_ion(loads('(cons 1 2)'))
    ion_string = convert_object_to_ion(python_string)
    ion_struct = convert_object_to_ion(python_dict)
    ion_symbol = convert_object_to_ion(SymbolToken(text='abc', sid=123))
    ion_timestamp = convert_object_to_ion(python_datetime)

    ion_null_clob = convert_object_to_ion(loads('null.clob'))
    ion_null_blob = convert_object_to_ion(loads('null.blob'))
    ion_null_bool = convert_object_to_ion(loads('null.bool'))
    ion_null_decimal = convert_object_to_ion(loads('null.decimal'))
    ion_null_float = convert_object_to_ion(loads('null.float'))
    ion_null_int = convert_object_to_ion(loads('null.int'))
    ion_null_list = convert_object_to_ion(loads('null.list'))
    ion_null_sexp = convert_object_to_ion(loads('null.sexp'))
    ion_null_string = convert_object_to_ion(loads('null.string'))
    ion_null_struct = convert_object_to_ion(loads('null.struct'))
    ion_null_symbol = convert_object_to_ion(loads('null.symbol'))
    ion_null_timestamp = convert_object_to_ion(loads('null.timestamp'))

    create_table(driver, TABLE_NAME)
    insert_documents(driver, TABLE_NAME, [{'Name': 'val'}])
    update_record_and_verify_type(driver, python_bytes, IonPyBytes, IonType.BLOB)
    update_record_and_verify_type(driver, python_bool, IonPyBool, IonType.BOOL)
    update_record_and_verify_type(driver, python_float, IonPyFloat, IonType.FLOAT)
    update_record_and_verify_type(driver, python_decimal, IonPyDecimal,
IonType.DECIMAL)
    update_record_and_verify_type(driver, python_string, IonPyText, IonType.STRING)
    update_record_and_verify_type(driver, python_int, IonPyInt, IonType.INT)
    update_record_and_verify_type(driver, python_null, IonPyNull, IonType.NULL)
    update_record_and_verify_type(driver, python_datetime, IonPyTimestamp,
IonType.TIMESTAMP)
    update_record_and_verify_type(driver, python_list, IonPyList, IonType.LIST)
    update_record_and_verify_type(driver, python_dict, IonPyDict, IonType.STRUCT)
    update_record_and_verify_type(driver, ion_clob, IonPyBytes, IonType.CLOB)
    update_record_and_verify_type(driver, ion_blob, IonPyBytes, IonType.BLOB)
```

```
    update_record_and_verify_type(driver, ion_bool, IonPyBool, IonType.BOOL)
    update_record_and_verify_type(driver, ion_decimal, IonPyDecimal,
 IonType.DECIMAL)
    update_record_and_verify_type(driver, ion_float, IonPyFloat, IonType.FLOAT)
    update_record_and_verify_type(driver, ion_int, IonPyInt, IonType.INT)
    update_record_and_verify_type(driver, ion_list, IonPyList, IonType.LIST)
    update_record_and_verify_type(driver, ion_null, IonPyNull, IonType.NULL)
    update_record_and_verify_type(driver, ion_sexp, IonPyList, IonType.SEXP)
    update_record_and_verify_type(driver, ion_string, IonPyText, IonType.STRING)
    update_record_and_verify_type(driver, ion_struct, IonPyDict, IonType.STRUCT)
    update_record_and_verify_type(driver, ion_symbol, IonPySymbol, IonType.SYMBOL)
    update_record_and_verify_type(driver, ion_timestamp, IonPyTimestamp,
 IonType.TIMESTAMP)
    update_record_and_verify_type(driver, ion_null_clob, IonPyNull, IonType.CLOB)
    update_record_and_verify_type(driver, ion_null_blob, IonPyNull, IonType.BLOB)
    update_record_and_verify_type(driver, ion_null_bool, IonPyNull, IonType.BOOL)
    update_record_and_verify_type(driver, ion_null_decimal, IonPyNull,
 IonType.DECIMAL)
    update_record_and_verify_type(driver, ion_null_float, IonPyNull, IonType.FLOAT)
    update_record_and_verify_type(driver, ion_null_int, IonPyNull, IonType.INT)
    update_record_and_verify_type(driver, ion_null_list, IonPyNull, IonType.LIST)
    update_record_and_verify_type(driver, ion_null_sexp, IonPyNull, IonType.SEXP)
    update_record_and_verify_type(driver, ion_null_string, IonPyNull,
 IonType.STRING)
    update_record_and_verify_type(driver, ion_null_struct, IonPyNull,
 IonType.STRUCT)
    update_record_and_verify_type(driver, ion_null_symbol, IonPyNull,
 IonType.SYMBOL)
    update_record_and_verify_type(driver, ion_null_timestamp, IonPyNull,
 IonType.TIMESTAMP)
    delete_table(driver, TABLE_NAME)


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Insert all the supported Ion types and Python values that are convertible to Ion
 into a ledger and verify that they
    are stored and can be retrieved properly, retaining their original properties.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            insert_and_verify_ion_types(driver)
    except Exception as e:
        logger.exception('Error updating and validating Ion types.')
```

```
        raise e


if __name__ == '__main__':
    main()
```

Amazon QLDB API 參考參考

本章說明可透過 HTTP 存取的亞馬遜 QLDB 的低階 API 作業AWS Command Line Interface(AWS CLI)，或AWS開發套件：

- Amazon QLDB— QLDB 資源管理 API (也稱為控制平面。此 API 僅用於管理分類帳資源和非交易資料作業。您可以使用這些操作來建立、刪除、描述、列出及更新總帳。您也可以以密碼方式驗證日誌資料，以及匯出或串流日誌區塊。
- Amazon QLDB 工作階段工作階段— QLDB 交易資料 API。您可以使用此 API 在分類帳上執行資料交易PartiQL PartiQL陳述式。

> **Important**
>
> 而不是直接與QLDB 工作階段工作階段API，我們建議使用 QLDB 驅動程式或 QLDB 殼層在分類帳上執行資料交易。
>
> - 如果您使用的是AWS開發套件中，使用 QLDB 驅動程式。驅動程式提供以上的高階抽象層QLDB 工作階段工作階段數據 API 和管理SendCommand為您提供操作。如需相關資訊和支援的程式設計語言清單，請參閱開始使用驅動程式。
> - 如果您使用的是AWS CLI，使用 QLDB 外殼。殼層是使用 QLDB 驅動程式與總帳互動的命令列介面。如需相關資訊，請參閱 使用亞馬遜 QLDB 外殼 (僅限資料 API)。

主題

動作

Amazon QLDB 支援下列動作：

- CancelJournalKinesisStream
- CreateLedger
- DeleteLedger

- DescribeJournalKinesisStream
- DescribeJournalS3Export
- DescribeLedger
- ExportJournalToS3
- GetBlock
- GetDigest
- GetRevision
- ListJournalKinesisStreamsForLedger
- ListJournalS3Exports
- ListJournalS3ExportsForLedger
- ListLedgers
- ListTagsForResource
- StreamJournalToKinesis
- TagResource
- UntagResource
- UpdateLedger
- UpdateLedgerPermissionsMode

Amazon QLDB 會話支援下列動作：

- 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

服務：Amazon QLDB

結束指定的 Amazon QLDB 日誌串流。在可以取消流之前，其當前狀態必須是ACTIVE。

取消串流後，您無法重新啟動該串流。取消的 QLDB 串流資源需要 7 天的保留期限，因此在此限制過期後會自動刪除這些資源。

請求語法

```
DELETE /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

分類帳的名稱。

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必要：是

streamId

要取消之 QLDB 日誌資料流的 UUID (以 62 基編碼的文字表示)。

長度約束：固定長度為 22。

模式：^[A-Za-z-0-9]+$

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "StreamId": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

StreamId

已取消之 QLDB 日誌串流的 UUID (基本 62 編碼文字)。

類型：字串

長度約束：固定長度為 22。

模式：^[A-Za-z-0-9]+$

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

ResourcePreconditionNotMetException

作業失敗，因為未事先滿足條件。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

CreateLedger

服務：Amazon QLDB

在目前區域 AWS 帳戶 中建立新分類帳。

請求語法

```
POST /ledgers HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "Tags": {
      "string" : "string"
   }
}
```

URI 請求參數

請求不會使用任何 URI 參數。

請求主體

請求接受採用 JSON 格式的下列資料。

DeletionProtection

指定是否要保護分類帳，不讓任何使用者刪除。如果在建立分類帳時未定義，則預設會啟用此功能 (`true`)。

如果已啟用刪除保護，則必須先停用該功能才能刪除分類帳。您可以呼叫 `UpdateLedger` 操作將此參數設定為 `false` 以停用該功能。

類型：布林值

必要：否

KmsKey

AWS Key Management Service (AWS KMS) 中的密鑰用於分類帳中靜態數據的加密。如需詳細資訊，請參閱《Amazon QLDB 開發人員指南》中的靜態加密。

使用以下其中一個選項來指定此參數：

- AWS_OWNED_KMS_KEY：使用代表您擁有和管理 AWS 的 AWS KMS 密鑰。
- 未定義：依預設，使用 AWS 擁有的 KMS 金鑰。
- 有效的對稱客戶受管 KMS 金鑰：使用在您帳戶中建立、擁有和管理的指定對稱加密 KMS 金鑰。

  Amazon QLDB 不支援非對稱金鑰。如需詳細資訊，請參閱 AWS Key Management Service 開發人員指南中的使用對稱和非對稱金鑰。

若要指定客戶受管 KMS 金鑰，您可使用其金鑰 ID、Amazon Resource Name (ARN)、別名名稱或別名 ARN。使用別名名稱時，請加上 "alias/" 字首。若要指定不同的索引鍵 AWS 帳戶，您必須使用金鑰 ARN 或別名 ARN。

例如：

- 金鑰 ID：1234abcd-12ab-34cd-56ef-1234567890ab
- 金鑰 ARN：arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
- 別名名稱：alias/ExampleAlias
- 別名 ARN：arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias

如需詳細資訊，請參閱 AWS Key Management Service 開發人員指南中的金鑰識別碼 (KeyId)。

類型：字串

長度限制：最大長度為 1600。

必要：否

Name

您要建立分類帳的名稱。在您目前「區域」中的所有分類帳 AWS 帳戶 中，此名稱必須是唯一的。

分類帳名稱的命名限制定義於《Amazon QLDB 開發人員指南》中的 Amazon QLDB 中的配額。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必要：是

PermissionsMode

要指派給您希望建立分類帳的許可模式。此參數可以有下列其中一個值：

- ALLOW_ALL：一種舊版許可模式，可使用 API 層級精細程度啟用對分類帳的存取控制。

  此模式允許具有指定分類帳 SendCommand API 許可的使用者 (因此為 ALLOW_ALL)，在此分類帳中的任何資料表上執行所有 PartiQL 命令。此模式會忽略您為分類帳建立之任何資料表層級或命令層級 IAM 許可政策。

- STANDARD：(建議) 一種許可模式，可使用更細的層級啟用對分類帳、資料表和 PartiQL 命令的存取控制。

  根據預設，此模式會拒絕所有使用者在此分類帳中任何資料表上執行任何 PartiQL 命令的請求。若要允許執行 PartiQL 命令，除了分類帳的 SendCommand API 許可之外，您還必須為特定資料表資源和 PartiQL 動作建立 IAM 許可政策。如需相關資訊，請參閱《Amazon QLDB 開發人員指南》中的開始使用標準許可模式。

> **Note**
>
> 強烈建議您使用 STANDARD 許可模式來最大化分類帳資料的安全性。

類型：字串

有效值:ALLOW_ALL | STANDARD

必要：是

Tags

要新增為標記至您要建立之分類帳的索引鍵值配對。標籤鍵會區分大小寫。標籤值區分大小寫，並且可以為 null。

類型：字串到字串映射

地圖項目：0 個項目的最小數目。項目數上限為 200。

索引鍵長度限制：長度下限為 1。長度上限為 128。

值長度限制：最小長度為 0。長度上限為 256。

必要：否

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "CreationDateTime": number,
   "DeletionProtection": boolean,
   "KmsKeyArn": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "State": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

Arn

分類帳的 Amazon 資源名稱（ARN）。

類型：字串

長度限制：長度下限為 20。長度上限為 1600。

CreationDateTime

建立分類帳時的日期和時間 (以紀元時間格式表示)。（紀元時間格式是自世界標準時間 1970 年 1 月 1 日上午 12:00:00 以來經過的秒數。）

類型：Timestamp

DeletionProtection

指定是否要保護分類帳，不讓任何使用者刪除。如果在建立分類帳時未定義，則預設會啟用此功能 (`true`)。

如果已啟用刪除保護，則必須先停用該功能才能刪除分類帳。您可以呼叫 `UpdateLedger` 操作將此參數設定為 `false` 以停用該功能。

類型：布林值

KmsKeyArn

客戶的 ARN 管理 KMS 金鑰，分類帳用於靜態加密。如果未定義此參數，則分類帳會使用 AWS 擁有的 KMS 金鑰進行加密。

類型：字串

長度限制：長度下限為 20。長度上限為 1600。

Name

分類帳的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

PermissionsMode

您所建立之分類帳的權限模式。

類型：字串

有效值:ALLOW_ALL | STANDARD

State

分類帳的目前狀態。

類型：字串

有效值:CREATING | ACTIVE | DELETING | DELETED

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

LimitExceededException

您已達到允許的最大資源數量上限。

HTTP 狀態碼：400

ResourceAlreadyExistsException

指定的資源已存在。

HTTP 狀態碼：409

ResourceInUseException

目前無法修改指定的資源。

HTTP 狀態碼：409

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

DeleteLedger

服務：Amazon QLDB

刪除分類帳及其所有內容。此動作不可復原。

如果已啟用刪除保護，則必須先停用該功能才能刪除分類帳。您可以呼叫 `UpdateLedger` 操作將此參數設定為 `false` 以停用該功能。

請求語法

```
DELETE /ledgers/name HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

您要刪除的分類帳名稱。

長度限制：長度下限為 1。長度上限為 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
```

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

ResourceInUseException

目前無法修改指定的資源。

HTTP 狀態碼：409

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

ResourcePreconditionNotMetException

作業失敗，因為未事先滿足條件。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

DescribeJournalKinesisStream

服務：Amazon QLDB

傳回有關指定 Amazon QLDB 日誌資料流的詳細資訊。輸出包括 Amazon 資源名稱 (ARN)、串流名稱、目前狀態、建立時間和原始串流建立請求的參數。

此動作不會傳回任何過期的日誌串流。如需詳細資訊，請參閱 Amazon QLDB 開發人員指南中的終端機串流到期日。

請求語法

```
GET /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

分類帳的名稱。

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必要：是

streamId

要描述之 QLDB 日誌串流的 UUID (以 Base62 編碼的文字表示)。

長度約束：固定長度為 22。

模式：^[A-Za-z-0-9]+$

必要：是

請求主體

請求沒有請求主體。

回應語法

```
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"
   }
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

Stream

請求所傳回之 QLDB 日誌資料流的相關資訊。DescribeJournalS3Export

類型：JournalKinesisStreamDescription 物件

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

ResourcePreconditionNotMetException

作業失敗，因為未事先滿足條件。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

DescribeJournalS3Export

服務：Amazon QLDB

傳回分錄匯出工作的相關資訊，包括分類帳名稱、匯出識別碼、建立時間、目前狀態，以及原始匯出建立請求的參數。

此動作不會傳回任何過期的匯出工作。如需詳細資訊，請參閱 Amazon QLDB 開發人員指南中的匯出任務到期。

如果具有給定的導出作業`ExportId`不存在，則拋出`ResourceNotFoundException`。

如果具有給定的分類帳`Name`不存在，則拋出`ResourceNotFoundException`。

請求語法

```
GET /ledgers/name/journal-s3-exports/exportId HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

exportId

要說明之分錄匯出工作的 UUID (以 Base62 編碼的文字表示)。

長度約束：固定長度為 22。

模式：`^[A-Za-z-0-9]+$`

必要：是

name

分類帳的名稱。

長度限制：長度下限為 1。長度上限為 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必要：是

請求主體

請求沒有請求主體。

回應語法

```
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"
   }
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

ExportDescription

`DescribeJournalS3Export`請求所傳回之分錄匯出工作的相關資訊。

類型：JournalS3ExportDescription 物件

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的開發
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

DescribeLedger

服務：Amazon QLDB

傳回分類帳的相關資訊，包括其狀態、權限模式、靜態加密設定以及建立時間。

請求語法

```
GET /ledgers/name HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

您要說明之分類帳的名稱。

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必要：是

請求主體

請求沒有請求主體。

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

Arn

分類帳的 Amazon 資源名稱（ARN）。

類型：字串

長度限制：長度下限為 20。長度上限為 1600。

CreationDateTime

建立分類帳時的日期和時間 (以紀元時間格式表示)。（紀元時間格式是自世界標準時間 1970 年 1 月 1 日上午 12:00:00 以來經過的秒數。）

類型：Timestamp

DeletionProtection

指定是否要保護分類帳，不讓任何使用者刪除。如果在建立分類帳時未定義，則預設會啟用此功能 (`true`)。

如果已啟用刪除保護，則必須先停用該功能才能刪除分類帳。您可以呼叫 `UpdateLedger` 操作將此參數設定為 `false` 以停用該功能。

類型：布林值

EncryptionDescription

有關分類帳中靜態數據加密的信息。這包括當前狀態， AWS KMS 密鑰以及密鑰無法訪問的時間（在發生錯誤的情況下）。如果未定義此參數，則分類帳會使用 AWS 擁有的 KMS 金鑰進行加密。

類型：LedgerEncryptionDescription 物件

Name

分類帳的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

PermissionsMode

分類帳的權限模式。

類型：字串

有效值:ALLOW_ALL | STANDARD

State

分類帳的目前狀態。

類型：字串

有效值:CREATING | ACTIVE | DELETING | DELETED

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET

- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

ExportJournalToS3

服務：Amazon QLDB

將日期和時間範圍內的日誌內容從分類帳匯出到指定的 Amazon Simple Storage Service (Amazon S3) 儲存貯體。日誌匯出任務可以使用 Amazon Ion 格式的文字或二進位表示法或 JSON 行文字格式來寫入資料物件。

如果具有給定的分類帳Name不存在，則拋出ResourceNotFoundException。

如果具有給定的分類帳處Name於CREATING狀態，則拋出ResourcePreconditionNotMetException。

您可以針對每個分類帳，啟動最多兩個並行分錄匯出請求。超出此限制時，會擲回日誌匯出要求LimitExceededException。

請求語法

```
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"
   }
}
```

URI 請求參數

請求會使用下列 URI 參數。

name

分類帳的名稱。

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

ExclusiveEndTime

要匯出之分錄內容範圍的專用結束日期與時間。

ExclusiveEndTime 必須採用 ISO 8601 日期和時間格式，並以國際標準時間 (UTC) 表示。例如：2019-06-13T21:36:34Z。

ExclusiveEndTime必須小於或等於目前的 UTC 日期和時間。

類型：Timestamp

必要：是

InclusiveStartTime

要匯出之分錄內容範圍的開始日期與時間 (含)。

InclusiveStartTime 必須採用 ISO 8601 日期和時間格式，並以國際標準時間 (UTC) 表示。例如：2019-06-13T21:36:34Z。

必InclusiveStartTime須在之前ExclusiveEndTime。

如果您提供的InclusiveStartTime是分類帳之前的CreationDateTime，Amazon QLDB 將其預設為分類帳。CreationDateTime

類型：Timestamp

必要：是

OutputFormat

匯出分錄資料的輸出格式。日誌匯出任務可以使用 Amazon Ion 格式的文字或二進位表示法或 JSON 行文字格式來寫入資料物件。

預設：ION_TEXT

在 JSON 行格式中，匯出資料物件中的每個日誌區塊都是以換行符分隔的有效 JSON 物件。您可以使用此格式將 JSON 匯出與 Amazon Athena 等分析工具直接整合， AWS Glue 因為這些服務可以自動剖析以換行符分隔的 JSON。

類型：字串

有效值:ION_BINARY | ION_TEXT | JSON

必要：否

RoleArn

IAM 角色的 Amazon 資源名稱 (ARN)，可授與日誌匯出任務的 QLDB 許可，以執行以下作業：

- 將物件寫入您的 Amazon S3 儲存貯體。
- (選擇性) 在 AWS Key Management Service (AWS KMS) 中使用您的客戶管理金鑰，對匯出的資料進行伺服器端加密。

若要在請求日誌匯出時將角色傳遞給 QLDB，您必須擁有對 IAM 角色資源執行iam:PassRole動作的權限。所有分錄匯出請求都需要此功能。

類型：字串

長度限制：長度下限為 20。長度上限為 1600。

必要：是

S3ExportConfiguration

您的匯出請求之 Amazon S3 儲存貯體目的地的組態設定。

類型：S3ExportConfiguration 物件

必要：是

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
```

```
    "ExportId": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

ExportId

QLDB 指派給每個分錄匯出工作的 UUID (以 Base62 編碼的文字表示)。

要描述您的出口請求並檢查工作狀態，您可以使用撥ExportId打電話DescribeJournalS3Export。

類型：字串

長度約束：固定長度為 22。

模式：^[A-Za-z-0-9]+$

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

ResourcePreconditionNotMetException

作業失敗，因為未事先滿足條件。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面

- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetBlock

服務：Amazon QLDB

傳回日誌中指定位址的區塊物件。如果提供，還返回指定塊的證明進DigestTipAddress行驗證。

如需區塊中資料內容的相關資訊，請參閱 Amazon QLDB 開發人員指南中的日誌內容。

如果指定的分類帳不存在或處於DELETING狀態，則拋出ResourceNotFoundException。

如果指定的分類帳處於CREATING狀態，則拋出ResourcePreconditionNotMetException。

如果指定地址沒有塊存在，則拋出InvalidParameterException。

請求語法

```
POST /ledgers/name/block HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

URI 請求參數

請求會使用下列 URI 參數。

name

分類帳的名稱。

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

BlockAddress

您要請求的塊的位置。地址是具有兩個字段的 Amazon 離子結構：`strandId`和`sequenceNo`。

例如：`{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`。

類型：ValueHolder 物件

必要：是

DigestTipAddress

摘要涵蓋要求證明的最新區塊位置。地址是具有兩個字段的 Amazon 離子結構：`strandId`和`sequenceNo`。

例如：`{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:49}`。

類型：ValueHolder 物件

必要：否

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "Block": {
      "IonText": "string"
   },
   "Proof": {
      "IonText": "string"
   }
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

Block

Amazon 離子格式的塊數據對象。

類型：ValueHolder 物件

Proof

GetBlock請求返回的 Amazon 離子格式的證明對象。證明包含使用默克爾樹（從指定塊開始）重新計算指定摘要所需的哈希值列表。

類型：ValueHolder 物件

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

ResourcePreconditionNotMetException

作業失敗，因為條件未事先滿足。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK

- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetDigest

服務：Amazon QLDB

傳回分錄中最新確認區塊中分類帳的摘要。該響應包括 256 位哈希值和塊地址。

請求語法

```
POST /ledgers/name/digest HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

分類帳的名稱。

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "Digest": blob,
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

Digest

256 位哈希值表示由請求返回的摘要。`GetDigest`

類型：Base64 編碼的二進位資料物件

長度約束：固定長度為 32。

DigestTipAddress

您要求的摘要所涵蓋的最新區塊位置。地址是具有兩個字段的 Amazon 離子結構：`strandId`和`sequenceNo`。

類型：ValueHolder 物件

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

ResourcePreconditionNotMetException

作業失敗，因為未事先滿足條件。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面

- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetRevision

服務：Amazon QLDB

傳回指定文件 ID 和區塊位址的修訂資料物件。如DigestTipAddress果有提供，也會傳回指定修訂版本的證明以進行驗證。

請求語法

```
POST /ledgers/name/revision HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   },
   "DocumentId": "string"
}
```

URI 請求參數

請求會使用下列 URI 參數。

name

分類帳的名稱。

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

BlockAddress

要驗證的文件修訂版本的區塊位置。地址是具有兩個字段的 Amazon 離子結構：strandId和sequenceNo。

例如：{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}。

類型：ValueHolder 物件

必要：是

DigestTipAddress

摘要涵蓋要求證明的最新區塊位置。地址是具有兩個字段的 Amazon 離子結構：strandId和sequenceNo。

例如：{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:49}。

類型：ValueHolder 物件

必要：否

DocumentId

要驗證的文件的 UUID (以 Base62 編碼的文字表示)。

類型：字串

長度約束：固定長度為 22。

模式：^[A-Za-z-0-9]+$

必要：是

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "Proof": {
      "IonText": "string"
   },
   "Revision": {
      "IonText": "string"
   }
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

Proof

GetRevision請求返回的 Amazon 離子格式的證明對象。證明包含使用默克爾樹重新計算指定摘要所需的哈希值列表，從指定的文檔修訂版本開始。

類型：ValueHolder 物件

Revision

Amazon 離子格式的文檔修訂數據對象。

類型：ValueHolder 物件

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

ResourcePreconditionNotMetException

作業失敗，因為條件未事先滿足。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

ListJournalKinesisStreamsForLedger

服務：Amazon QLDB

傳回指定分類帳的所有 Amazon QLDB 日誌資料流。

此動作不會傳回任何過期的日誌串流。如需詳細資訊，請參閱 Amazon QLDB 開發人員指南中的終端機串流到期日。

此操作返回最大的`MaxResults`項目。它被分頁，以便您可以通過`ListJournalKinesisStreamsForLedger`多次調用來檢索所有項目。

請求語法

```
GET /ledgers/name/journal-kinesis-streams?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

分類帳的名稱。

長度限制：長度下限為 1。長度上限為 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必要：是

MaxResults

在單一`ListJournalKinesisStreamsForLedger`要求中傳回的結果數目上限。(傳回的實際結果數量可能會較少。)

有效範圍：最小值為 1。最大值為 100。

NextToken

一個分頁令牌，表示您要檢索結果的下一頁。如果您在先前`ListJournalKinesisStreamsForLedger`呼叫的回應`NextToken`中收到的值，您應該在此處使用該值作為輸入。

長度約束：最小長度為 4。長度上限為 1024。

模式：^[A-Za-z-0-9+/=]+$

請求主體

請求沒有請求主體。

回應語法

```
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"
      }
   ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

NextToken

- 如果NextToken為空，則表示結果的最後一頁已處理，並且沒有更多的結果要擷取。

- 如果不*NextToken*是空的，則可以使用更多結果。若要擷取結果的下一頁，請在後續`ListJournalKinesisStreamsForLedger`呼叫`NextToken`中使用的值。

  類型：字串

  長度約束：最小長度為 4。長度上限為 1024。

  模式：`^[A-Za-z-0-9+/=]+$`

Streams

目前與指定分類帳相關聯的 QLDB 分錄資料流。

類型：JournalKinesisStreamDescription 物件陣列

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

ResourcePreconditionNotMetException

作業失敗，因為未事先滿足條件。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET

- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

ListJournalS3Exports

服務：Amazon QLDB

針對與目 AWS 帳戶 前與區域相關聯的所有分類帳，傳回所有分錄匯出工單。

此操作返回最大的`MaxResults`項目，並分頁，以便您可以通過調用`ListJournalS3Exports`多次檢索所有項目。

此動作不會傳回任何過期的匯出工作。如需詳細資訊，請參閱 Amazon QLDB 開發人員指南中的匯出任務到期。

請求語法

```
GET /journal-s3-exports?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

MaxResults

在單一`ListJournalS3Exports`要求中傳回的結果數目上限。(傳回的實際結果數量可能會較少。)

有效範圍：最小值為 1。最大值為 100。

NextToken

一個分頁令牌，表示您要檢索結果的下一頁。如果您在先前`ListJournalS3Exports`呼叫的回應`NextToken`中收到的值，則應在此處使用該值作為輸入。

長度約束：最小長度為 4。長度上限為 1024。

模式：`^[A-Za-z-0-9+/=]+$`

請求主體

請求沒有請求主體。

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

JournalS3Exports

與目 AWS 帳戶 前與區域相關聯之所有分類帳的分錄匯出工單。

類型：JournalS3ExportDescription 物件陣列

NextToken

- 如果NextToken為空，則表示結果的最後一頁已處理，並且沒有更多的結果要擷取。
- 如果不*NextToken*是空的，則會有更多可用的結果。若要擷取結果的下一頁，請在後續ListJournalS3Exports呼叫NextToken中使用的值。

類型：字串

長度約束：最小長度為 4。長度上限為 1024。

模式：^[A-Za-z-0-9+/=]+$

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

ListJournalS3ExportsForLedger

服務：Amazon QLDB

傳回指定分類帳的所有分錄匯出工作。

此操作返回最大的`MaxResults`項目，並分頁，以便您可以通過調用`ListJournalS3ExportsForLedger`多次檢索所有項目。

此動作不會傳回任何過期的匯出工作。如需詳細資訊，請參閱 Amazon QLDB 開發人員指南中的匯出任務到期。

請求語法

```
GET /ledgers/name/journal-s3-exports?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

MaxResults

在單一`ListJournalS3ExportsForLedger`要求中傳回的結果數目上限。(傳回的實際結果數量可能會較少。)

有效範圍：最小值為 1。最大值為 100。

name

分類帳的名稱。

長度限制：長度下限為 1。長度上限為 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必要：是

NextToken

一個分頁令牌，表示您要檢索結果的下一頁。如果您在先前`ListJournalS3ExportsForLedger`呼叫的回應`NextToken`中收到的值，則應在此處使用該值作為輸入。

長度約束：最小長度為 4。長度上限為 1024。

模式：^[A-Za-z-0-9+/=]+$

請求主體

請求沒有請求主體。

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

JournalS3Exports

目前與指定分類帳相關聯的分錄匯出工作。

類型：JournalS3ExportDescription 物件陣列

NextToken

- 如果NextToken為空，則表示結果的最後一頁已處理，並且沒有更多的結果要擷取。
- 如果不*NextToken*是空的，則會有更多可用的結果。若要擷取結果的下一頁，請在後續ListJournalS3ExportsForLedger呼叫NextToken中使用的值。

類型：字串

長度約束：最小長度為 4。長度上限為 1024。

模式：^[A-Za-z-0-9+/=]+$

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

ListLedgers

服務：Amazon QLDB

傳回與目 AWS 帳戶 前與區域相關聯的所有分類帳。

此操作返回最大的`MaxResults`項目，並分頁，以便您可以通過調用`ListLedgers`多次檢索所有項目。

請求語法

```
GET /ledgers?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

MaxResults

在單一`ListLedgers`要求中傳回的結果數目上限。(傳回的實際結果數量可能會較少。)

有效範圍：最小值為 1。最大值為 100。

NextToken

一個分頁令牌，表示您要檢索結果的下一頁。如果您在先前`ListLedgers`呼叫的回應`NextToken`中收到的值，則應在此處使用該值作為輸入。

長度約束：最小長度為 4。長度上限為 1024。

模式：`^[A-Za-z-0-9+/=]+$`

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "Ledgers": [
```

```
      {
         "CreationDateTime": number,
         "Name": "string",
         "State": "string"
      }
   ],
   "NextToken": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

Ledgers

與目前與區域相關聯的 AWS 帳戶 分類帳。

類型：LedgerSummary 物件陣列

NextToken

一個分頁令牌，指示是否有更多可用的結果：

- 如果NextToken為空，則表示結果的最後一頁已處理，並且沒有更多的結果要擷取。
- 如果不*NextToken*是空的，則會有更多可用的結果。若要擷取結果的下一頁，請在後續ListLedgers呼叫NextToken中使用的值。

類型：字串

長度約束：最小長度為 4。長度上限為 1024。

模式：^[A-Za-z-0-9+/=]+$

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面

- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

ListTagsForResource

服務：Amazon QLDB

傳回指定 Amazon QLDB 資源的所有標籤。

請求語法

```
GET /tags/resourceArn HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

resourceArn

要列出標籤的 Amazon 資源名稱（ARN）。例如：

arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger

長度限制：長度下限為 20。長度上限為 1600。

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

Tags

目前與指定的 Amazon QLDB 資源相關聯的標籤。

類型：字串到字串映射

地圖項目：0 個項目的最小數目。項目數上限為 200。

索引鍵長度限制：長度下限為 1。長度上限為 128。

值長度限制：最小長度為 0。長度上限為 256。

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP

- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

StreamJournalToKinesis

服務：Amazon QLDB

為指定的 Amazon QLDB 分類帳建立日誌串流。串流會擷取認可至總類日誌的每個文件修訂版本，並將資料傳送至指定的 Amazon Kinesis Data Streams 資源。

請求語法

```
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"
   }
}
```

URI 請求參數

請求會使用下列 URI 參數。

name

分類帳的名稱。

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

ExclusiveEndTime

指定串流結束時間的專屬日期和時間。如果您未定義此參數，串流會無限期地執行，直到您取消它為止。

`ExclusiveEndTime` 必須採用 `ISO 8601` 日期和時間格式，並以國際標準時間 (UTC) 表示。例如：`2019-06-13T21:36:34Z`。

類型：Timestamp

必要：否

InclusiveStartTime

開始串流日誌資料的包含開始日期和時間。此參數必須是 `ISO 8601` 日期和時間格式，並以國際標準時間 (UTC) 表示。例如：`2019-06-13T21:36:34Z`。

`InclusiveStartTime` 不能在未來，且必須在 `ExclusiveEndTime` 之前。

如果您提供位於分類帳 `CreationDateTime` 之前的分類帳 `InclusiveStartTime`，QLDB 實際上會將其預設為分類帳的 `CreationDateTime`。

類型：Timestamp

必要：是

KinesisConfiguration

串流請求之 Kinesis Data Streams 目的地的組態設定。

類型：KinesisConfiguration 物件

必要：是

RoleArn

IAM 角色的 Amazon Resource Name (ARN)，可授予日誌串流的 QLDB 許可，以便將資料記錄寫入 Kinesis Data Streams 資源。

若要在請求日誌串流時將角色傳遞至 QLDB，則您必須擁有針對 IAM 角色資源執行 `iam:PassRole` 動作的許可。所有日誌串流請求都需有此許可。

類型：字串

長度限制：長度下限為 20。長度上限為 1600。

必要：是

StreamName

您要指派給 QLDB 日誌串流的名稱。使用者定義的名稱可協助識別和指示串流用途。

對於特定分類帳，您的串流名稱在其他作用中串流間必須是唯一的。串流名稱與分類帳名稱有相同的命名限制，如《Amazon QLDB 開發人員指南》中 Amazon QLDB 中的配額所定義。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必要：是

Tags

鍵值對作為標籤添加到您要創建的流。標籤鍵會區分大小寫。標籤值區分大小寫，並且可以為 null。

類型：字串到字串映射

地圖項目：0 個項目的最小數目。項目數上限為 200。

索引鍵長度限制：長度下限為 1。長度上限為 128。

值長度限制：最小長度為 0。長度上限為 256。

必要：否

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "StreamId": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

StreamId

QLDB 指派給每個 QLDB 日誌串流的 UUID (以 Base62 編碼的文字表示)。

類型：字串

長度約束：固定長度為 22。

模式：^[A-Za-z-0-9]+$

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

ResourcePreconditionNotMetException

作業失敗，因為未事先滿足條件。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK

- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

TagResource

服務：Amazon QLDB

將一個或多個標籤新增至指定的 Amazon QLDB 資源。

一個資源最多可以有 50 個標籤。如果您嘗試為資源建立超過 50 個標籤，您的要求會失敗並傳回錯誤。

請求語法

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

URI 請求參數

請求會使用下列 URI 參數。

resourceArn

Amazon 資源名稱（ARN）要添加標籤。例如：

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

長度限制：長度下限為 20。長度上限為 1600。

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

Tags

鍵值對作為標籤添加到指定的 QLDB 資源。標籤鍵會區分大小寫。如果您指定資源已存在的索引鍵，您的要求會失敗並傳回錯誤。標籤值區分大小寫，並且可以為 null。

類型：字串到字串映射

地圖項目：0 個項目的最小數目。項目數上限為 200。

索引鍵長度限制：長度下限為 1。長度上限為 128。

值長度限制：最小長度為 0。長度上限為 256。

必要：是

回應語法

```
HTTP/1.1 200
```

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++

- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

UntagResource

服務：Amazon QLDB

從指定的 Amazon QLDB 資源中移除一個或多個標籤。您最多可以指定 50 個要移除的標籤鍵。

請求語法

```
DELETE /tags/resourceArn?tagKeys=TagKeys HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

resourceArn

要從中刪除標籤的 Amazon 資源名稱（ARN）。例如：

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

長度限制：長度下限為 20。長度上限為 1600。

必要：是

TagKeys

要刪除的標籤鍵的列表。

陣列成員：項目數下限為 0。項目數上限為 200。

長度限制：長度下限為 1。長度上限為 128。

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
```

回應元素

如果動作成功，則服務會傳回具空 HTTP 內文的 HTTP 200 回應。

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

UpdateLedger

服務：Amazon QLDB

更新分類帳上的屬性。

請求語法

```
PATCH /ledgers/name HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string"
}
```

URI 請求參數

請求會使用下列 URI 參數。

name

分類帳的名稱。

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

DeletionProtection

指定是否要保護分類帳，不讓任何使用者刪除。如果在建立分類帳時未定義，則預設會啟用此功能 (true)。

如果已啟用刪除保護，則必須先停用該功能才能刪除分類帳。您可以呼叫 UpdateLedger 操作將此參數設定為 false 以停用該功能。

類型：布林值

必要：否

KmsKey

AWS Key Management Service (AWS KMS) 中的密鑰用於分類帳中靜態數據的加密。如需詳細資訊，請參閱《Amazon QLDB 開發人員指南》中的靜態加密。

使用以下其中一個選項來指定此參數：

- AWS_OWNED_KMS_KEY：使用代表您擁有和管理 AWS 的 AWS KMS 密鑰。
- 未定義：不變更分類帳的 KMS 金鑰。
- 有效的對稱客戶受管 KMS 金鑰：使用在您帳戶中建立、擁有和管理的指定對稱加密 KMS 金鑰。

  Amazon QLDB 不支援非對稱金鑰。如需詳細資訊，請參閱 AWS Key Management Service 開發人員指南中的使用對稱和非對稱金鑰。

若要指定客戶受管 KMS 金鑰，您可使用其金鑰 ID、Amazon Resource Name (ARN)、別名名稱或別名 ARN。使用別名名稱時，請加上 "alias/" 字首。若要指定不同的索引鍵 AWS 帳戶，您必須使用金鑰 ARN 或別名 ARN。

例如：

- 金鑰 ID：1234abcd-12ab-34cd-56ef-1234567890ab
- 金鑰 ARN：arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
- 別名名稱：alias/ExampleAlias
- 別名 ARN：arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias

如需詳細資訊，請參閱 AWS Key Management Service 開發人員指南中的金鑰識別碼 (KeyId)。

類型：字串

長度限制：最大長度為 1600。

必要：否

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

Arn

分類帳的 Amazon 資源名稱（ARN）。

類型：字串

長度限制：長度下限為 20。長度上限為 1600。

CreationDateTime

建立分類帳時的日期和時間 (以紀元時間格式表示)。（紀元時間格式是自世界標準時間 1970 年 1 月 1 日上午 12:00:00 以來經過的秒數。）

類型：Timestamp

DeletionProtection

指定是否要保護分類帳，不讓任何使用者刪除。如果在建立分類帳時未定義，則預設會啟用此功能 (`true`)。

如果已啟用刪除保護，則必須先停用該功能才能刪除分類帳。您可以呼叫 `UpdateLedger` 操作將此參數設定為 `false` 以停用該功能。

類型：布林值

EncryptionDescription

有關分類帳中靜態數據加密的信息。這包括當前狀態， AWS KMS 密鑰以及密鑰無法訪問的時間（在發生錯誤的情況下）。

類型：LedgerEncryptionDescription 物件

Name

分類帳的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

State

分類帳的目前狀態。

類型：字串

有效值:CREATING | ACTIVE | DELETING | DELETED

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

UpdateLedgerPermissionsMode

服務：Amazon QLDB

更新分類帳的權限模式。

> **Important**
>
> 切換到STANDARD許可模式之前，您必須先建立所有必要的 IAM 政策和表格標記，以避免對使用者造成干擾。若要進一步了解，請參閱 Amazon QLDB 開發人員指南中的遷移至標準許可模式。

請求語法

```
PATCH /ledgers/name/permissions-mode HTTP/1.1
Content-type: application/json

{
   "PermissionsMode": "string"
}
```

URI 請求參數

請求會使用下列 URI 參數。

name

分類帳的名稱。

長度限制：長度下限為 1。長度上限為 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

PermissionsMode

指定給分類帳的權限模式。此參數可以有下列其中一個值：

- ALLOW_ALL：一種舊版許可模式，可使用 API 層級精細程度啟用對分類帳的存取控制。

  此模式允許具有指定分類帳 SendCommand API 許可的使用者 (因此為 ALLOW_ALL)，在此分類帳中的任何資料表上執行所有 PartiQL 命令。此模式會忽略您為分類帳建立之任何資料表層級或命令層級 IAM 許可政策。

- STANDARD：(建議) 一種許可模式，可使用更細的層級啟用對分類帳、資料表和 PartiQL 命令的存取控制。

  根據預設，此模式會拒絕所有使用者在此分類帳中任何資料表上執行任何 PartiQL 命令的請求。若要允許執行 PartiQL 命令，除了分類帳的 SendCommand API 許可之外，您還必須為特定資料表資源和 PartiQL 動作建立 IAM 許可政策。如需相關資訊，請參閱《Amazon QLDB 開發人員指南》中的開始使用標準許可模式。

> **Note**
>
> 強烈建議您使用 STANDARD 許可模式來最大化分類帳資料的安全性。

類型：字串

有效值:ALLOW_ALL | STANDARD

必要：是

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "Name": "string",
   "PermissionsMode": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

Arn

分類帳的 Amazon 資源名稱（ARN）。

類型：字串

長度限制：長度下限為 20。長度上限為 1600。

Name

分類帳的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

PermissionsMode

分類帳的目前權限模式。

類型：字串

有效值:ALLOW_ALL | STANDARD

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

InvalidParameterException

請求中的一個或多個參數無效。

HTTP 狀態碼：400

ResourceNotFoundException

指定的資源不存在。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

Amazon QLDB 會議

Amazon QLDB Session 支援下列動作：

- SendCommand

SendCommand

服務：Amazon QLDB Session

將命令傳送至 Amazon QLDB 分類帳。

> **Note**
>
> 建議您不要直接與此 API 互動，而是使用 QLDB 驅動程式或 QLDB 命令介面在分類帳上執行資料交易。
>
> - 如果您使用的是 AWS SDK，請使用 QLDB 驅動程式。驅動程式在此 QLDB 工作階段資料 API 上方提供了一個高層級的抽象層，並為您管理SendCommand作業。如需相關資訊和支援的程式設計語言清單，請參閱 Amazon QLDB 開發人員指南中的開始使用驅動程式。
> - 如果您正在使用 AWS Command Line Interface（AWS CLI），請使用 QLDB 外殼。殼層是使用 QLDB 驅動程式與總帳互動的命令列介面。如需相關資訊，請參閱使用 QLDB 殼層存取 Amazon QL DB。

請求語法

```
{
   "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": {
   }
}
```

請求參數

如需所有動作的一般參數資訊，請參閱《Common Parameters》。

請求接受採用 JSON 格式的下列資料。

AbortTransaction

中止目前交易的命令。

類型：AbortTransactionRequest 物件

必要：否

CommitTransaction

命令來提交指定的事務。

類型：CommitTransactionRequest 物件

必要：否

EndSession

結束目前階段作業的指令。

類型：EndSessionRequest 物件

必要：否

ExecuteStatement

在指定的事務中執行語句的命令。

類型：ExecuteStatementRequest 物件

必要：否

FetchPage

獲取頁面的命令。

類型：FetchPageRequest 物件

必要：否

SessionToken

指定目前命令的工作階段權杖。會話令牌在會話的整個生命週期中是恆定的。

若要取得工作階段權杖，請執行StartSession命令。這SessionToken對於在目前作業階段期間發出的每個後續指令都需要這樣做。

類型：字串

長度約束：最小長度為 4。長度上限為 1024。

模式：^[A-Za-z-0-9+/=]+$

必要：否

StartSession

啟動新工作階段的命令。會話令牌作為響應的一部分獲得。

類型：StartSessionRequest 物件

必要：否

StartTransaction

啟動新交易的命令。

類型：StartTransactionRequest 物件

必要：否

回應語法

```
{
   "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"
   }
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

AbortTransaction

包含已中止交易的詳細資訊。

類型：AbortTransactionResult 物件

CommitTransaction

包含已認可交易的詳細資訊。

類型：CommitTransactionResult 物件

EndSession

包含已結束工作階段的詳細資訊。

類型：EndSessionResult 物件

ExecuteStatement

包含已執行陳述式的詳細資訊。

類型：ExecuteStatementResult 物件

FetchPage

包含擷取頁面的詳細資訊。

類型：FetchPageResult 物件

StartSession

包含已啟動工作階段的詳細資料，其中包含工作階段權杖。這SessionToken對於在目前作業階段期間發出的每個後續指令都需要這樣做。

類型：StartSessionResult 物件

StartTransaction

包含已啟動交易的詳細資訊。

類型：StartTransactionResult 物件

錯誤

如需所有動作常見錯誤的資訊，請參閱常見錯誤。

BadRequestException

如果請求格式錯誤或包含錯誤，例如無效的參數值或缺少必要參數，則傳回。

HTTP 狀態碼：400

CapacityExceededException

當請求超過分類帳的處理能力時傳回。

HTTP 狀態碼：400

InvalidSessionException

如果會話因為超時或過期而不存在，則返回。

HTTP 狀態碼：400

LimitExceededException

如果超過使用中工作階段數目等資源限制，則傳回。

HTTP 狀態碼：400

OccConflictException

當交易因樂觀並行控制 (OCC) 的驗證階段失敗而無法寫入日誌時傳回。

HTTP 狀態碼：400

RateExceededException

當要求的速率超過允許的輸送量時傳回。

HTTP 狀態碼：400

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的開發
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

資料類型

目前支援下列資料類 QLDB：

- JournalKinesisStreamDescription
- JournalS3ExportDescription
- KinesisConfiguration

- LedgerEncryptionDescription
- LedgerSummary
- S3EncryptionConfiguration
- S3ExportConfiguration
- ValueHolder

亞馬遜 QLDB 會話支援下列資料類型：

- 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

服務：Amazon QLDB

Amazon QLDB 日誌串流的相關資訊，包括 Amazon 資源名稱 (ARN)、串流名稱、建立時間、目前狀態以及原始串流建立請求的參數。

目錄

KinesisConfiguration

QLDB 日誌串流之 Amazon Kinesis Data Streams 目的地的組態設定。

類型：KinesisConfiguration 物件

必要：是

LedgerName

分類帳的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必要：是

RoleArn

IAM 角色的 Amazon Resource Name (ARN)，可授予日誌串流的 QLDB 許可，以便將資料記錄寫入 Kinesis Data Streams 資源。

類型：字串

長度限制：長度下限為 20。長度上限為 1600。

必要：是

Status

QLDB 日誌資料流的目前狀態。

類型：字串

有效值:ACTIVE | COMPLETED | CANCELED | FAILED | IMPAIRED

必要：是

StreamId

QLDB 日誌串流的 UUID (以 62 基編碼文字表示)。

類型：字串

長度約束：固定長度為 22。

模式：^[A-Za-z-0-9]+$

必要：是

StreamName

使用者定義的 QLDB 日誌資料流名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必要：是

Arn

QLDB 日誌串流的 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 20。長度上限為 1600。

必要：否

CreationTime

建立 QLDB 日誌串流時的日期和時間 (以紀元時間格式表示)。 (紀元時間格式是自世界標準時間 1970 年 1 月 1 日上午 12:00:00 以來經過的秒數。)

類型：Timestamp

必要：否

ErrorCause

說明串流狀態為IMPAIRED或的原因的錯誤訊息FAILED。這不適用於具有其他狀態值的串流。

類型：字串

有效值:KINESIS_STREAM_NOT_FOUND | IAM_PERMISSION_REVOKED

必要：否

ExclusiveEndTime

指定串流結束時間的專屬日期和時間。如果此參數未定義，則串流會無限期執行，直到您取消為止。

類型：Timestamp

必要：否

InclusiveStartTime

開始串流日誌資料的包含開始日期和時間。

類型：Timestamp

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发
- AWS 適用於紅寶石 V3 的 SDK

JournalS3ExportDescription

服務：Amazon QLDB

分錄匯出工作的相關資訊，包括分類帳名稱、匯出識別碼、建立時間、目前狀態，以及原始匯出建立請求的參數。

目錄

ExclusiveEndTime

在原始匯出請求中指定之分錄內容範圍的專用結束日期與時間。

類型：Timestamp

必要：是

ExportCreationTime

建立匯出工作時的日期和時間 (以紀元時間格式表示)。 (紀元時間格式是自世界標準時間 1970 年 1 月 1 日上午 12:00:00 以來經過的秒數。)

類型：Timestamp

必要：是

ExportId

分錄匯出工作的 UUID (以 Base62 編碼的文字表示)。

類型：字串

長度約束：固定長度為 22。

模式：`^[A-Za-z-0-9]+$`

必要：是

InclusiveStartTime

原始匯出請求中指定之分錄內容範圍的開始日期與時間 (含)。

類型：Timestamp

必要：是

LedgerName

分類帳的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必要：是

RoleArn

IAM 角色的 Amazon 資源名稱 (ARN)，可授與日誌匯出任務的 QLDB 許可，以執行以下作業：

- 將物件寫入您的亞馬遜簡單儲存服務 (Amazon S3) 儲存貯體。
- (選擇性) 在 AWS Key Management Service (AWS KMS) 中使用您的客戶管理金鑰，對匯出的資料進行伺服器端加密。

類型：字串

長度限制：長度下限為 20。長度上限為 1600。

必要：是

S3ExportConfiguration

日誌匯出任務寫入日誌內容的 Amazon 簡單儲存貯體服務 (Amazon S3) 儲存貯體位置。

類型：S3ExportConfiguration 物件

必要：是

Status

分錄匯出工作的目前狀態。

類型：字串

有效值:`IN_PROGRESS | COMPLETED | CANCELLED`

必要：是

OutputFormat

匯出分錄資料的輸出格式。

類型：字串

有效值:`ION_BINARY | ION_TEXT | JSON`

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

KinesisConfiguration

服務：Amazon QLDB

Amazon QLDB 日誌串流的 Amazon Kinesis Data Streams 目的地組態設定。

目錄

StreamArn

Kinesis Data Streams 資源的 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 20。長度上限為 1600。

必要：是

AggregationEnabled

讓 QLDB 在單一 Kinesis Data Streams 記錄中發佈多筆資料記錄，增加每個 API 呼叫傳送的記錄數目。

預設：`True`

> **Important**
>
> 記錄彙總對處理記錄具有重要影響，並且需要在串流取用者中取消彙總。如需進一步了解，請參閱《Amazon Kinesis Data Streams 開發人員指南》中的 KPL 重要概念和取用者取消彙總。

類型：布林值

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件

- AWS 適用於紅寶石 V3 的 SDK

LedgerEncryptionDescription

服務：Amazon QLDB

Amazon QLDB 分類帳中靜態資料加密的相關資訊。這包括當前狀態，key in AWS Key Management Service （AWS KMS）以及密鑰無法訪問時（在出現錯誤的情況下）。

如需詳細資訊，請參閱《Amazon QLDB 開發人員指南》中的靜態加密。

目錄

EncryptionStatus

總帳的目前靜態加密狀態。這可以是下列其中一個值：

- ENABLED：使用指定的密鑰完全啟用加密。
- UPDATING：分類帳正在主動處理指定的索引鍵變更。

  QLDB 中的關鍵變更是非同步的。在處理金鑰變更時，可完全存取分類帳，而不會造成任何效能影響。更新金鑰所需的時間長度視分類帳大小而有所不同。
- KMS_KEY_INACCESSIBLE：無法存取指定的客戶管理 KMS 金鑰，且分類帳會受損。金鑰已停用或刪除，或是金鑰的授權遭到撤銷。當分類帳受損時，它無法訪問，也不接受任何讀取或寫入請求。

  在您還原金鑰的授權或重新啟用已停用的金鑰之後，受損的分類帳會自動返回作用中狀態。不過，刪除客戶受管 KMS 金鑰是無法復原的。刪除金鑰後，您將無法再存取受該金鑰保護的分類帳，而且資料永久無法復原。

類型：字串

有效值:ENABLED | UPDATING | KMS_KEY_INACCESSIBLE

必要：是

KmsKeyArn

客戶受管 KMS 金鑰的 Amazon 資源名稱 (ARN)，分類帳用於靜態加密。如果未定義此參數，則分類帳會使用 AWS 擁有的 KMS 金鑰進行加密。將分類帳的加密組態更新為 AWS 擁有的 KMS 金鑰AWS_OWNED_KMS_KEY時，它會顯示。

類型：字串

長度限制：長度下限為 20。長度上限為 1600。

必要：是

InaccessibleKmsKeyDateTime

在發生錯誤的情況下， AWS KMS 金鑰首次無法存取時，以紀元時間格式顯示的日期和時間。（紀元時間格式是世界標準時間 1970 年 1 月 1 日上午 12:00:00 以來經過的秒數。）

如果 AWS KMS 金鑰是可存取的，則此參數未定義。

類型：Timestamp

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

LedgerSummary

服務：Amazon QLDB

分類帳的相關資訊，包括其名稱、州別及建立時間。

目錄

CreationDateTime

建立分類帳時的日期和時間 (以紀元時間格式表示)。（紀元時間格式是自世界標準時間 1970 年 1 月 1 日上午 12:00:00 以來經過的秒數。）

類型：Timestamp

必要：否

Name

分類帳的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必要：否

State

分類帳的目前狀態。

類型：字串

有效值:`CREATING | ACTIVE | DELETING | DELETED`

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的開發

- AWS 適用於紅寶石 V3 的 SDK

S3EncryptionConfiguration

服務：Amazon QLDB

日誌匯出任務用於在 Amazon Simple Storage Service (Amazon S3) 儲存貯體中寫入資料的加密設定。

目錄

ObjectEncryptionType

Amazon S3 物件加密類型。

若要進一步了解 Amazon S3 中的伺服器端加密選項，請參閱 Amazon S3 開發人員指南中的使用伺服器端加密保護資料。

類型：字串

有效值:`SSE_KMS | SSE_S3 | NO_ENCRYPTION`

必要：是

KmsKeyArn

() 中對稱加密金鑰的 Amazon 資源名稱 AWS Key Management Service (AWS KMS ARN)。Amazon S3 不支援非對稱 KMS 金鑰。

`KmsKeyArn`如果您指定`SSE_KMS`為，則必須提供`ObjectEncryptionType`。

`KmsKeyArn`如果您指定`SSE_S3`為，則不需要`ObjectEncryptionType`。

類型：字串

長度限制：長度下限為 20。長度上限為 1600。

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件

- AWS 適用於紅寶石 V3 的 SDK

S3ExportConfiguration

服務：Amazon QLDB

日誌匯出任務寫入日誌內容的 Amazon 簡單儲存貯體服務 (Amazon S3) 儲存貯體位置。

目錄

Bucket

日誌匯出任務用來寫入日誌內容的 Amazon S3 儲存貯體名稱。

儲存貯體名稱必須符合 Amazon S3 儲存貯體命名慣例。如需詳細資訊，請參閱 Amazon S3 開發人員指南中的儲存貯體限制和限制。

類型：字串

長度限制：長度下限為 3。長度上限為 255。

模式：`^[A-Za-z-0-9-_.]+$`

必要：是

EncryptionConfiguration

日誌匯出任務用於在 Amazon S3 儲存貯體中寫入資料的加密設定。

類型：S3EncryptionConfiguration 物件

必要：是

Prefix

日誌匯出任務在其中寫入日誌內容的 Amazon S3 儲存貯體的前置詞。

前綴必須符合 Amazon S3 金鑰命名規則和限制。如需詳細資訊，請參閱 Amazon S3 開發人員指南中的物件金鑰和中繼資料。

以下是有效`Prefix`值的範例：

- `JournalExports-ForMyLedger/Testing/`
- `JournalExports`
- `My:Tests/`

類型：字串

長度限制：長度下限為 0。長度上限為 128。

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发
- AWS 適用於紅寶石 V3 的 SDK

ValueHolder

服務：Amazon QLDB

可以包含多種編碼格式之值的結構。

目錄

IonText

結構中包含的 Amazon 離子明文值。`ValueHolder`

類型：字串

長度限制：長度下限為 1。最大長度

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

Amazon QLDB 課程

Amazon QLDB 會話支援下列資料類型：

- AbortTransactionRequest
- AbortTransactionResult
- CommitTransactionRequest
- CommitTransactionResult
- EndSessionRequest
- EndSessionResult
- ExecuteStatementRequest
- ExecuteStatementResult

- FetchPageRequest
- FetchPageResult
- IOUsage
- Page
- StartSessionRequest
- StartSessionResult
- StartTransactionRequest
- StartTransactionResult
- TimingInformation
- ValueHolder

AbortTransactionRequest

服務：Amazon QLDB Session

包含要中止之交易的詳細資訊。

目錄

此異常結構的成員與前後關聯相關。

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发
- AWS 適用於紅寶石 V3 的 SDK

AbortTransactionResult

服務：Amazon QLDB Session

包含已中止交易的詳細資訊。

目錄

TimingInformation

包含指令的伺服器端效能資訊。

類型：TimingInformation 物件

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

CommitTransactionRequest

服務：Amazon QLDB Session

包含要提交之交易的詳細資訊。

目錄

CommitDigest

指定要認可之交易的提交摘要。對於每個活動事務，提交摘要必須傳遞。如果客戶端上計算的摘要與 QLDB 計算的摘要不匹配，QLDB 驗證`CommitDigest`和拒絕提交錯誤。

該`CommitDigest`參數的目的是確保 QLDB 認可交易當且僅當服務器已經處理客戶端發送的確切語句集，以相同的順序，客戶端發送它們，並且沒有重複。

類型：Base64 編碼的二進位資料物件

必要：是

TransactionId

指定要認可之交易的交易 ID。

類型：字串

長度約束：固定長度為 22。

模式：`^[A-Za-z-0-9]+$`

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

CommitTransactionResult

服務：Amazon QLDB Session

包含已認可交易的詳細資訊。

目錄

CommitDigest

已提交交易的提交摘要。

類型：Base64 編碼的二進位資料物件

必要：否

ConsumedIOs

包含已使用之 I/O 要求數目的測量結果。

類型：IOUsage 物件

必要：否

TimingInformation

包含命令的伺服器端效能資訊。

類型：TimingInformation 物件

必要：否

TransactionId

已認可交易的交易 ID。

類型：字串

長度約束：固定長度為 22。

模式：`^[A-Za-z-0-9]+$`

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

EndSessionRequest

服務：Amazon QLDB Session

指定結束工作階段的請求。

目錄

此異常結構的成員與前後關聯相關。

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发
- AWS 適用於紅寶石 V3 的 SDK

EndSessionResult

服務：Amazon QLDB Session

包含已結束工作階段的詳細資訊。

目錄

TimingInformation

包含指令的伺服器端效能資訊。

類型：TimingInformation 物件

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

ExecuteStatementRequest

服務：Amazon QLDB Session

指定要執行陳述式的要求。

目錄

Statement

指定要求的陳述式。

類型：字串

長度限制：長度下限為 1。最大長度為 10 萬。

必要：是

TransactionId

指定要求的交易識別碼。

類型：字串

長度約束：固定長度為 22。

模式：`^[A-Za-z-0-9]+$`

必要：是

Parameters

指定要求中參數化陳述式的參數。

類型：ValueHolder 物件陣列

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件

- AWS 適用於紅寶石 V3 的 SDK

ExecuteStatementResult

服務：Amazon QLDB Session

包含已執行陳述式的詳細資訊。

目錄

ConsumedIOs

包含已使用之 I/O 要求數目的測量結果。

類型：IOUsage 物件

必要：否

FirstPage

包含第一個擷取頁面的詳細資訊。

類型：Page 物件

必要：否

TimingInformation

包含指令的伺服器端效能資訊。

類型：TimingInformation 物件

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

FetchPageRequest

服務：Amazon QLDB Session

指定要擷取的頁面的詳細資訊。

目錄

NextPageToken

指定要擷取之頁面的下一個頁面標記。

類型：字串

長度約束：最小長度為 4。長度上限為 1024。

模式：`^[A-Za-z-0-9+/=]+$`

必要：是

TransactionId

指定要擷取之頁面的交易 ID。

類型：字串

長度約束：固定長度為 22。

模式：`^[A-Za-z-0-9]+$`

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

FetchPageResult

服務：Amazon QLDB Session

包含擷取的頁面。

目錄

ConsumedIOs

包含已使用之 I/O 要求數目的測量結果。

類型：IOUsage 物件

必要：否

Page

包含擷取頁面的詳細資訊。

類型：Page 物件

必要：否

TimingInformation

包含指令的伺服器端效能資訊。

類型：TimingInformation 物件

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的開發
- AWS 適用於紅寶石 V3 的 SDK

IOUsage

服務：Amazon QLDB Session

包含已呼叫之命令的 I/O 使用狀況測量結果。

目錄

ReadIOs

命令發出的讀取 I/O 要求數目。

類型：Long

必要：否

WriteIOs

命令發出的寫入 I/O 要求數目。

類型：Long

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的開發
- AWS 適用於紅寶石 V3 的 SDK

Page

服務：Amazon QLDB Session

包含擷取頁面的詳細資訊。

目錄

NextPageToken

下一頁的權杖。

類型：字串

長度約束：最小長度為 4。長度上限為 1024。

模式：`^[A-Za-z-0-9+/=]+$`

必要：否

Values

包含多種編碼格式值的結構。

類型：ValueHolder 物件陣列

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

StartSessionRequest

服務：Amazon QLDB Session

指定啟動新工作階段的請求。

目錄

LedgerName

要針對其啟動新階段作業的分類帳名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

StartSessionResult

服務：Amazon QLDB Session

包含已啟動階段作業的詳細資訊。

目錄

SessionToken

已啟動階段作業的工作階段權杖。這SessionToken對於在目前作業階段期間發出的每個後續指令都需要這樣做。

類型：字串

長度約束：最小長度為 4。長度上限為 1024。

模式：^[A-Za-z-0-9+/=]+$

必要：否

TimingInformation

包含指令的伺服器端效能資訊。

類型：TimingInformation 物件

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

StartTransactionRequest

服務：Amazon QLDB Session

指定啟動交易的請求。

目錄

此異常結構的成員與前後關聯相關。

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发
- AWS 適用於紅寶石 V3 的 SDK

StartTransactionResult

服務：Amazon QLDB Session

包含已啟動交易的詳細資訊。

目錄

TimingInformation

包含指令的伺服器端效能資訊。

類型：TimingInformation 物件

必要：否

TransactionId

已啟動交易的交易 ID。

類型：字串

長度約束：固定長度為 22。

模式：`^[A-Za-z-0-9]+$`

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

TimingInformation

服務：Amazon QLDB Session

包含指令的伺服器端效能資訊。Amazon QLDB 會擷取接收請求到傳送對應回應的時間之間的計時資訊。

目錄

ProcessingTimeMilliseconds

QLDB 花在處理命令的時間量度，以毫秒為單位。

類型：Long

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

ValueHolder

服務：Amazon QLDB Session

可以包含多種編碼格式之值的結構。

目錄

IonBinary

`ValueHolder`結構中包含的 Amazon 離子二進制值。

類型：Base64 編碼的二進位資料物件

長度限制：長度下限為 1。最大長度為 131072。

必要：否

IonText

結構中包含的 Amazon 離子明文值。`ValueHolder`

類型：字串

長度限制：長度下限為 1。最大長度。

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

常見錯誤

本部分列出所有 AWS 服務 API 動作的常見錯誤。如需此服務之 API 動作的特定錯誤，請參閱該 API 動作的主題。

AccessDeniedException

您沒有足夠存取權可執行此動作。

HTTP 狀態碼：400

IncompleteSignature

請求簽署不符合 AWS 標準。

HTTP 狀態碼：400

InternalFailure

由於不明的錯誤、例外狀況或故障，處理請求失敗。

HTTP 狀態碼：500

InvalidAction

請求的動作或操作無效。確認已正確輸入動作。

HTTP 狀態碼：400

InvalidClientTokenId

提供的 X.509 憑證或 AWS 存取金鑰 ID 不存在於我們的記錄中。

HTTP 狀態碼：403

NotAuthorized

您沒有執行此動作的許可。

HTTP 狀態碼：400

OptInRequired

AWS 存取金鑰 ID 需要訂閱服務。

HTTP 狀態碼：403

RequestExpired

請求送達服務已超過戳印日期於請求上之後的 15 分鐘，或者已超過請求過期日期之後的 15 分鐘 (例如預先簽章的 URL)，或者請求上的日期戳印在未來將超過 15 分鐘。

HTTP 狀態碼：400

ServiceUnavailable

由於伺服器暫時故障，請求失敗。

HTTP 狀態碼：503

ThrottlingException

由於請求調節，因此請求遭到拒絕。

HTTP 狀態碼：400

ValidationError

輸入不符合 AWS 服務規定的限制。

HTTP 狀態碼：400

常見參數

以下清單內含所有動作用來簽署 Signature 第 4 版請求的參數以及查詢字串。任何專屬於特定動作的參數則列於該動作的主題中。如需簽名版本 4 的詳細資訊，請參閱 IAM 使用者指南中的簽署AWS API 請求。

Action

要執行的動作。

類型：字串

必要：是

Version

編寫請求所憑藉的 API 版本，以 YYYY-MM-DD 格式表示。

類型：字串

必要：是

X-Amz-Algorithm

建立請求簽章時所使用的雜湊演算法。

條件：當您在查詢字串中而非 HTTP 授權標頭中納入驗證資訊時，應指定此參數。

類型：字串

有效值: AWS4-HMAC-SHA256

必要：有條件

X-Amz-Credential

憑證範圍值，此為一個字串，其中包含您的存取金鑰、日期、您的目標區域、您請求的服務，以及終止字串 (“aws4_request”)。值以下列格式表示：access_key/YYYYMMDD/region/service/aws4_request。

如需詳細資訊，請參閱 IAM 使用者指南中的建立已簽署的AWS API 請求。

條件：當您在查詢字串中而非 HTTP 授權標頭中納入驗證資訊時，應指定此參數。

類型：字串

必要：有條件

X-Amz-Date

用來建立簽署的日期。格式必須是 ISO 8601 基本格式 (YYYYMMDD'T'HHMMSS'Z')。例如，以下日期時間是有效的 X-Amz-Date 值：20120325T120000Z

條件： 對所有請求而言，X-Amz-Date 皆為選用，可用來覆寫用於簽署請求的日期。如果規定日期標頭採用 ISO 8601 基本格式，則不需要 X-Amz-Date。當使用 X-Amz-Date 時，其一律會覆寫日期標頭的值。如需詳細資訊，請參閱 IAM 使用者指南中的 AWSAPI 請求簽名元素。

類型：字串

必要：有條件

X-Amz-Security-Token

透過呼叫AWS Security Token Service (AWS STS) 所取得的臨時安全字符。如需支援 Security Token 的臨時安全憑證的服務清單AWS STS，請AWS 服務前往 IAM 使用者指南中的《可搭配 IAM 運作》。

條件：如果您使用 Security Token 的臨時安全憑證AWS STS，則必須納入安全字符。

類型：字串

必要：有條件

X-Amz-Signature

指定從要簽署的字串和衍生的簽署金鑰中計算出的十六進位編碼簽章。

條件：當您在查詢字串中而非 HTTP 授權標頭中納入驗證資訊時，應指定此參數。

類型：字串

必要：有條件

X-Amz-SignedHeaders

指定納入作為標準請求一部分的所有 HTTP 標頭。如需有關指定已簽署標頭的詳細資訊，請參閱 IAM 使用者指南中的建立已簽署AWS API 請求。

條件：當您在查詢字串中而非 HTTP 授權標頭中納入驗證資訊時，應指定此參數。

類型：字串

必要：有條件

亞馬遜 QLDB 中的配額和限制

本節說明 Amazon QLDB 中的目前配額，也稱為限制。

主題

預設配額

QLDB 具有下列預設配額，如同 Amazon QLDB 端點和中的配額中也列出的預設配額AWS 一般參考。這些配額是AWS 帳戶每個區域的配額。若要請求增加您在區域中的帳戶配額，您可以使用 Service Quotas 主控台。

請登入，AWS Management Console並開啟 Service Quotas 主控台，網址為 https://console.aws.amazon.com/servicequotas/。

資源	預設配額
在目前區域中，您可以在此科目建立的有效分類帳數目上限	5
每個分類帳可匯出至 Amazon S3 的有效日誌數目上限	2
每個分類帳至 Kinesis 資料串流的作用中日誌串流數目上限	5

固定配額

除了預設配額之外，QLDB 還具有下列每個分類帳的固定配額。使用 Service Quotas 無法增加配額：

資源	固定配額
並行作用中階段作業數目	1500
作用中表格數目	20
表格總數 (作用中與非作用中) **Note** 在 QLDB 中，刪除的表格會被視為非作用中，並計入此總配額。	40
每份資料表的索引數目	5
交易中的文件數	40
要在交易中編輯的修訂數	1
文件大小 (以IonBinary 格式編碼)	128 KB
陳述式參數大小 (IonBinary 格式)	128 KB
陳述式參數大小 (IonText格式)	1 MB
陳述式字串長度	100,000 個字元
交易規模	4 MB
交易逾時	30 秒
已完成分錄匯出工作的到期期	7 天
終端機日誌串流的到期期	7 天

分類帳配額

若要請求增加區域中的帳戶分類帳配額，您可以使用 Service Quotas 主控台。

開啟 Service Quotas 主控台，網址為 https://console.aws.amazon.com/servicequotas/。

某些 QLDB 使用案例需要根據業務成長，AWS 帳戶每個區域的分類帳數量不斷擴大。例如，您可能需要建立專用分類帳，以隔離客戶或資料。在這種情況下，請考慮利用多帳戶架構來處理 QLDB 配額。如需詳細資訊，請參閱AWS白皮書 SaaS 租戶隔離策略中的帳戶孤立隔離。

文件大小

以IonBinary格式編碼的文件大小上限為 128 KB。我們無法為IonText格式的文檔大小提供確切的限制，因為從文本到二進制的轉換會根據每個文檔的結構而有很大差異。QLDB 支援開放內容的文件，因此每個獨特的文件結構都會改變大小計算。

交易大小

QLDB 中的交易大小上限為 4 MB。交易的大小是根據下列因素的總和來計算。

三角洲

由交易中的所有陳述式所產生的文件變更。在影響數個文件的交易中，總差異大小是每個受影響文件的個別增量的總和。

中繼資料

與每個受影響文件相關聯的系統產生的交易中繼資料。

索引

如果在受交易影響的資料表上定義索引，則相關聯的索引項目也會產生差異。

歷史記錄

因為所有文件修訂都會保留在 QLDB 中，所有交易也會附加至歷史記錄。

插入 — 插入表格中的每個文件也會在其記錄表格中插入一個副本。例如，新插入的 100 KB 文件會在交易中產生至少 200 KB 的差異。（這是一個粗略的估計，不包括元數據或索引。）

更新 — 任何文件更新 (即使是單一欄位) 都會在歷程記錄中建立整個文件的新修訂版本，加上或減去更新的差異值。這意味著大型文檔中的小更新仍然會生成大型事務增量。例如，在現有的 100 KB 文件中新增 2 KB 的資料，會在歷史記錄中建立新的 102 KB 修訂版。這增加了交易中至少 104 KB 的總增量。(同樣地，這個估計值不包含中繼資料或索引。)

刪除 — 與更新類似，任何刪除傳遞都會在記錄中建立新的文件修訂版本。不過，新建立的DELETE修訂版本小於原始文件，因為它有空的使用者資料，且只包含中繼資料。

命名限制條件

下表說明 Amazon QLDB 中的命名限制。

分類帳名稱	• 只能包含 1—32 個英數字元或連字號。 • 第一個和最後一個字元必須有一個字母或數字。 • 不能是所有數字。 • 不能連續包含兩個連字號。 • 區分大小寫。
期刊串流名稱	
資料表名稱	• 只能包含 1—128 個英數字元或底線。 • 第一個字元必須為字母或底線。 • 其餘字元可包含英數字元和底線的任意組合。 • 區分大小寫。 • 不得為 QLDB PartiQL 留字。

Amazon QLDB 相關資訊

以下相關資源可協助您使用此服務。

主題

- 技術文件
- GitHub 儲存庫
- AWS部落格文章和文章
- 媒體
- 一般 AWS 資源

技術文件

- 亞馬遜 QLDB 常見問題解答 — 有關產品的常見問題。
- 亞馬遜 QLDB 定價 —AWS 定價資訊和範例。
- AWS re:Post—AWS 社區論壇的問題和答案 (Q&A).
- Amazon Ion — Amazon Ion 資料格式的開發人員指南、使用者指南和參考資料。
- PartiQL — PartiQL 查詢語言的規格文件和一般教學課程。
- QLDB 工作坊 — 研討會提供使用 Amazon QLDB 建置 system-of-record 應用程式的實際實作範例，包括下列實驗室：
  - 學課程
  - 使用亞馬遜離子，並將離子與 JSON（Java）之間進行轉換
  - 使用AWS Glue和 Amazon Athena 為資料湖啟用 QLDB 資料
  - 將 QLDB 資料留
- 使用 Amazon QLDB 防竄改品質資料 — 這是一種AWS解決方案實作，說明如何使用 QLDB 維護準確的資料變更歷史記錄，防止攻擊者竄改高品質資料。 AWS解決方案實作可協助您解決常見問題並使用AWS.

GitHub 儲存庫

驅動程式

- .NET 驅動程式 — QLDB 驅動程式的 .NET 實作。

- 轉到驅動程序-QLDB 驅動程序的 Go 實現。
- Java 驅動程式 — QLDB 驅動程式的 Java 實作方式。
- Node.js 驅動程式 — 一個 Node.js 實作的 QLDB 驅動程式。
- Python 驅動程式 — 一個 Python 實作的 QLDB 驅動程式。

命令列

- QLDB 外殼 — 用於 QLDB 交易資料 API 的命令列介面的 Python 和 Rust 實作。

範例應用程式

- Java DMV 應用程式 — 以機動車輛部門 (DMV) 使用案例為基礎的教學應用程式。它演示了使用 QLDB 和 Java 的 QLDB 驅動程序的基本操作和最佳實踐。
- .NET DMV 應用程式 — 以 DMV 為基礎的教學課程應用程式，示範使用 QLDB 和適用於 .NET 的 QLDB 驅動程式的基本作業和最佳作法。
- Node.js DMV 應用程式 — 以 DMV 為基礎的教學課程應用程式，示範使用 QLDB 和 Node.js 的 QLDB 驅動程式的基本作業和最佳做法。
- Python DMV 應用程式 — 以 DMV 為基礎的教學課程應用程式，示範使用 QLDB 和適用於 Python 的 QLDB 驅動程式的基本作業和最佳作法。
- 分類帳載入器 — 一種 Java 架構，可使用支援的交付通道 (Amazon SQS、Amazon SNS、Kinesis 資料串流AWS DMS、Amazon MSK 或 EventBridge) 以高速非同步將資料載入 QLDB 分類帳。
- 匯出處理器 — 一種可擴充的 Java 架構，可透過讀取匯出的輸出並依序逐一瀏覽匯出的區塊來處理 Amazon S3 中 QLDB 匯出的工作。
- QLDB 串流 Python 中的範例 Lambda — 此應用程式示範如何使用AWS Lambda函數將 QLDB 資料傳送至 Amazon SNS 主題 (已訂閱 Amazon SQS 佇列) 來使用 QLDB 串流。
- QLDB 串流 OpenSearch 整合範例 — 示範如何使用串流將亞馬遜 OpenSearch 服務與 QLDB 整合的 Python 應用程式。
- 雙重項目應用程式 — 一種 Java 應用程式，示範如何使用 QLDB 建立雙重項目財務分類帳應用程式的模型。
- 適用於 Node.js 的 QLDB KVS — 適用於 QLDB 的簡單鍵值存儲介面庫，具有用於文檔驗證的額外功能。

Amazon Ion PartiQL on

- Amazon Ion 程式庫 — Ion 團隊支援程式庫、工具和文件。
- PartiQL 實作 — PartiQL 的實作、規格和教學課程。

AWS部落格文章和文章

- Earnin 如何使用 Amazon QLDB 建立分類帳服務 (2023 年 2 月 16 日) — 說明 Earnin.com 如何使用 QLDB 建立分類帳服務，為使用者提供功能齊全的現代行動金融應用程式。
- 使用和亞馬遜雅典娜 (2022 年 12 月 19 日) 匯出AWS Glue和分析 Amazon QLDB 日誌資料 — 討論如何使用 QLDBAWS Glue 和 Athena 中的匯出功能，為您的總帳架構提供報告和分析功能。
- 新世界國際如何透過 Amazon QLDB 提升客戶體驗並防止仿冒品 (2022 年 8 月 3 日) — 描述新世界國際如何使用 Amazon QLDB 建立數位真實性驗證服務，以通知客戶奢侈品的真實性，並提供產品的購買和分銷歷史。
- BungkuSit 使用 Amazon QLDB 和 G VeriDoc lobal 的 ISV 技術來改善客戶和交付代理商體驗 (2022 年 4 月 29 日) — 顯示一家物流公司 BungkuSit 如何使用 QLDB 實作集中式、透明的產業間通訊平台，並使用不可變、可加密驗證的交易日誌。
- 使用 Amazon QLDB 做為具有 REST API 和 JSON 的不可變金鑰值存放區 (2022 年 2 月 14 日) — 介紹了一種將 QLDB 當作不可變金鑰值存放區使用的方法，並透過 REST API 使用其稽核功能。
- 使用 Amazon QLDB 建置核心銀行系統 (2022 年 1 月 21 日) — 示範如何使用 QLDB 建置核心銀行分類帳系統。它還說明了為什麼 QLDB 是一個適合金融服務業需求的良好 fit-for-purpose 數據庫。
- Specright 如何使用 Amazon QLDB 建立可追蹤的供應鏈網路 (2022 年 1 月 21 日) — 顯示 Specright 如何使用 QLDB 建立可追蹤的供應鏈網路，讓批發品牌、零售商和製造商能夠共用關鍵的供應鏈資料和包裝規格資料。
- FedMR 如何透過 Amazon QLDB 交付加密安全且可驗證的醫療資料 (2021 年 12 月 23 日) — 探索 FEMR 團隊如何使用 QLDB 和其他AWS受管服務來實現救援工作。
- 監控 Amazon QLDB 查詢存取模式 (2021 年 11 月 8 日) — 顯示如何將交易中執行的 QLDB 交易和 PartiQL 陳述式與透過 Amazon API Gateway 接收的 API 請求建立關聯。
- 使用AWS Lambda(2021 年 9 月 28 日) 在 Amazon QLDB 上建置簡單的 CRUD 作業和資料串流 — 示範如何使用AWS Lambda函數在 QLDB 上執行 CRUD (建立、讀取、更新和刪除) 操作。
- 使用 Amazon QLDB 進行的真實世界加密驗證 (2021 年 8 月 26 日) — 在實際使用案例中討論 QLDB 分類帳中加密驗證的價值。
- 使用 Accord 專案和 Amazon QLDB 驗證交付條件：第 1 部分，第 2 部分 (2021 年 6 月 28 日) — 討論如何使用開放原始碼 Accord 專案和 QLDB 應用智慧法律合約技術來驗證交付條件。

- 透過AWS CDK(2021 年 6 月 7 日) 的 Amazon QLDB 資料串流 — 顯示如何使用設定 QLDB、使用 AWS Lambda函數填入 QLDB 資料，以及設定 QLDB 串流以提供總帳資料的資料復原能力。AWS Cloud Development Kit (AWS CDK)
- QLDB 中的精細存取控制示範 (2021 年 6 月 1 日) — 說明如何開始使用 QLDB 分類帳的精細AWS Identity and Access Management (IAM) 許可。
- 使用 DynamoDB 串流和 QLDB 串流進行串流處理 (2020 年 11 月 23 日) — 提供 DynamoDB 串流和 QLDB 串流之間的簡短比較，以及開始使用它們的秘訣。
- 將資料從 Amazon QLDB 串流到 OpenSearch (2020 年 8 月 19 日) — 說明如何將資料從 QLDB 串流到 Amazon OpenSearch 服務以支援富文字搜尋和下游分析，例如跨記錄的彙總或指標。
- 如何使用 Nodejs 以近乎即時的方式將資料從 Amazon QLDB 串流至 Dynamo DB — 描述如何將資料從 QLDB 串流到 DynamoDB，以支援單位數延遲和無限擴展的金鑰值查詢。
- 透過以下方式為 Amazon QLDB 建置 GraphQL 介面AWS AppSync：第 1 部分，第 2 部分 (2020 年 5 月 4 日) — 討論如何整合 QLDB，並AWS AppSync在 QLDB 分類帳之上提供多功能的圖形支援 API。

媒體

AWS 影片

- AWS教學課程和示範：QLDB 到極光串流 (2023 年 3 月 17 日；21 分鐘) — 本影片說明實作 QLDB 串流功能的基本概念，並示範如何設定從 QLDB 到 Amazon Aurora MySQL 下游資料庫的資料串流。
- ArcBlock：利用 Amazon QLDB 建立去中心化身分解決方案 (2022 年 5 月 31 日；4 分鐘) — ArcBlock 逐步介紹他們使用 QLDB 建置的去中心化身分識別解決方案。
- AWSRe: Invent 2020：使用 Amazon QLDB 做為核心商業應用程式的資料 system-of-trust 庫 (2021 年 2 月 5 日；30 分鐘) — 了解早期 Amazon QLDB 使用者如何將分類帳資料庫的獨特屬性應用於資料來源和加密驗證，以實作內建資料完整性的記錄系統。
- AWS重新發佈 2020：使用 Amazon QLDB 建置無伺服器應用程式 (2021 年 2 月 5 日，28 分鐘) — 了解如何將 Amazon QLDB 與 Amazon Kinesis 和 Amazon DynamoDB 等服務結合，來建置AWS Lambda、測試和最佳化功能齊全的無伺服器應用程式。
- AWSRe: Invent 2020：建置以稽核為基礎的應用程式，使用 Amazon QLDB 維護資料完整性 (2021 年 2 月 5 日，18 分鐘) — 本會議深入探討 Amazon QLDB 可以解決的問題、回答您何時以及為何使用總帳資料庫的問題，以及分享 Osano 等客戶的使用案例。

- 如何使用 Amazon QLDB 與 .NET 應用程式集中存放不可變日誌 (2020 年 12 月 7 日，10 分鐘) — 示範如何搭配 .NET 應用程式使用 QLDB 來集中存放不可變日誌。
- 虛擬研討會：使用 QLDB 和 Java 構建基於分類帳的記錄系統-AWS 在線技術講座（2020 年 7 月 29 日；87 分鐘）-由講師指導的研討會，通過「使用離子沉浸日實驗室」，使用 Amazon Ion 庫和 Java 的 QLDB 驅動程序構建數據加載程序。
- 開發人員研討會：在 ABT 節點上使用AWS不可變的分類帳資料庫 QLDB（2020 年 6 月 22 日，34 分鐘）— 有關如何在 ABT 節點上設置和配置 QLDB 的 step-by-step 指南。它還說明如何安裝和配置 ArcBlock區塊鏈資源管理器和登機閘口區塊以連接到 QLDB。
- AWS技術講座：客戶透過 Amazon QLDB 建置事件觸發式記錄系統應用程式的觀點 (2020 年 3 月 31 日，29 分鐘) — Matt Lewis 的技術演講 —AWS 資料英雄兼英國駕駛員和車輛授權機構 (DVLA) 首席架構師 — 說明 DVLA 在 QLDB 及其應用程式事件驅動架構的使用案例。
- AWS回复：Invent 2019：為什麼需要分類帳數據庫：寶馬，DVLA 和 Sage 討論用例（2019 年 12 月 5 日；47 分鐘）— 客戶的演示 BMW，英國政府組織 DVLA 和 Sage 討論了使用分類帳數據庫並共享其用於 QLDB 的用例的原因。
- AWSRe: Invent 2019：亞馬遜 QLDB：工程師深入探討為什麼這是一個改變遊戲規則的原因（2019 年 12 月 5 日；50 分鐘）— 安德魯·特定（AWS傑出工程師）的演講，討論了 QLDB 獨特的日誌優先架構以及其各種創新。它包括加密雜湊、默克爾樹、多個可用區域複寫和 PartiQL 支援。
- 使用 Amazon QLDB 建置應用程式，這是首個同類型的總帳資料庫-AWS 線上技術會談 (2019 年 11 月 19 日，51 分鐘) — 深入的技術演講，說明 QLDB 的獨特功能以及如何使用核心功能的詳細資訊。它包括查詢數據的完整歷史記錄，加密驗證文檔以及設計數據模型。

播客

- 為什麼客戶選擇亞馬遜 QLDB？（2020 年 7 月 5 日；33 分鐘）— 一場討論，解釋了分類帳數據庫的定義，它與區塊鏈有何不同，以及客戶當今如何使用它。

一般 AWS 資源

- 課程和研討會 – 連結至以角色為基礎的專門課程以及自主進度實驗室，協助加強您的 AWS 技能，並取得實際體驗。
- AWS 開發人員中心 – 研究教學課程、下載工具，以及瞭解 AWS 開發人員活動。
- AWS 開發人員工具 – 連結至開發人員工具、軟體開發套件、IDE 工具組和命令列工具，用來開發及管理 AWS 應用程式。
- 入門資源中心 – 瞭解如何設定 AWS 帳戶、加入 AWS 社群，並啟動您的第一個應用程式。

- 實用的教學 step-by-step 課程啟動您的第一個應用程式AWS。
- AWS 白皮書 – 連結至完整的技術 AWS 白皮書清單，其中涵蓋了架構、安全和成本等主題，並由 AWS 解決方案架構師或其他技術專家撰寫。
- AWS Support 中心 – 建立和管理您的 AWS Support 案例的中心。這也包含與其他實用資源的連結，例如論壇、技術常見問答集、服務運作狀態以及 AWS Trusted Advisor。
- AWS Support— 有關的資訊的主要網頁AWS Support，它是快速回應支援頻道 one-on-one，可協助您在雲端中建置並執行應用程式。
- 聯絡我們 – 查詢有關 AWS 帳單、帳戶、事件、濫用與其他問題的聯絡中心。
- AWS 網站條款 – 我們的著作權與商標；您的帳戶、授權與網站存取；以及其他主題的詳細資訊。

Amazon QLDB 的版本歷史記錄

下表說明 Amazon QLDB 版本的重要變更，以及 Amazon QLDB 開發人員指南中的重要變更。如需有關此文件更新的通知，您可以訂閱 RSS 摘要。

- 應用程式介面版本:
- 最新文件更新時間：2023 年 1 月 3 日

變更	描述	日期
更新的 IAM 指引	更新了指南以符合 IAM 最佳實務。如需詳細資訊，請參閱 IAM 中的安全最佳實務。	2023 年 1 月 3 日
更新到AWS受管理的策略	亞馬遜 QLDB 更新了現有的AWS受管政策`AmazonQLDBFullAccess` 和`AmazonQLDBConsoleFullAccess` 。這些原則擁有新的權限，可讓主參與者使用 PartiQL 預存程序來編輯文件修訂版本。如需詳細資訊，請參閱 Amazon QLDB 的AWS受管政策。	2022 年 11 月 4 日
資料編輯	亞馬遜 QLDB 現在支援在 2021 年 7 月 22 日或之後建立的分類帳的`REDACT_REVISION` PartiQL 預存程序。使用此預存程序，您可以永久刪除歷史記錄中的無效文件版次，並仍維護分類帳的整體資料完整性。如需詳細資訊，請參閱修訂文件修訂版本修訂。	2022 年 11 月 3 日

Node.js 驅動程式版本	適用於 Node.js 3.0 版的亞馬遜 QLDB 驅動程式現已正式推出。此版本引入了對AWS SDK for JavaScript v3 的支持。如需版本說明，請參閱 GitHub 儲存庫 awslab/amazon-qldb-driver-nodejs.	2022 年 9 月 26 日
去驅動程序 v3	Go 版本 3.0 的亞馬遜 QLDB 驅動程序現已正式推出。此版本引入了對AWS SDK for Go v2 的支持。如需版本說明，請參閱 GitHub 儲存庫 awslab/amazon-qldb-driver-go.	2022 年 8 月 11 日
新的 PartiQL 查詢編輯器	現在通常可以在 Amazon QLDB 主控台上全新的 PartiQL 查詢編輯器全面推出。新的 QLDB PartiQL 編輯器為編寫查詢、偵錯交易和探索結果提供了改良的介面。如需開啟和使用編輯器的相關資訊，請參閱使用主控台存取 Amazon QLDB。	2022 年 6 月 22 日
.NET 驅動程序的離子對象映射器	.NET 版本 1.3 的亞馬遜 QLDB 驅動程序引入了對離子對象映射器的支持。此功能可讓您完全避免在 Amazon Ion 類型和原生 C# 類型之間手動轉換的需求。有關 Ion 對象映射器的完整更改歷史記錄，請參閱 GitHub 存儲庫中的更改日誌 .md 文件`amzn/ion-object-mapper-dotnet` 。	2022 年 1 月 19 日

JSON 分錄匯出格式	亞馬遜 QLDB 現在支援日誌匯出的 JSON 行輸出格式。如需詳細資訊，請參閱 Amazon QLDB 匯出日誌資料。	2021 年 12 月 21 日
疑難排解 Amazon QLDB	已新增疑難排解主題，提供使用 Amazon QLDB 時可能遇到的常見錯誤彙總清單的指引。	2021 年 12 月 8 日
推出新區域	Amazon QLDB 現已在加拿大 (中部) 區域現已在加拿大 (中部) 區域提供。如需可用區域的完整清單，請參閱中的 Amazon QLDB 端點和配額Amazon Web Services 一般參考。	2021 年 11 月 11 日
預防跨服務混淆代理人	Amazon QLDB 現在支援在 IAM 資源政策中使用`aws:SourceArn` 和`aws:SourceAccount` 全域條件內容金鑰，以避免混淆代理人問題。如需詳細資訊，請參閱預防跨服務混淆代理人。	2021 年 11 月 8 日
亞馬遜 QLDB 外殼 V2	亞馬遜 QLDB 外殼 2.0 版，這是寫在 Rust，現已正式推出。如需版本說明，請參閱 GitHub 儲存庫 awslab/amazon-qldb-shell.	2021 年 10 月 14 日

更新到AWS受管理的策略	亞馬遜 QLDB 更新了現有的AWS受管政策AmazonQLDBFullAccess 和AmazonQLDBConsoleFullAccess 。這些政策具有新增的許可，允許主體將您帳戶中的任何 IAM 角色資源傳遞至 QLDB 服務。所有日誌匯出和串流請求都需有此許可。如需詳細資訊，請參閱 Amazon QLDB 的AWS受管政策。	2021 年 9 月 2 日
客戶管理的AWS KMS金鑰	Amazon QLDB 現在支援使用新總帳資源AWS Key Management Service (AWS KMS) 中的客戶受管金鑰進行靜態加密。如需詳細資訊，請參閱中的靜態加密。	2021 年 7 月 22 日
更新到AWS受管理的策略	Amazon QLDB 更新了現有的AWS受管政策，AmazonQLDBReadOnly 以移除先前列出兩次的重複qldb:GetBlock 動作。如需詳細資訊，請參閱 Amazon QLDB 的AWS受管政策。	2021 年 7 月 1 日
推出新區域	Amazon QLDB 現已在歐洲 (倫敦) 區域現已在歐洲 (倫敦) 區域提供。如需可用區域的完整清單，請參閱中的 Amazon QLDB 端點和配額Amazon Web Services 一般參考。	2021 年 6 月 24 日

更新到AWS受管理的策略	亞馬遜 QLDB 更新了現有的AWS受管政策AmazonQLDBFullAccess 和AmazonQLDBConsoleFullAccess ．這些原則具有新增的權限，STANDARD可讓主參與者更新所有分類帳中的權限模式，並在所有權限分類帳中執行所有 PartiQL 命令。如需詳細資訊，請參閱 Amazon QLDB 的AWS受管政策。	2021 年 5 月 27 日
標準權限模式	Amazon QLDB 現在支援分類帳資源的STANDARD許可模式。使用標準許可模式，您可以使用更細的層級來控制分類帳、資料表和 PartiQL 命令的存取。如需詳細資訊，請參閱標準權限模式入門。	2021 年 5 月 27 日
PartiQL 陳述式統計	PartiQL 陳述式統計資料功能現在可在 Amazon QLDB 主控台的查詢編輯器中使用。如需詳細資訊，請參閱取得陳述式統計資料	2021 年 5 月 24 日
教學課程：使用AWS SDK 驗證資料	已新增包含 step-by-step 程式碼範例的教學課程，示範如何透過AWS SDK 使用 QLDB API 在 Amazon QLDB 中驗證修訂雜湊和區塊雜湊。若要深入了解，請參閱教學課程：使用AWS SDK 驗證資料。	2021 年 5 月 6 日

. NET 驅動程式	適用於 .NET 1.2 版的亞馬遜 QLDB 驅動程式現已正式推出。此版本引入了異步 API。如需版本說明，請參閱 GitHub 儲存庫 awslab/amazon-qldb-driver-dotnet.	2021 年 4 月 1 日
PartiQLDROP INDEX 陳述式	亞馬遜 QLDB 中的 PartiQL 現在支持掉落索引語句。如需詳細資訊，請參閱卸除索引。	2021 年 3 月 3 日
PartiQL 陳述式統計	PartiQL 陳述式統計資料功能現已在最新版本的 Amazon QLDB 驅動程式中提供，適用於所有支援的語言，包括 .NET、Go 和 Python。如需詳細資訊，請參閱取得陳述式統計資料	2021 年 2 月 25 日
TypeScript 快速入門教學	在 Node.js 的亞馬遜 QLDB 驅動程序的快速入門教程中添加了 TypeScript 代碼示例。	2020 年 12 月 28 日
PartiQL 陳述式統計	適用於 Java 和 Node.js 的最新版本 Amazon QLDB 驅動程式現在提供陳述式執行統計資料，可協助您執行更有效率的 PartiQL 陳述式。如需詳細資訊，請參閱取得陳述式統計資料	2020 年 12 月 22 日

驅動程序食譜參考和快速入門教程	已新增 Go 和 Node.js 驅動程式的食譜參考資料、Java 和 Python 驅動程式的快速入門教學課程，以及在 QLDB 中使用亞馬遜離子的指南。如需詳細資訊，請參閱 Amazon QLDB 驅動程式入門。	2020 年 11 月 24 日
亞馬遜 QLDB 外殼 1.1 版	亞馬遜 QLDB 外殼 1.1 版現已正式推出。如需版本說明，請參閱 GitHub 儲存庫 awslab/amazon-qldb-shell.	2020 年 10 月 26 日
適用於 Go 的亞馬遜 QLDB 驅動程序	Amazon QLDB Driver for Go 現已全面推出。此驅動程式是開放 GitHub 原始碼的，可讓您使用與 QLDB 的交易資料 API 互動。AWS SDK for Go如需版本說明，請參閱 GitHub儲存庫 awslab/amazon-qldb-driver-go.	2020 年 10 月 20 日
Node.js 驅動程式安裝建議	已新增新章節，提供 Node.js 適用之 Amazon QLDB 驅動程式的設定建議，包括如何透過重複使用連線以保持連線來減少延遲。	2020 年 10 月 16 日
在非空白資料表上建立索引	Amazon QLDB 現在支援在非空白資料表上建立索引。如需詳細資訊，請參閱管理索引。	2020 年 9 月 30 日
最佳化查詢效能	新增一節說明 Amazon QLDB 中的查詢限制，並提供有關在這些限制下優化查詢效能的指導。	2020 年 9 月 18 日

Java 驅動程序的食譜參考	已新增食譜參考資料，其中顯示適用於 Java 的 Amazon QLDB 驅動程式常見使用案例的程式碼範例。	2020 年 9 月 3 日
驅動程式的工作階段管理	新增一節，提供 Amazon QLDB 中驅動程式的工作階段管理概觀，並說明 QLDB 驅動程式在執行資料交易時如何處理工作階段。	2020 年 9 月 1 日
Java 驅動程序 2.0	適用於 Java 2.0 版的亞馬遜 QLDB 驅動程序現已正式推出。如需版本說明，請參閱 GitHub 儲存庫 awslab/amazon-qldb-driver-java.	2020 年 8 月 28 日
Node.js 驅動程式	適用於 Node.js 2.0 版的亞馬遜 QLDB 驅動程式現已正式推出。如需版本說明，請參閱 GitHub 儲存庫 awslab/amazon-qldb-driver-nodejs.	2020 年 8 月 27 日
相關資訊	新增包含相關資訊連結和其他資源的新主題，可協助您瞭解並使用 Amazon QLDB。	2020 年 8 月 24 日
Python Driver for v3.0	適用於 Python 3.0 版的亞馬遜 QLDB 驅動程序現已正式推出。如需版本說明，請參閱 GitHub 儲存庫 awslab/amazon-qldb-driver-python.	2020 年 8 月 20 日

適用於 Go 的亞馬遜 QLDB 驅動程序	適用於 Go 的 Amazon QLDB 驅動程式的預覽版現已推出。此驅動程式可讓您使AWS SDK for Go用與 QLDB 的交易資料 API 互動。如需詳細資訊，請參閱 Amazon QLDB 驅動程式 (預覽)。	2020 年 8 月 6 日
Python 驅動程序的食譜參考	已新增食譜參考資料，其中顯示適用於 Python 的 Amazon QLDB 驅動程式常見使用案例的程式碼範例。	2020 年 7 月 24 日
亞馬遜 QLDB 中的唯一 ID	新增全新章節說明 Amazon QLDB 中的唯一 ID 的屬性和使用準則。	2020 年 7 月 9 日
AWS CloudFormation日誌串流的資源	Amazon QLDB 現在支援使用 AWS CloudFormation範本建立日誌串流的資源。若要取得更多資訊，請參閱《AWS CloudFormation使用指南》中的AWS::QLDB::Stream資源。	2020 年 7 月 9 日
.NET 驅動程式的食譜參考	已新增食譜參考資料，其中顯示適用於 .NET 之 Amazon QLDB 驅動程式常見使用案例的程式碼範例。	2020 年 7 月 1 日
用於. NET 的亞馬遜 QLDB 驅動程序	適用於 .NET 的亞馬遜 QLDB 驅動程序現已正式推出。此驅動程式是開放 GitHub 原始碼的，可讓您使用與 QLDB 的交易資料 API 互動。AWS SDK for .NET如需版本說明，請參閱 GitHub 儲存庫 awslab/amazon-qldb-driver-dotnet.	2020 年 6 月 26 日

Node.js 的亞馬遜 QLDB 驅動程序	適用於 Node.js 的亞馬遜 QLDB 驅動程式現已正式推出。這個驅動程式是開 GitHub 放原始碼的，可讓您 JavaScript 在 Node.js 中使用AWS SDK 與 QLDB 的交易資料 API 互動。如需版本說明，請參閱 awslabs/amazon-qldb-driver-nodejs GitHub 儲存庫。	2020 年 6 月 5 日
亞馬遜 QLDB 日誌流	Amazon QLDB 現在可讓您建立串流，擷取提交到日誌的每個文件修訂版本，並以近乎即時的方式將這些資料傳送到 Amazon Kinesis 資料串流。如需詳細資訊，請參閱 Amazon QLDB 中的日誌資料。	2020 年 5 月 19 日
亞馬遜離子代碼示例	已新增程式碼範例，使用適用於 Java、Node.js 和 Python 的 QLDB 驅動程式，在亞馬遜 QLDB 分類帳中查詢和處理亞馬遜離子資料。如需詳細資訊，請參閱 Amazon QLDB onon 代碼範例。	2020 年 5 月 12 日
並行模型與日誌內容	擴充 Amazon QLDB 並行模型主題，以新增有關使用索引限制最佳並行控制 (OCC) 衝突的相關資訊。已新增描述 Amazon QLDB 中期刊內容的新章節。	2020 年 5 月 4 日
Node.js 驅動程式的快速入門指南	為 Node.js 的亞馬遜 QLDB 驅動程序添加了一個快速入門指南。	2020 年 5 月 1 日

亞馬遜 QLDB 外殼	Amazon QLDB 外殼現已全面推出。此殼層是開放原始碼，GitHub 並提供命令列介面，可讓您在分類帳資料上執行 PartiQL 陳述式。如需版本說明，請參閱 awslabs/amazon-qldb-shell GitHub 儲存庫。	2020 年 4 月 20 日
合規性認證	Amazon QLDB 現已獲得合AWS規計劃的認證，包括 HIPAA 和 ISO。如需詳細資訊，請參閱 Amazon QLDB 合規驗證。	2020 年 4 月 3 日
擴充的驅動程式	擴展 Amazon QLDB 驅動程式建議部分，使其適用於所有支援的程式設計語言。	2020 年 4 月 2 日
亞馬遜 QLDB 外殼	亞馬遜 QLDB 殼層的預覽版現已推出。此命令介面提供命令列介面，可讓您在分類帳資料上執行 PartiQL 陳述式。如需詳細資訊，請參閱使用 QLDB 殼層存取 Amazon QLDB (僅限資料 API) (預覽版)。	2020 年 3 月 23 日
Java Driver for v1.1	適用於 Java 1.1 版的亞馬遜 QLDB 驅動程序現已正式推出。如需版本說明，請參閱 awslabs/amazon-qldb-driver-java GitHub 儲存庫。	2020 年 3 月 20 日

用於. NET 的亞馬遜 QLDB 驅動程序	亞馬遜 QLDB 現在提供了 .NET 驅動程式的預覽版本。此驅動程式可讓您使AWS SDK for .NET用與 QLDB 的交易資料 API 互動。如需詳細資訊，請參閱 Amazon QLDB 驅動程式 (預覽版)。	2020 年 3 月 13 日
亞馬遜 QLDB 驅動程序	Amazon QLDB Driver for Python 現已全面推出。此驅動程式是開放 GitHub 原始碼的，可讓您使用與 QLDB 的交易資料 API 互動。AWS SDK for Python (Boto3)如需版本說明，請參閱 awslabs/amazon-qldb-driver-python GitHub 儲存庫。	2020 年 3 月 11 日
PartiQLUNDROP TABLE 句和嵌套 DML	Amazon QLDB 中的 PartiQL 現在支援資料操作語言 (DML) 陳述式，其中巢狀集合會指定為FROM子句中的來源。如需詳細資訊，請參閱 P artiQL 參考資料中的 FROM 陳述式。QLDB 分區也支援取消刪除資料表陳述式。如需詳細資訊，請參閱取消刪除資料表。	2020 年 2 月 26 日
擴展的介紹主題	擴大介紹什麼是亞馬遜 QLDB？ 主題包含 Amazon QLDB 的新章節概觀以及從關聯式到分類帳的概觀。	2020 年 1 月 24 日

支援函數的 PartiQL 參考	擴充 Amazon QLDB 參考資 PartiQL，以包含有關支援 SQL 函數的詳細使用資訊。如需詳細資訊，請參閱 PartiQL 函數。	2020 年 1 月 2 日
驅動程式建議和常見錯誤	已新增新章節，說明適用於 Java 之 Amazon QLDB 驅動程式的驅動程式建議，以及所有驅動程式語言的常見錯誤。	2020 年 1 月 2 日
推出新區域	Amazon QLDB 現在在亞太區域 (首爾)、亞太區域 (新加坡)、亞太區域 (雪梨) 和歐洲 (法蘭克福) 區域全面提供 Amazon QLDB。如需可用區域的完整清單，請參閱中的 Amazon QLDB 端點和配額Amazon Web Services 一般參考。	2019 年 11 月 19 日
Node.js 的亞馬遜 QLDB 驅動程序	亞馬遜 QLDB 現在提供 Node.js 驅動程式的預覽版本。此驅動程式可讓您 JavaScript 在 Node.js 中使用的AWS SDK 與 QLDB 的交易資料 API 互動。如需詳細資訊，請參閱適用於 Node.js 的 Amazon QLDB 驅動程式 (預覽)。	2019 年 11 月 13 日

亞馬遜 QLDB 驅動程序	亞馬遜 QLDB 現在提供了 Python 驅動程序的預覽版本。此驅動程式可讓您使AWS SDK for Python (Boto3)用與 QLDB 的交易資料 API 互動。如需詳細資訊，請參閱 Amazon QLDB Python 驅動程式 (預覽)。	2019 年 10 月 29 日
公開發行	這是 Amazon QLDB 的初始公有版本。此版本包含開發人員指南和整合分類帳管理 API 參考資料。	2019 年 9 月 10 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。