aws DevAx connect

開発者のためのAmazon QLDB入門

Yuki Nakatake
Senior Solutions Architect

自己紹介

名前

中武 優樹（なかたけ ゆうき) aka ザビオ
@zabbiozabbio

所属

ブロックチェーンスペシャリスト
シニアソリューションアーキテクト

好きなAWSサービス

Amazon Aurora , Amazon Managed Blockchain , Amazon Quantum Ledger Database (QLDB)

本セッションの対象者

- QLDBを全く知らない開発者
- QLDBがどこに利用できるかを知りたい方
- ブロックチェーンに馴染みのある方

本セッションのゴール

- QLDBの概要と使い所を知る！

アジェンダ

- 台帳とは
- QLDBについて
- Demo
- まとめ

台帳とは

aws DevAx connect

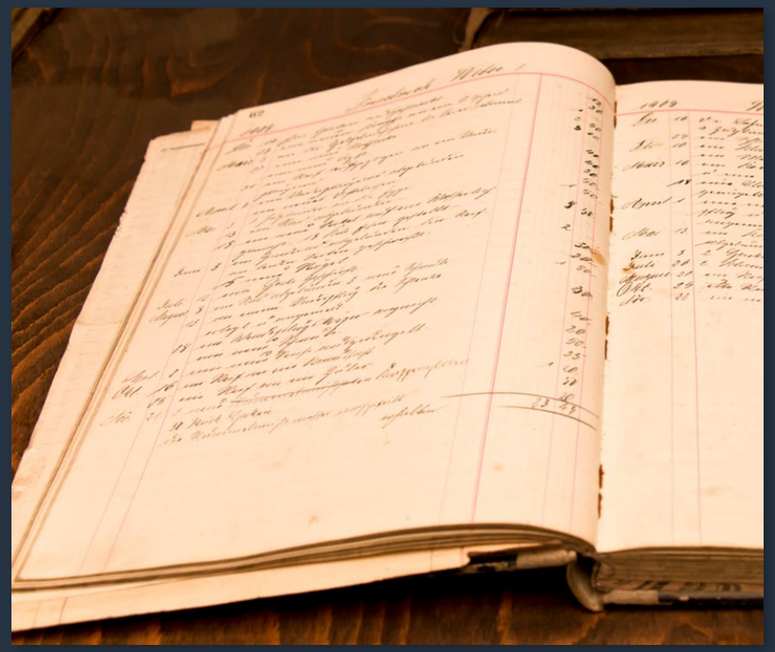
記録保持の歴史

古代メソポタミア

楔形文字-紀元前3500年

古代エジプト

最古のパピルス書面—紀元前2500年



複式簿記

最初に考案された—西暦1494年

すでに世の中に存在している台帳

銀行や金融機関

取引や口座の追跡

製造業

製造時に使用される
コンポーネントの記録

所有権

資産の所有権の記録

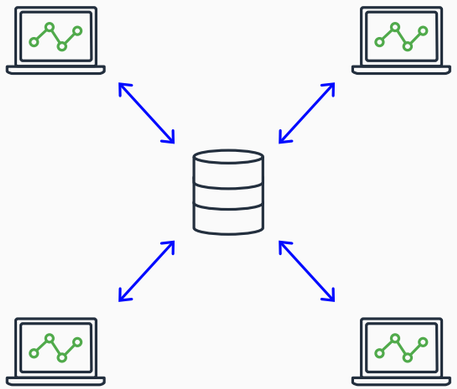
台帳データベースの活用シーン

以下の課題に直面するケース

- 顧客、監査人、規制当局にデータの整合性を証明する必要がある
- 監査において手間をかけない簡単な方法が必要

お客様の悩み – 台帳の必要性

1 中央集権型台帳



2 非中央集権型台帳

ヘルスケア
病院の薬や設備などの在庫の確認および追跡

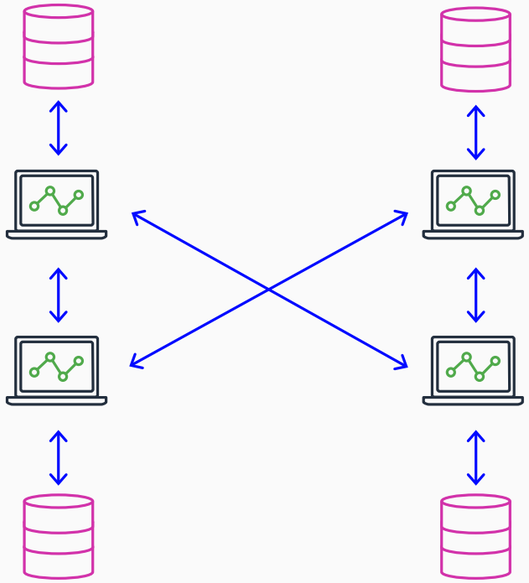
陸運局
所有者履歴の追跡

製造業
リコールされた製品の流通を追跡

人事および給与
個人情報の変更の追跡

お客様の悩み – 台帳の必要性

1 中央集権型台帳



2 非中央集権型台帳

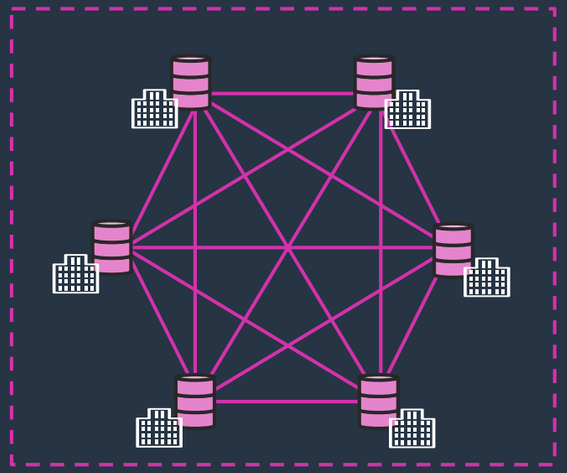
金融機関
個人間送金

住宅ローン会社
シンジケートローン処理

サプライチェーン
供給者と消費者間の取引

小売り
ポイント制度

ニーズのまとめ



ブロックチェーン・ネットワーク

改ざんできない、同一のデータをステークホルダー間で共有する

ユースケース
多くのステークホルダーとデータを共有する作業が発生する場合
・発注書、請求書、といった、突合作業が必要なもの
・取引の内容をみんなで共有して検証し合うことでデータの整合性を担保するもの

台帳データベース

ブロックチェーン同様の技術を利用し、データ変更履歴を改ざんできない、1組織で変更履歴を管理する必要のあるお客様向けの、台帳データベース

ユースケース
1組織内で変更履歴を確実に残す必要があるもの
・監査、法規制、命、会社ブランドに関わるもの

現在の台帳実装の課題

従来型DB

多くのリソース消費

管理とスケールが
困難

実装上のバグが
発生しやすい

検証不可能

ブロックチェーン

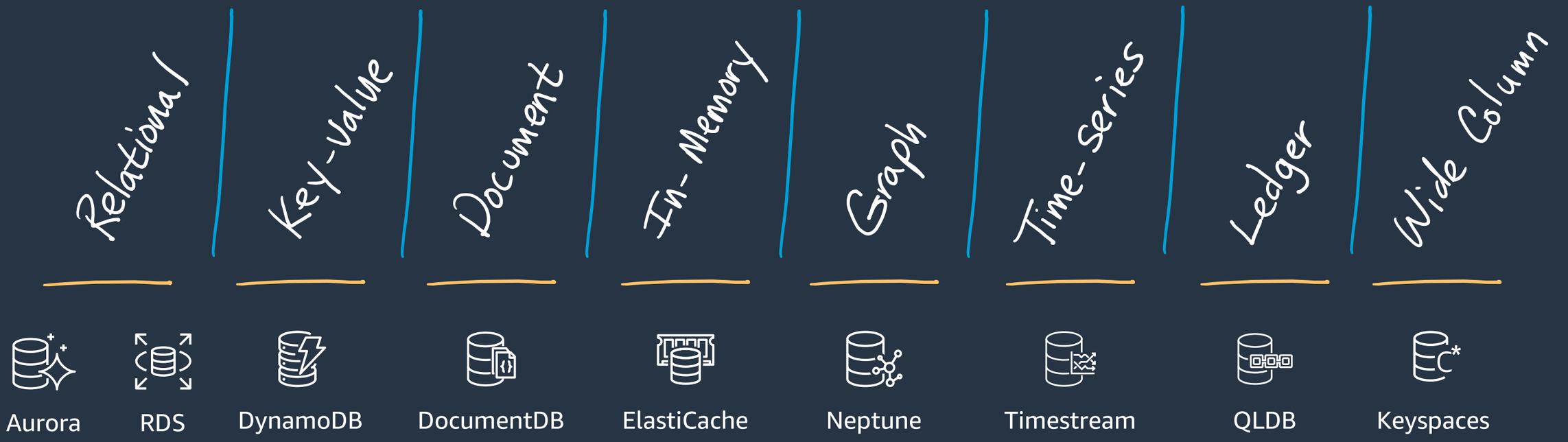
別の目的のための
設計

不要な複雑さの
追加

QLDBについて

aws DevAx connect

Purpose-built databases



The most complete family of purpose-built databases

Purpose-built databases

The most complete family of purpose-built databases

Amazon Quantum Ledger Database (QLDB)

アプリケーションのデータに加えられた、すべての変更の履歴を追跡および検証可能なマネージド台帳データベース。

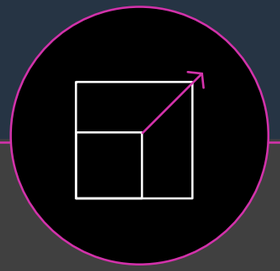
イミュータブル

データに対するすべての変更の順序付けられたレコードを維持します。これは削除または変更することはできません。 完全な履歴を問い合わせそして分析する機能を持っています。

暗号的に検証可能

暗号化を使用してデータの履歴の安全な出力ファイルを生成します。

スケーラブル



一般的なブロックチェーンフレームワークの元帳の2~3倍のトランザクションを実行します。

容易な操作性

使いやすく、SQL APIなどの使い慣れたデータベース機能を使用してデータを照会できます。

Amazon QLDB の機能

不変

追記のみ

暗号学に基づく検証

データ整合性のための
ハッシュチェーン

高いスケーラビリティ

サーバーレス

使いやすい

柔軟なドキュメントモデルと
使い慣れたSQL言語

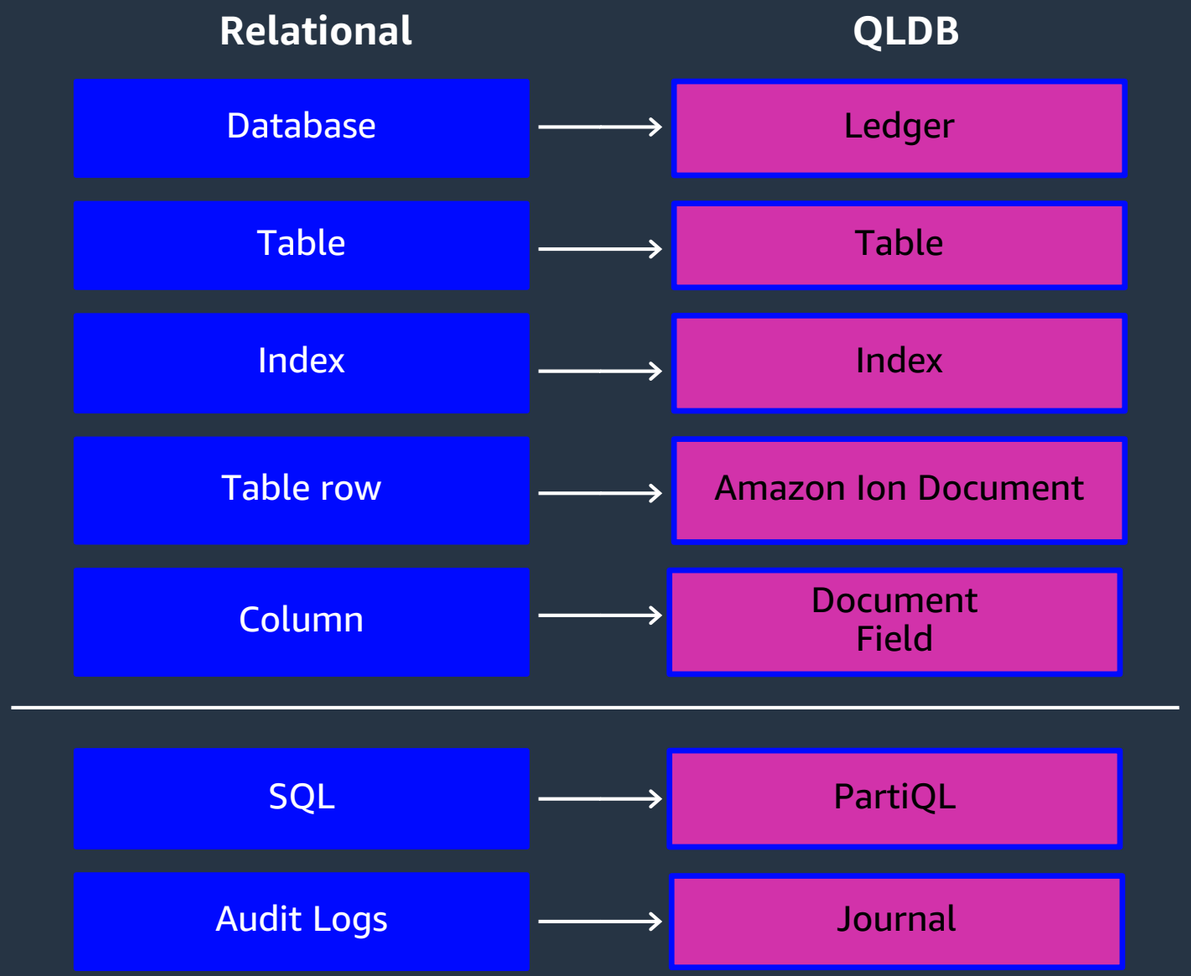
ACIDトランザクション

完全なSerializable分離

ジャーナル先行

ジャーナルこそが
データベース本体

RDBとのマッピング



Easy to use - Amazon Ion & PartiQL

Amazon Ion

```
/* Ion supports comments.
This is a “Vehicle” document */
    {
    ‘VIN’ : ‘KM8SRDHF6EU074761’ ,
    ‘MfgDate’:  `2017-03-01T` ,
    ‘Type’:     ‘Truck’ ,
    ‘Mfgr’:     ‘Ford’ ,
    ‘Model’:    ‘F150’
    ‘Color”:    ‘Black’ ,
    ‘Specs’: {
                ‘EngSize’ : 3.3 ,(decimal)
                ‘CurbWeight’: 4878 , (int)
                ‘HP’: 327   ,(int)
                ‘BatterySize’ : NULL.int ,
              }
            }
```

PartiQL

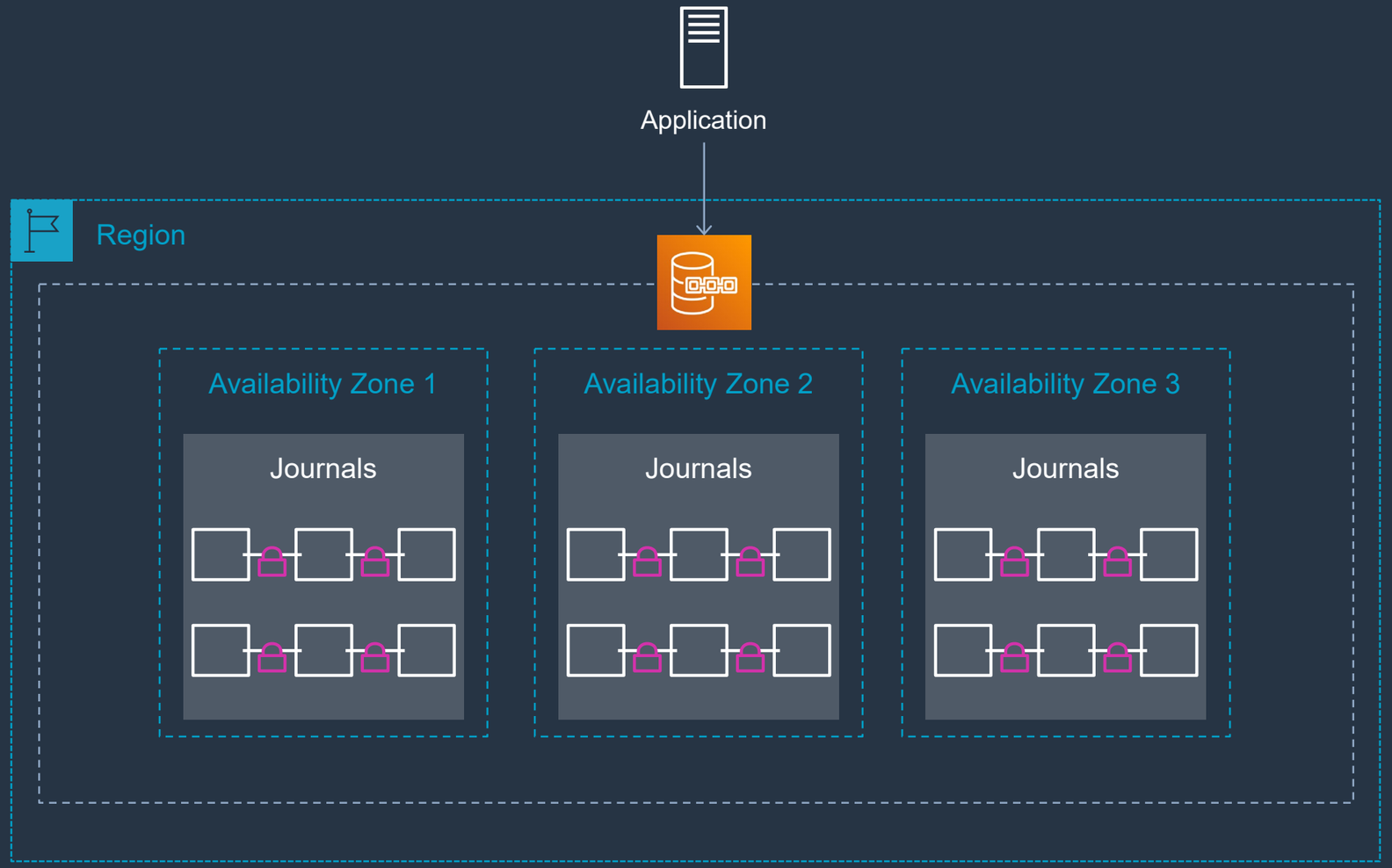
```
INSERT INTO cars
      { 'Manufacturer':'Tesla',
        'Model':'Model S',
        'Year': 2012,
        'VIN': 123456789,
        'Owner':'Traci Russell'
     }

UPDATE cars SET owner = 'Ronnie Nash'

WHERE VIN = 123456789

SELECT * FROM cars
```

Transactions (ACID)

HIGHEST TO LOWEST

Isolation Level	Potential Issues
Serializable	—
Snapshot Isolation	Potential write skew
Repeatable read	Phantom reads
Read committed	Phantom reads/non-repeatable reads
Read uncommitted	Phantom reads/non-repeatable reads/dirty reads

Serverless, scalable, highly available



Multi-AZ for high availability

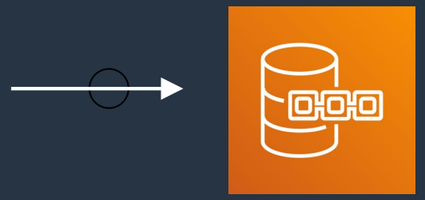
Multiple copies per AZ providing strong durability

QLDB Export 機能

How Amazon QLDB works

Application data

クレジットおよびデビット取引、保険金請求履歴、サプライチェーン資産追跡、車両記録など



Amazon Quantum Ledger Database

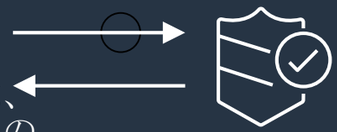
Current state and indexed history

銀行口座の現在の値とその履歴など、データの現在の値と履歴状態を保存します

データ変更履歴にアクセスする

Journal

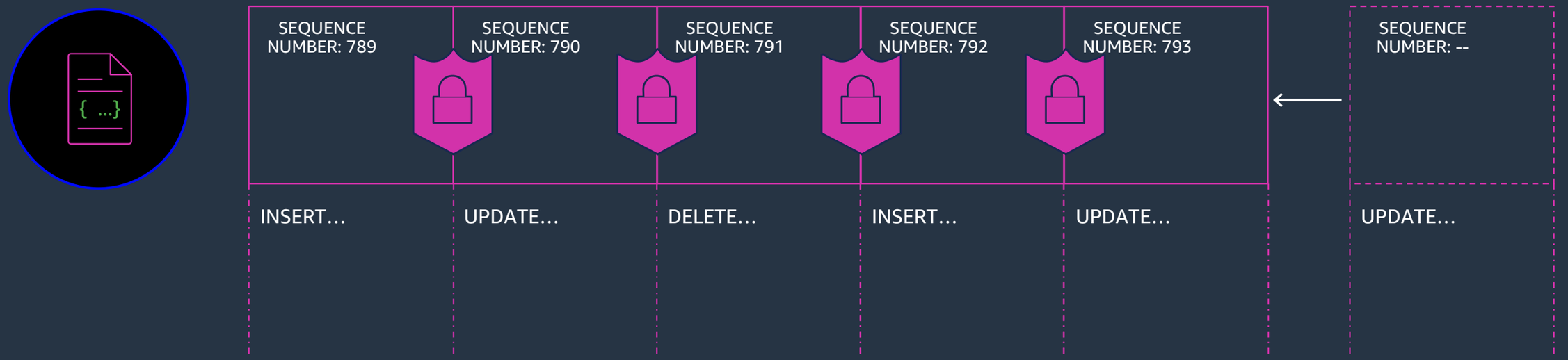
追加専用の不変ジャーナルには、各変更履歴が順序付けされたものが、暗号化され検証可能なエントリが格納されます。変更履歴は、クレジットやデビットなどのブロックとして連結されます



データの変更履歴を暗号化して検証する

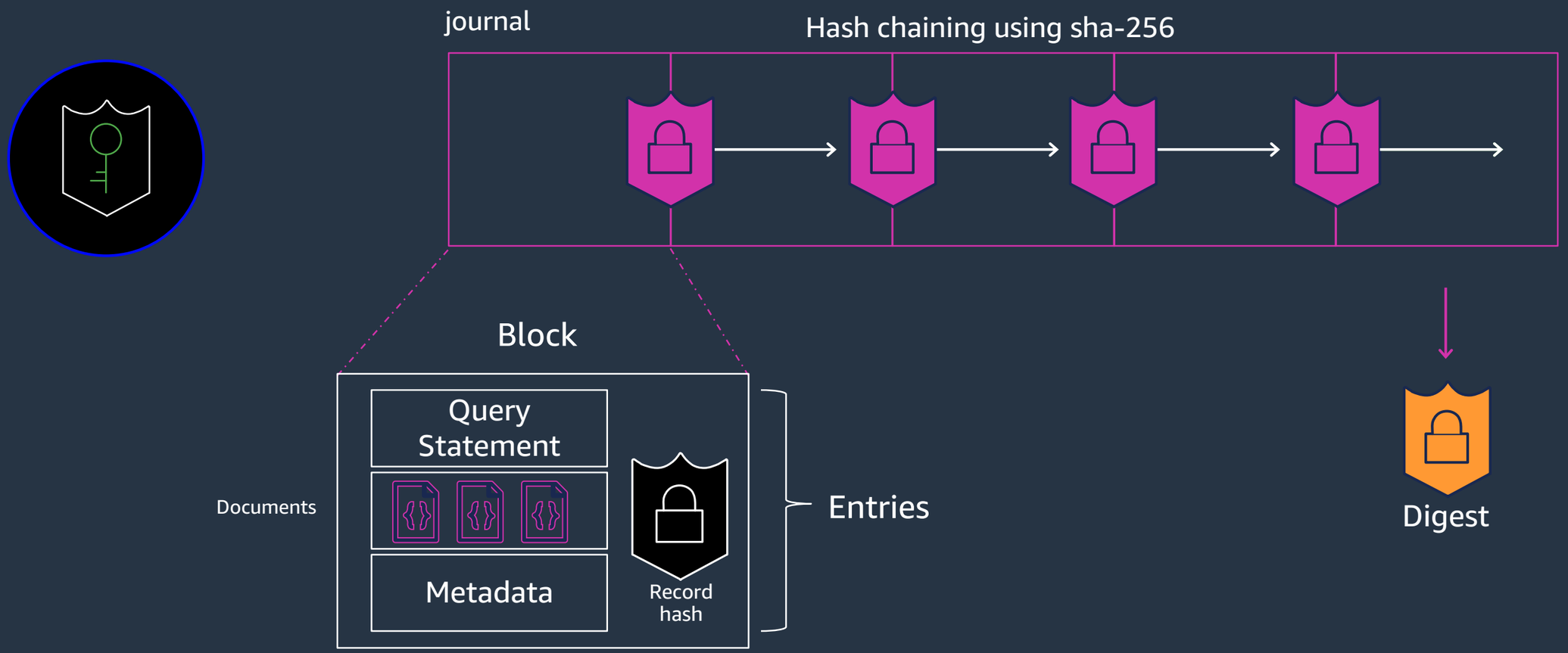
Immutable & Complete

- データは変更できず、ジャーナルは追記のみ
- ジャーナルは、コミットされたすべてのトランザクションとリビジョンの完全な記録です。
- コミットされた操作は、読み取りも含めてすべてジャーナルに書き込まれます。



暗号学的に検証可能

- データが改ざんされていないことを暗号学を使って証明することができる
- 暗号ハッシュはデータの "finger print "である。
- 検証にはジャーナルダイジェストの再計算が必要である

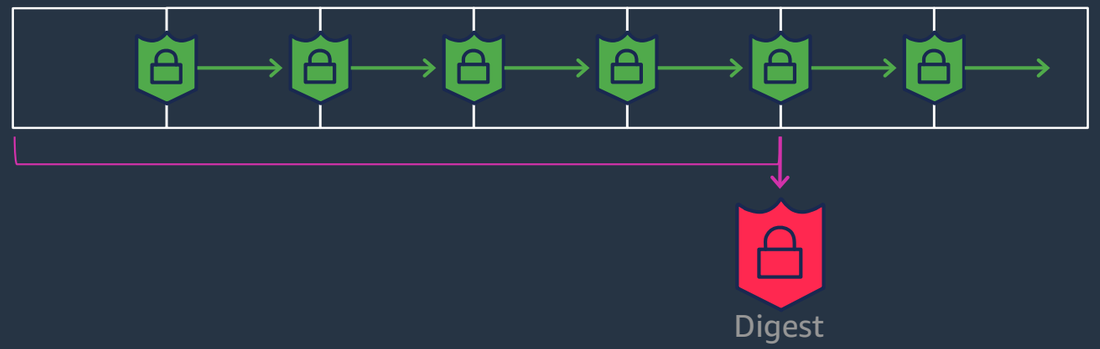


ジャーナルの検証

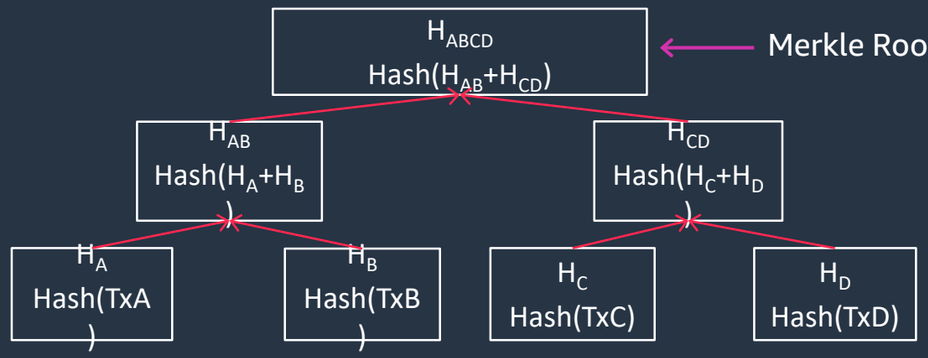
QLDBの検証性を確認するための4つの基本ステップ

SHA256: Unique Signature of a document

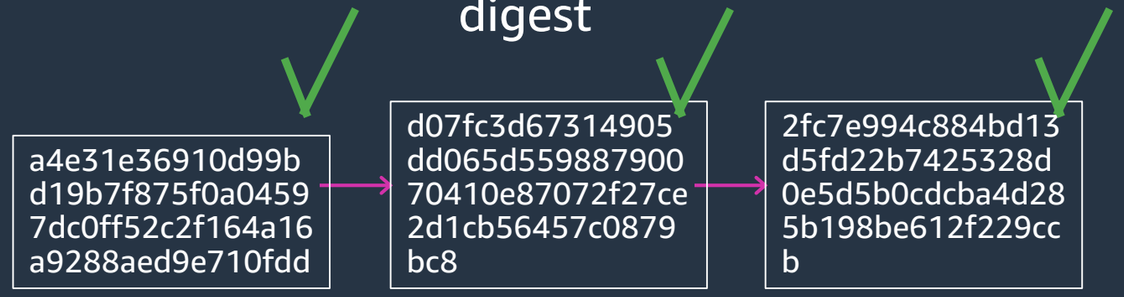
Digest: Periodic hash covering all history



Merkle Trees: Chaining past hashes together



Proof: A chain of hashes linking a document to its digest



aws DevAx connect

Cryptographic verifiability: SHA-256

QLDBはSHA-256アルゴリズムを使って、ユニークな固定長の出力（ハッシュ）を作ります。1文字でも部分的に変更すると、出力（ハッシュ）が異なります。

```
vehicle = {
            ‘VIN’ :       “KM8SRDHF6EU074761”,
            ‘Type’:       “Truck”
            ‘Model’:      “F150”
            ‘Specs’: {
                  ‘EngSize’ : 3.3
                  ‘CurbWeight’: 4,878
                  ‘HP’: 327
            }
}
```

SHA-256

```
a4e31e36910d99bd19b7f
875f0a04597dc0ff52c2f16
4a16a9288aed9e710fdd
```

```
vehicle = {
            ‘VIN’ :       “KM8SRDHF6EU074761”,
            ‘Type’:       “Truck”
            ‘Model’:      “F150”
            ‘Specs’: {
                  ‘EngSize’ : 3.3
                  ‘CurbWeight’: 4,879
                  ‘HP’: 327
            }
}
```

SHA-256

```
1931845740892 0af2d2cb
eacd90c7afe0fbd7f6ff316
972c8f656c8bbc402dd1
```

Cryptographic verifiability: SHA-256

SHA-256は一方通行です。出力が与えられた場合、入力を計算することは不可能です。

```
vehicle = {
            ‘VIN’ :       “KM8SRDHF6EU074761”,
            ‘Type’:       “Truck”
            ‘Model’:      “F150”
            ‘Specs’: {
                  ‘EngSize’ : 3.3
                  ‘CurbWeight’: 4,878
                  ‘HP’: 327
            }
}
```

SHA-256

a4e31e36910d99bd19b7f
875f0a04597dc0ff52c2f16
4a16a9288aed9e710fdd

SHA-256

19318457408920af2d2cb
eacd90c7afe0fbd7f6ff316
972c8f656c8bbc402dd1

How it works

```
INSERT INTO cars
    { 'Manufacturer':'Tesla',
      'Model':'Model S',
      'Year':2012,
      'VIN':123456789,
      'Owner':'Traci Russell' }
```

C

cars

ID	Manufacturer	Model	Year	VIN	Owner

H

history

ID	Version	Start	Manufacturer	Model	Year	VIN	Owner

journal

J

```
INSERT cars                    H (T1)
ID:1
Manufacturer: Tesla
Model: Model S
Year: 2012
VIN: 123456789
Owner: Traci Russell

Metadata: {
Date:07/16/2012
}
```

How it works

```
INSERT INTO cars
    { 'Manufacturer':'Tesla',
      'Model':'Model S',
      'Year':2012,
      'VIN':123456789,
      'Owner':'Traci Russell' }
```

C

cars

ID	Manufacturer	Model	Year	VIN	Owner
1	Tesla	Model S	2012	123456789	Traci Russell

H

history

ID	Version	Start	Manufacturer	Model	Year	VIN	Owner
1	0	7/16/2012	Tesla	Model S	2012	123456789	Traci Russell

journal

J

```
INSERT cars                    H (T1)
ID:1
Manufacturer: Tesla
Model: Model S
Year: 2012
VIN: 123456789
Owner: Traci Russell

Metadata: {
Date:07/16/2012
}
```

How it works

```
UPDATE cars SET owner = 'Ronnie Nash' WHERE
VIN = 123456789
```

C

cars

ID	Manufacturer	Model	Year	VIN	Owner
1	Tesla	Model S	2012	123456789	Ronnie Nash

H

history

ID	Version	Start	Manufacturer	Model	Year	VIN	Owner
1	0	7/16/2012	Tesla	Model S	2012	123456789	Traci Russell
1	1	8/03/2013	Tesla	Model S	2012	123456789	Ronnie Nash

journal

J

INSERT cars
ID:1
Manufacturer: Tesla
Model: Model S
Year: 2012
VIN: 123456789
Owner: Traci Russell

Metadata: {
Date:07/16/2012
}

H (T_1)

UPDATE cars
ID:1
Owner: Ronnie Nash

Metadata: {
Date:08/03/2013
}

H (T_2)

How it works

```
DELETE FROM cars WHERE VIN = 123456789
```

C

cars

ID	Manufacturer	Model	Year	VIN	Owner
1	Tesla	Model S	2012	123456789	Ronnie Nash

H

history

ID	Version	Start	Manufacturer	Model	Year	VIN	Owner
1	0	7/16/2012	Tesla	Model S	2012	123456789	Traci Russell
1	1	8/03/2013	Tesla	Model S	2012	123456789	Ronnie Nash
1	2	9/02/2016	*Deleted*				

journal

J

INSERT cars
ID:1
Manufacturer: Tesla
Model: Model S
Year: 2012
VIN: 123456789
Owner: Traci Russell

Metadata: {
Date:07/16/2012
}

H (T_1)

UPDATE cars
ID:1
Owner: Ronnie Nash

Metadata: {
Date:08/03/2013
}

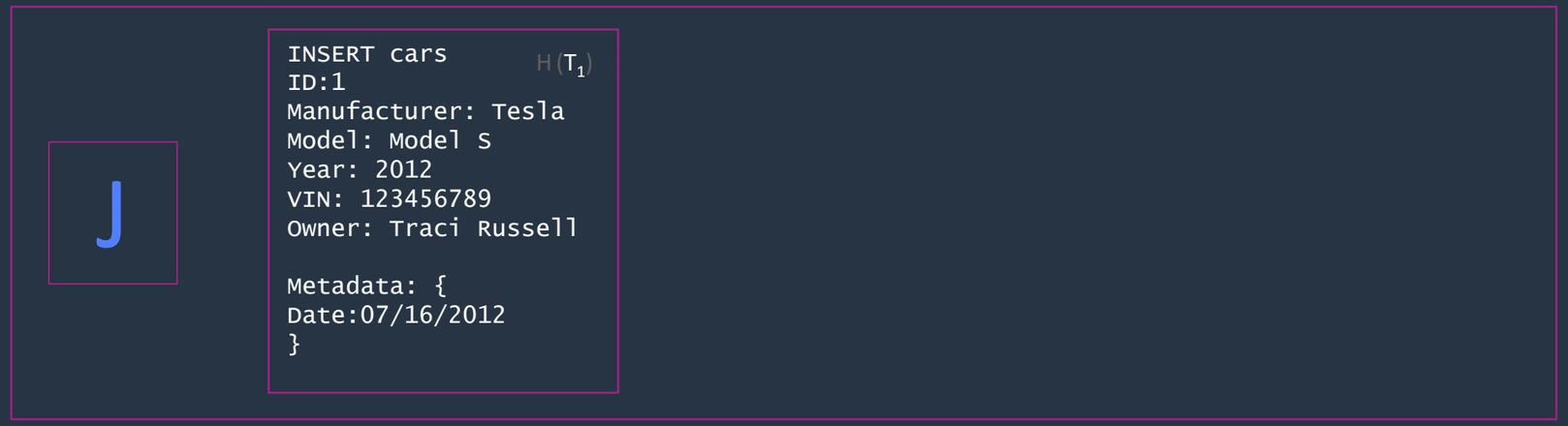
H (T_2)

DELETE cars
ID:1

Metadata: {
  Date: 09/02/2016
}

H (T_3)

Walk through a hash chain

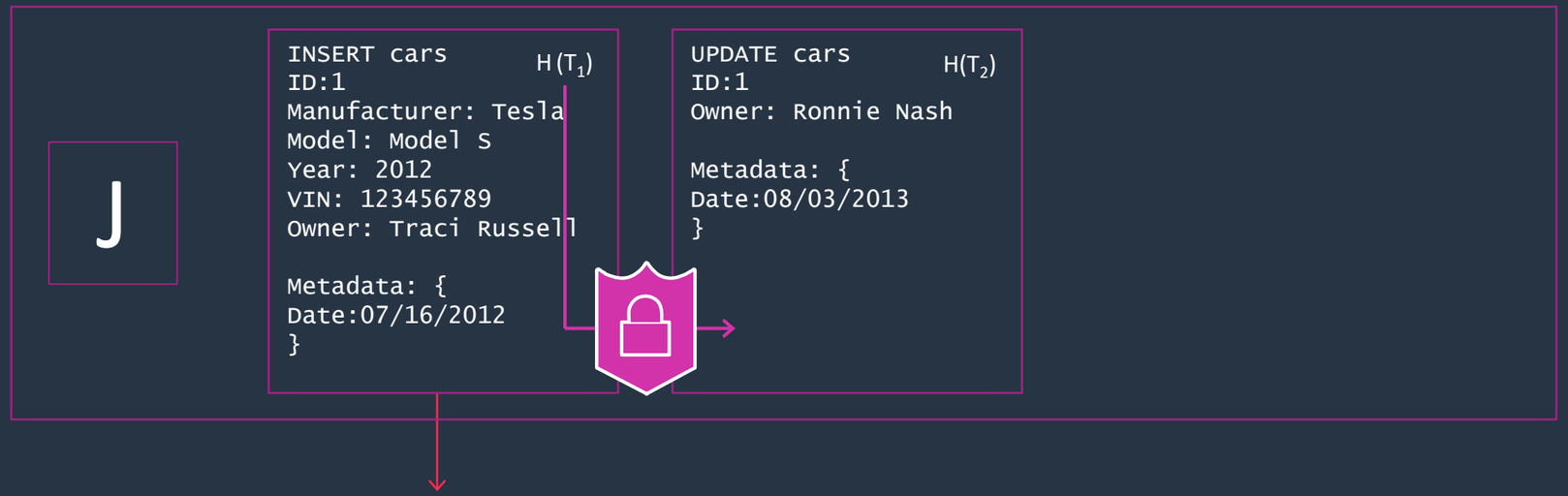



```
INSERT cars
ID:1
Manufacturer: Tesla
Model: Model S
Year: 2012
VIN: 123456789
Owner: Traci Russell

Metadata: {
Date:07/16/2012
}
```

SHA-256

$H(T_1)$ = 2526f16306c819d651af075934170d2430d246d9ab98d975d28a83baded47ca7

Hashing and chaining transactions



$H(T_1)$ = 2526f16306c819d651af075934170d2430d246d9ab98d975d28a83baded47ca7

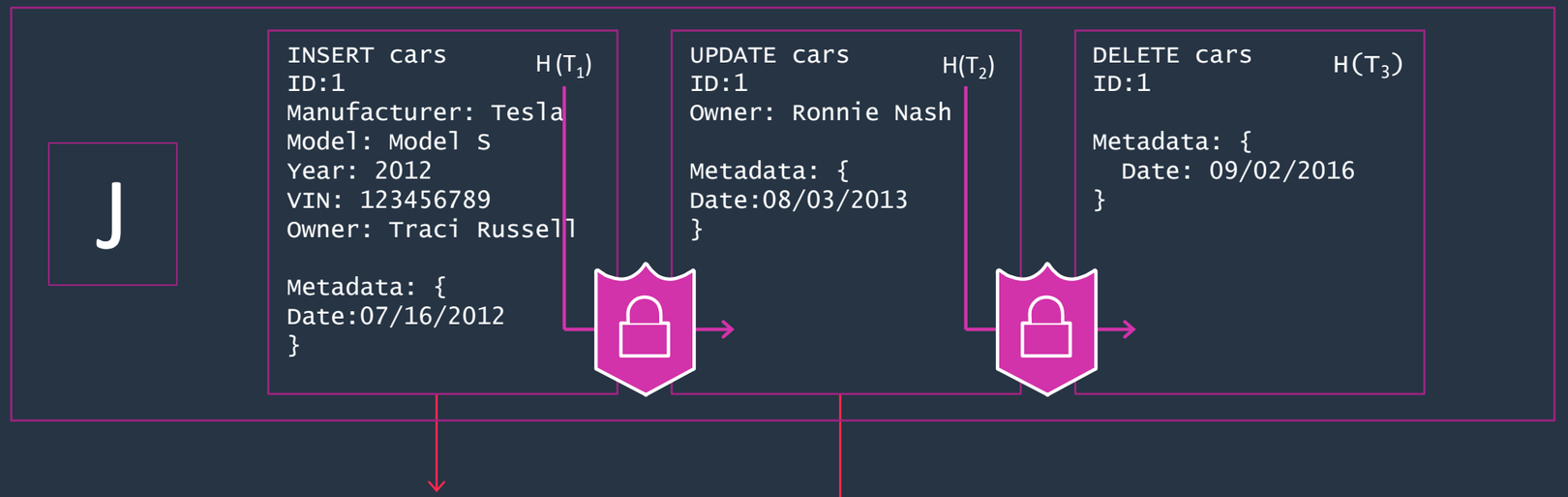
$H(T_1)$ +

```
UPDATE cars
ID:1
Owner: Ronnie Nash

Metadata: {
Date:08/03/2013
}
```

SHA-256

$H(T_2)$ = 86a90e4166453d9423b84d47dcbd97c0e3099b1a1f0d7cfca6c191d8fd8994ff

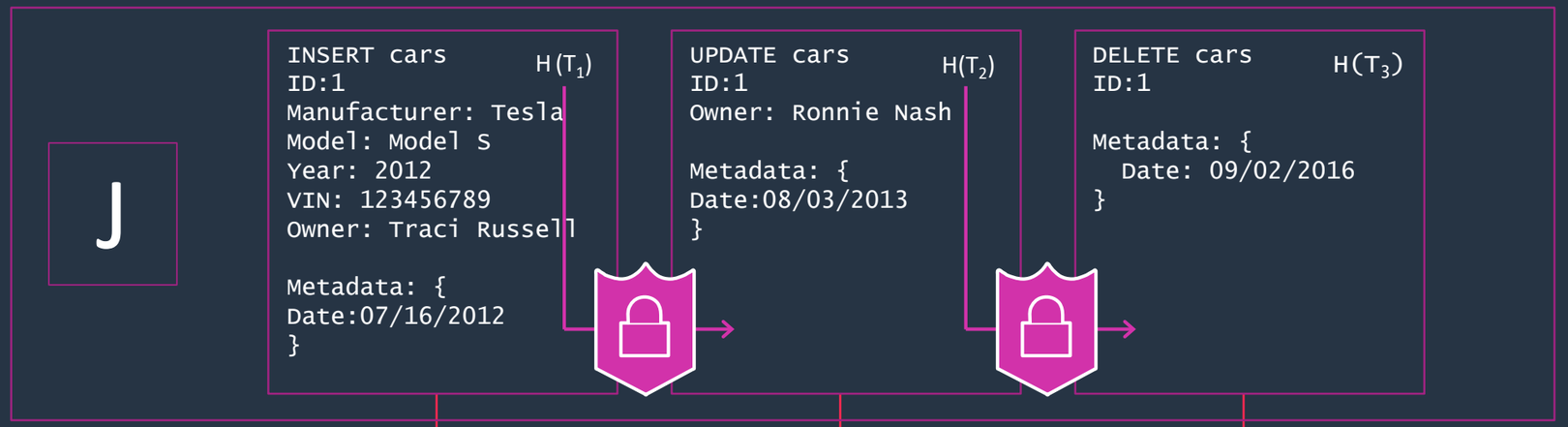
Hashing and chaining transactions



$H(T_1)$ = 2526f16306c819d651af075934170d2430d246d9ab98d975d28a83baded47ca7

$H(T_2)$ = 86a90e4166453d9423b84d47dcbd97c0e3099b1a1f0d7cfca6c191d8fd8994ff

Hashing and chaining transactions

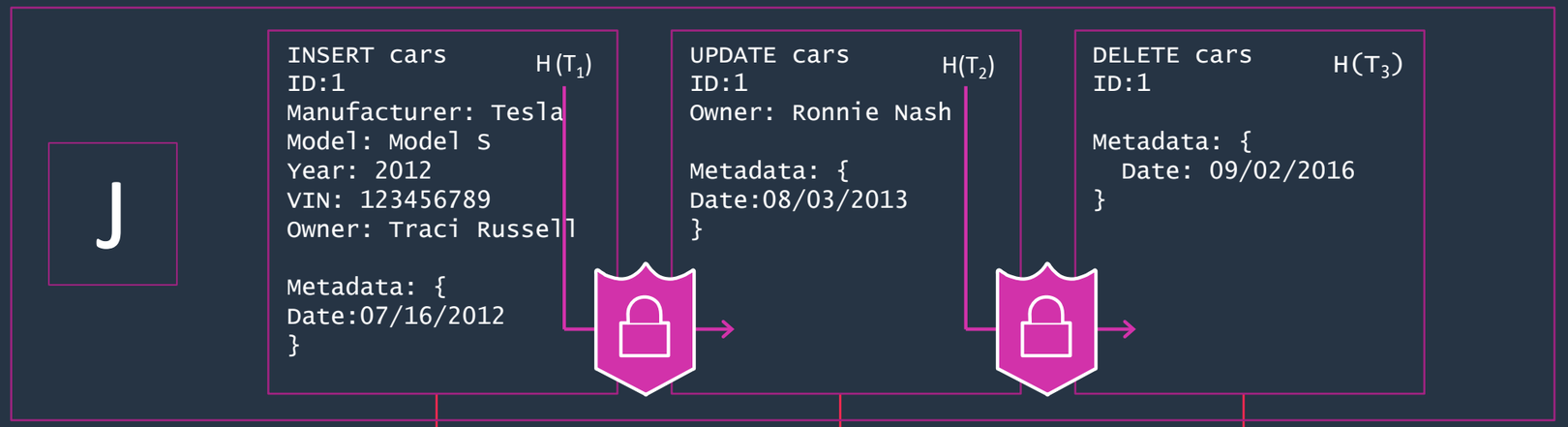


$H(T_1)$ = 2526f16306c819d651af075934170d2430d246d9ab98d975d28a83baded47ca7

$H(T_2)$ = 86a90e4166453d9423b84d47dcbd97c0e3099b1a1f0d7cfca6c191d8fd8994ff

$H(T_3)$ = ae2d64e562ec754ec3194c744eec72c9fdafffc6b559e0414d0e75bf96ca92ad

A digest is a hash value at a point in time

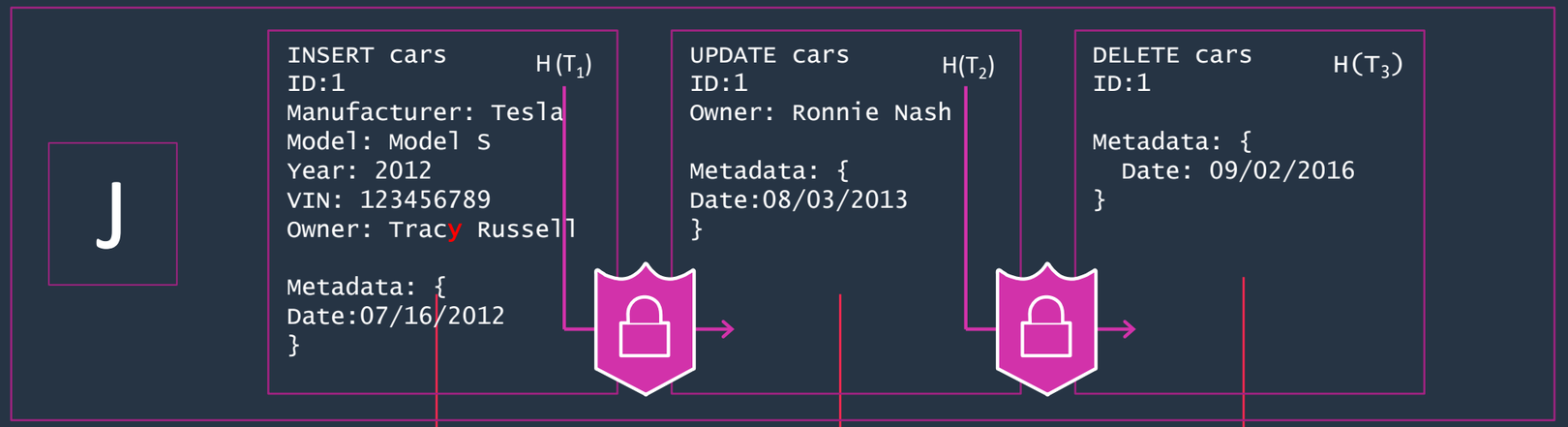


$H(T_1)$ = 2526f16306c819d651af075934170d2430d246d9ab98d975d28a83baded47ca7

$H(T_2)$ = 86a90e4166453d9423b84d47dcbd97c0e3099b1a1f0d7cfca6c191d8fd8994ff

$H(T_3)$ = ae2d64e562ec754ec3194c744eec72c9fdafffc6b559e0414d0e75bf96ca92ad

Changing committed data breaks the chain



$H(T_1)$ = ~~2526f16306c819d651af07593417Od2430d246d9ab98d975d28a83baded47ca7~~

$H(T_1)$ = 25d0b44e6e8878151646ffc1fea4eb85c3e4bf4baec212a9fcf67b6d5a81e01a

$H(T_2)$ = ~~86a90e4166453d9423b84d47dcbd97c0e3099b1a1f0d7cfca6c191d8fd8994ff~~

$H(T_2)$ = a90a9898c7e4b1aab19c705b554afd9e0bf6539bb0346df19be362ff63001098

$H(T_3)$ = ~~ac2d64e562ec754ec3194c744eec72c9fdafffc6b559e0414d0c75bf96ca92ad~~

$H(T_3)$ = c6268578a24dbe0c7cfba07bd967411a35462b8c875d42f1991faad02c0ac93c

Amazon QLDBの得意分野は？

Amazon QLDB is ideal for

AUDIT DATABASE

重要なビジネス情報への変更を、不変的かつ検証可能な記録として保存

イベントドリブン

アプリケーションやエンティティの状態変化を、不変のイベント・シーケンスとしてモデル化する。

トランザクション・システム・オブ・レコード

重要なビジネス情報の一貫した権威ある記録を保存する

ブロックチェーンの代替

一元化された権威モデルで、暗号化された検証可能な状態と変化を保存する

Amazon QLDB is not ideal for

ANALYTICAL DATABASE

Amazon QLDBは、多くのSQL分析関数をサポートしません。読み取りの多いワークロードは、Amazon QLDBのトランザクション・スケーリング・プロパティと競合する可能性があります。

NONTRANSACTIONAL WORKLOADS

Amazon QLDBは、デフォルトで、並行処理と強力な一貫性モデルの下で、高レベルのトランザクション分離を提供します；より低い分離レベルは、現在サポートされていません

Amazon QLDBでよくはまるところ

- Indexを作らない、ダメゼッタイ！
- QLDBの仕様を確認しないで使う
- 利用前にかならずドキュメントをご参照ください

QLDBにおける使用しない方が良いSQL

--述語句の無いSQL

- SELECT * FROM Kuwano

--COUNT()は最適化された関数ではありません

- SELECT COUNT(*) FROM Kuwano

--不等号（>）はインデックス検索の対象にならない

- SELECT * FROM Kuwano WHERE "Year" > 2018

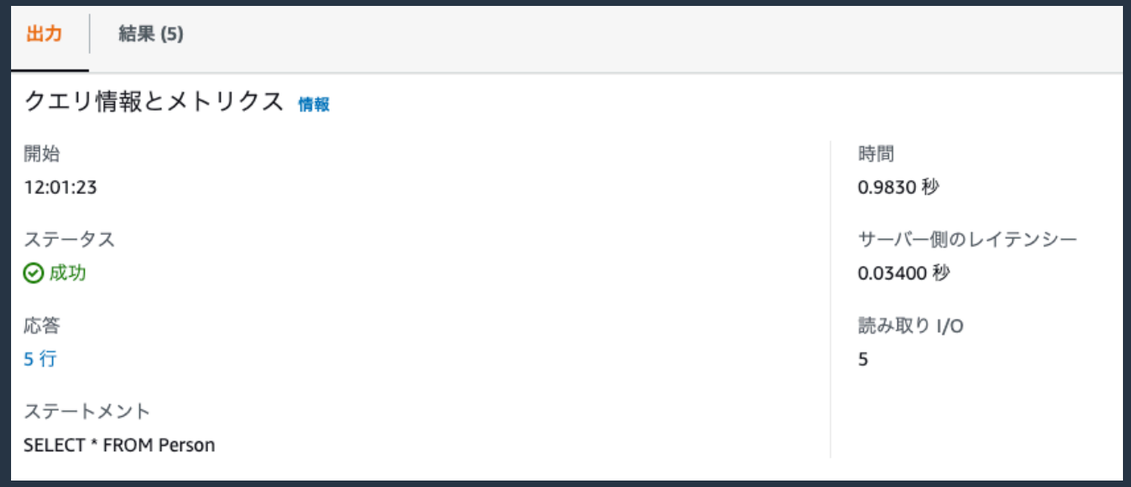
--Inequality (LIKE)

- SELECT * FROM Kuwano WHERE DB LIKE ‘Mongo%'

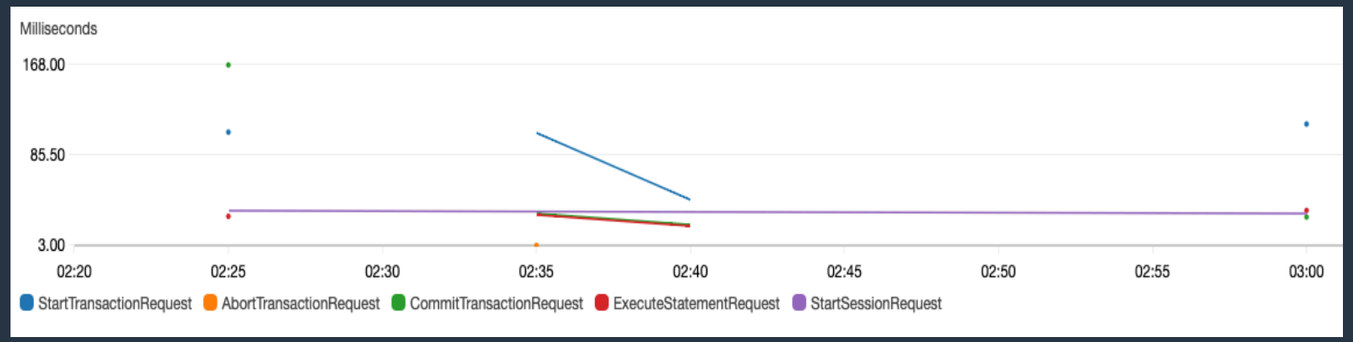
パフォーマンスの確認方法

- 現時点では、Explain のような機能は提供しておりませんmmmmm

I/O使用量



CloudWatch



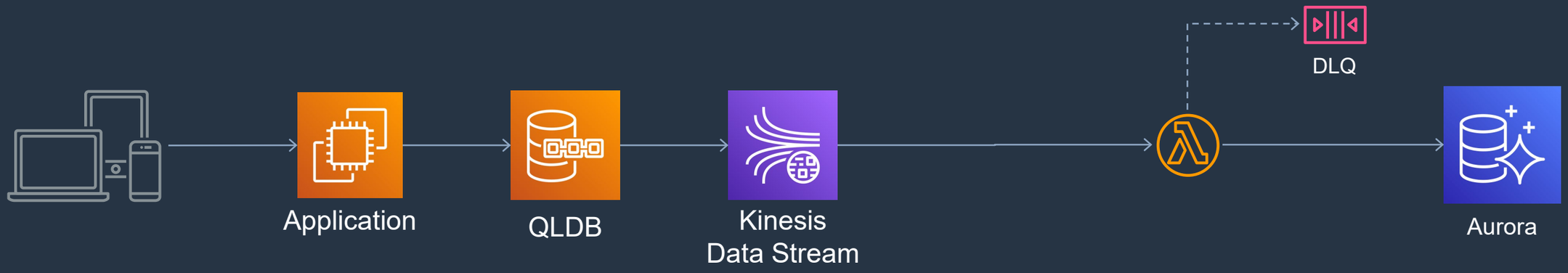
QLDB Streaming

Event processing (e.g., AWS Lambda triggers)

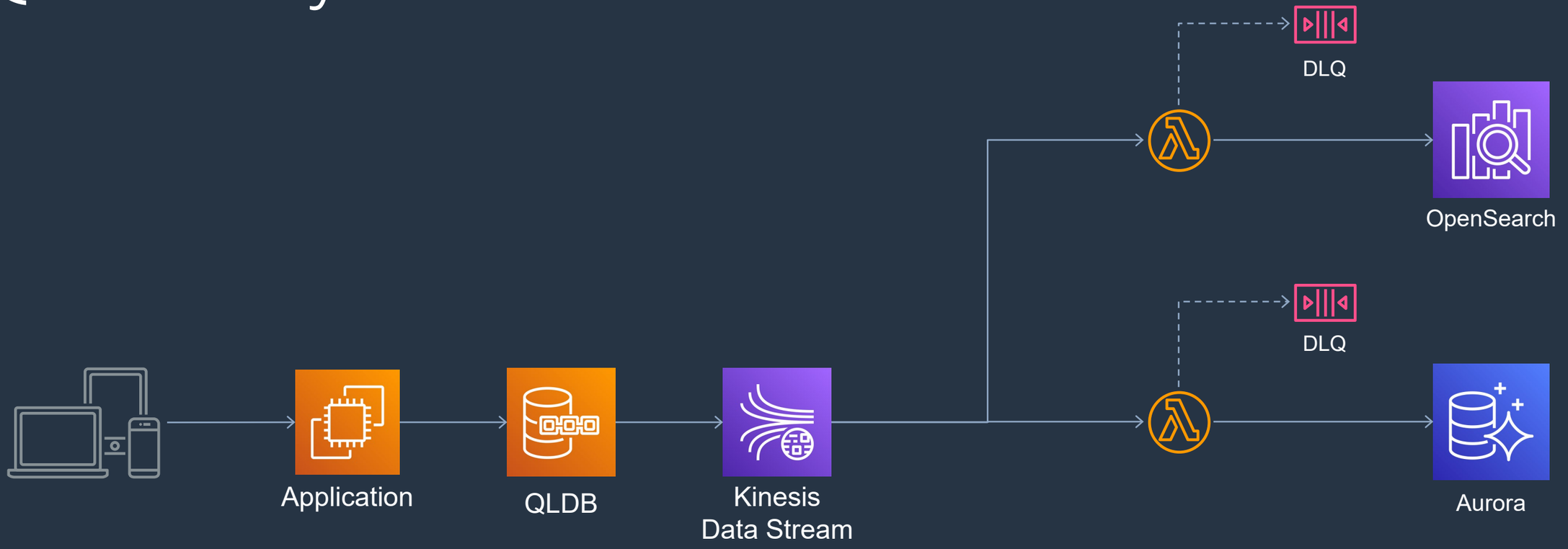
Real-time analytics (e.g. Amazon Athena, Amazon Kinesis Data Firehose)

Linkage to other purpose-built database (e.g. Amazon Elasticsearch Service (Amazon ES), Amazon Neptune)

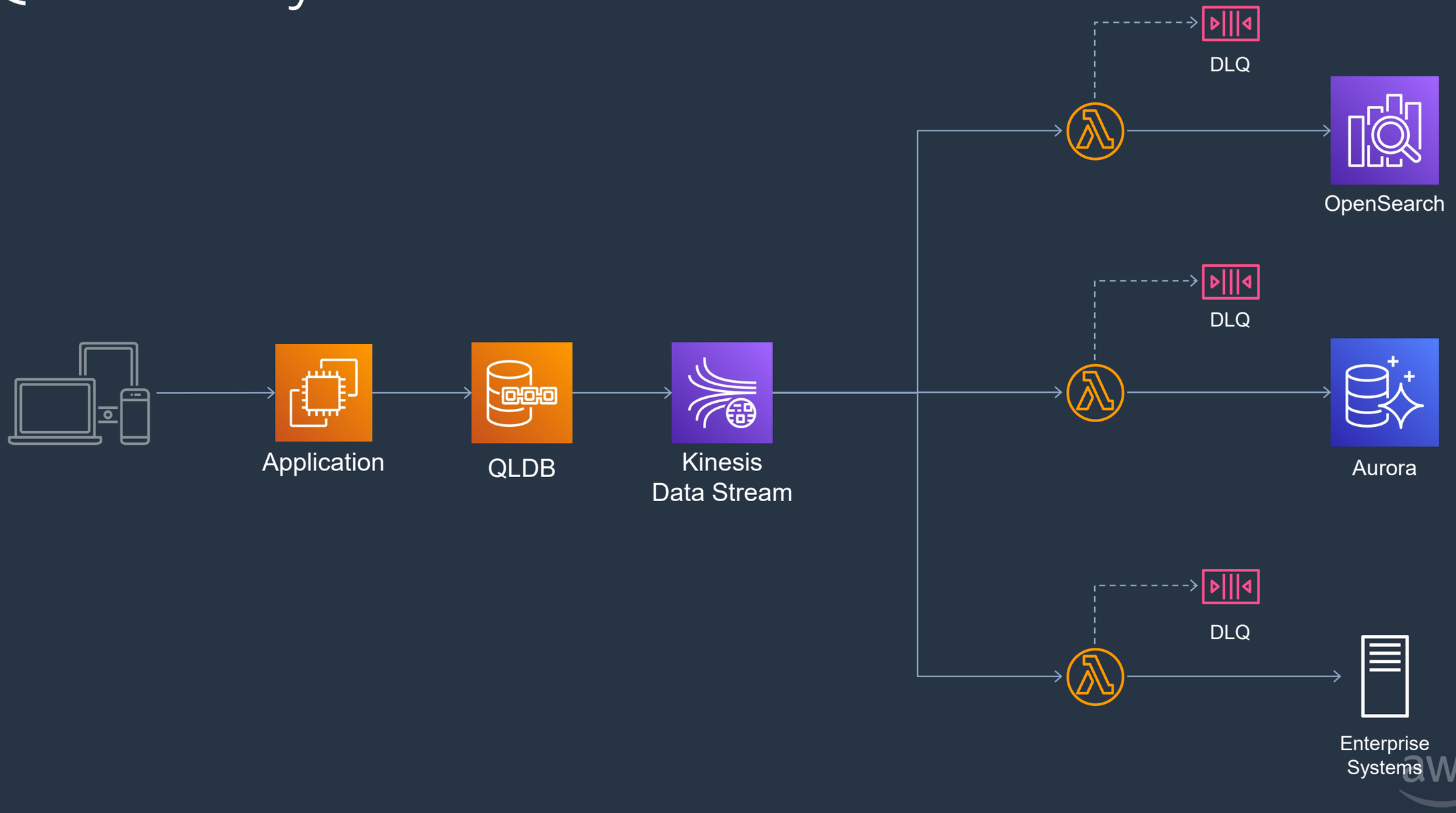
QLDB as System of Record



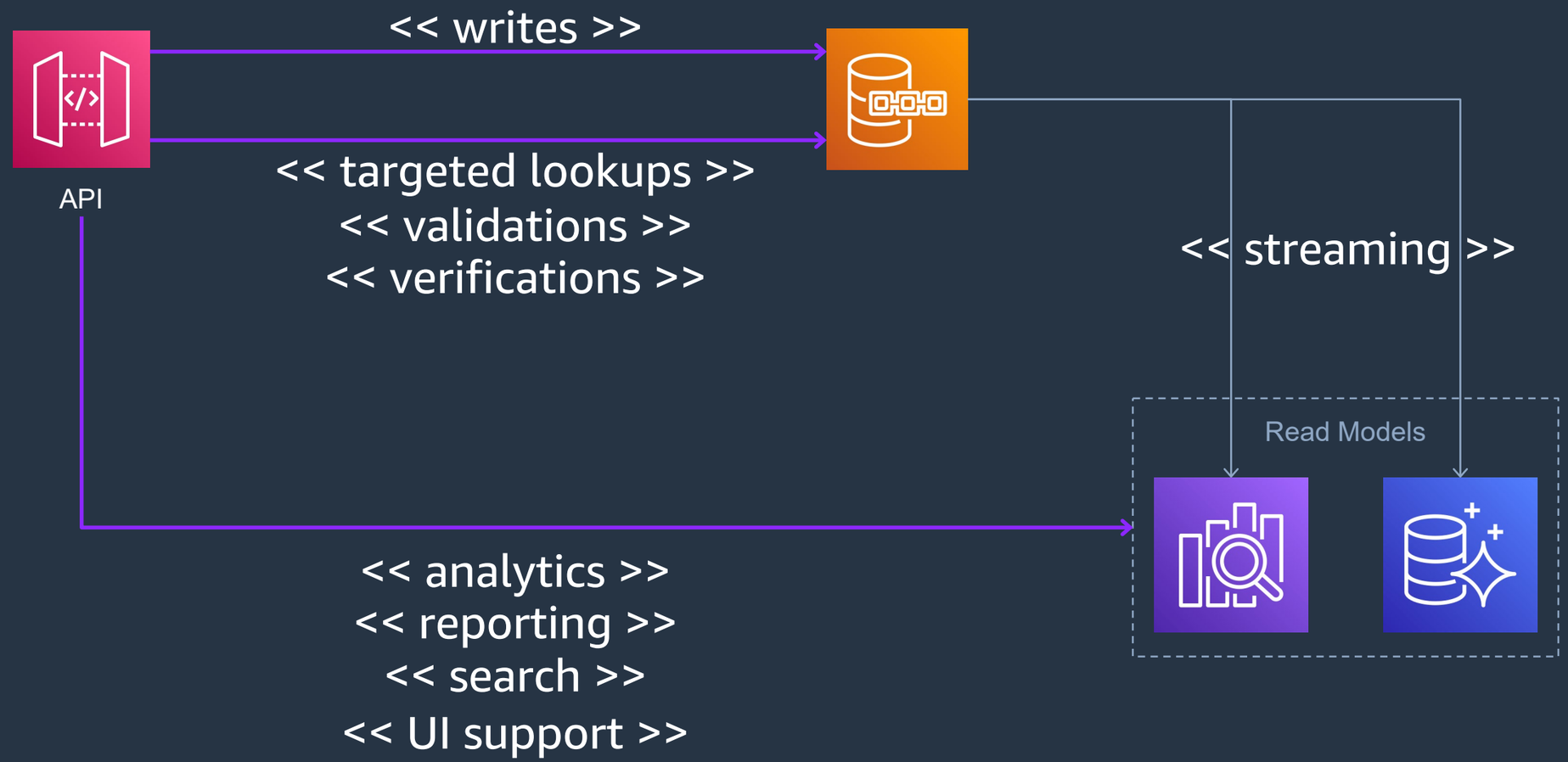
QLDB as System of Record



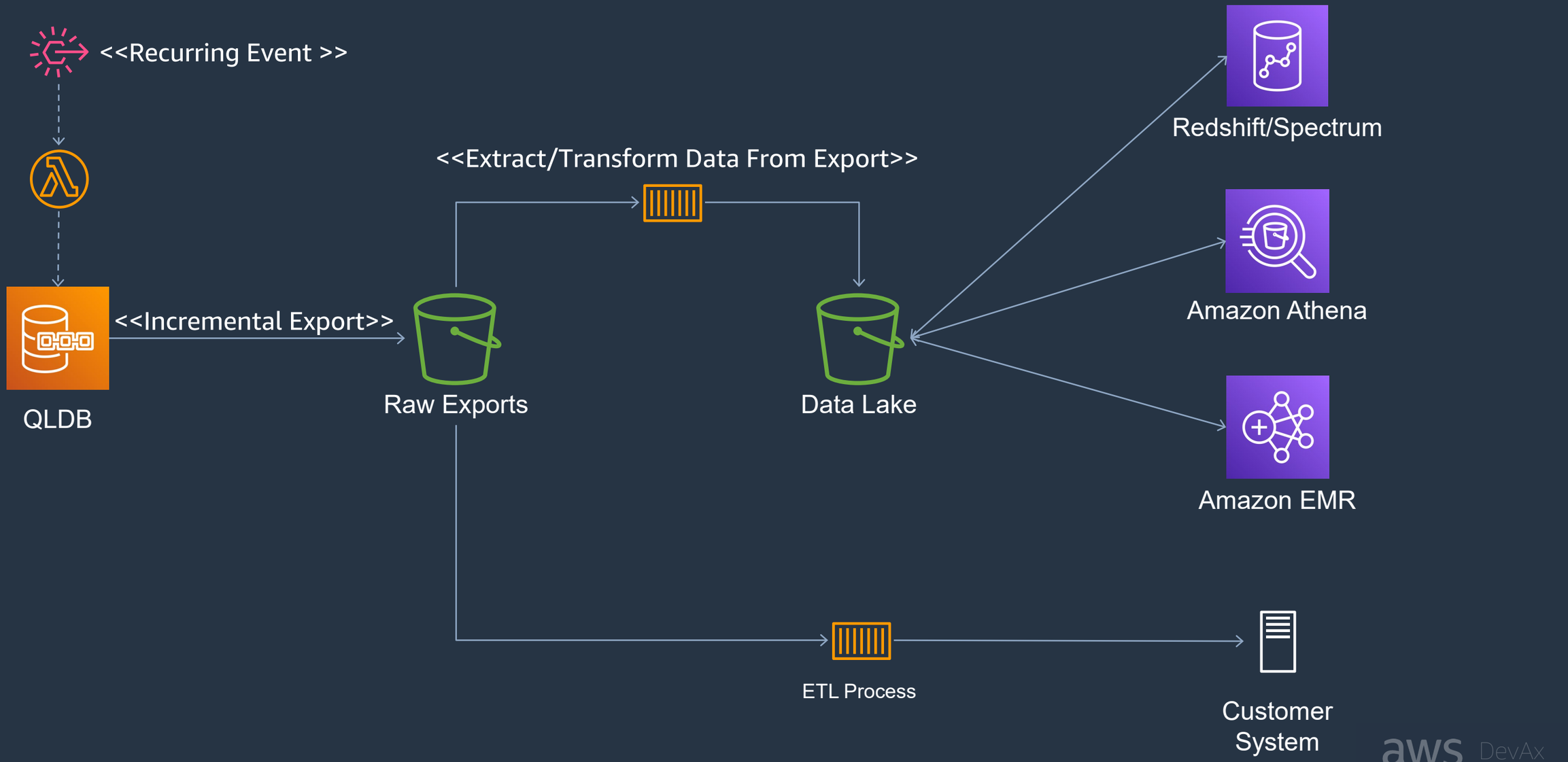
QLDB as System of Record



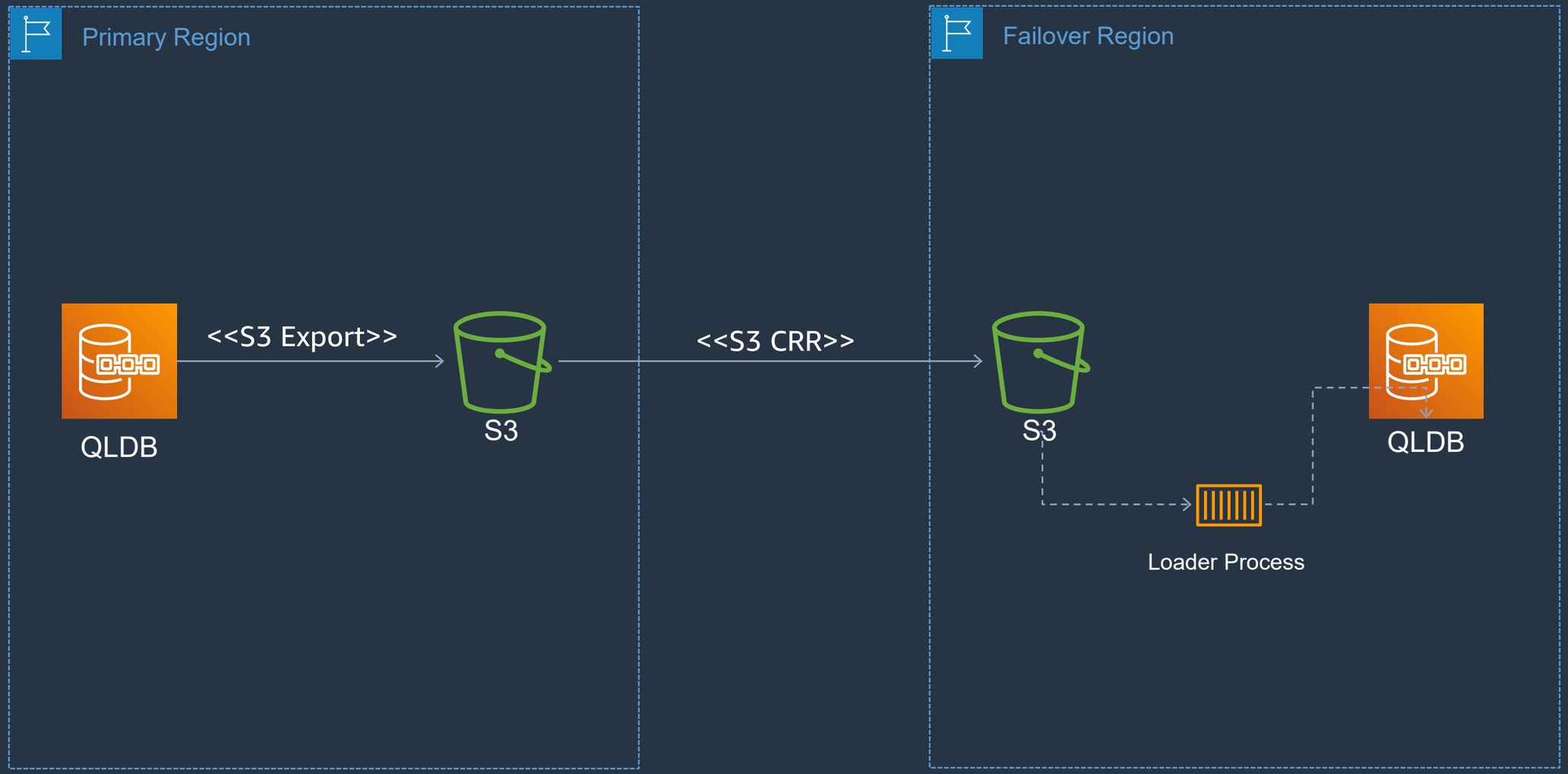
Query Patterns



QLDB Data Lake or Batch Integration



QLDB Disaster Recovery



サンプルコード

aws DevAx connect

製品テスト結果の想定

ID	Process	Lot	Operator	AppliedVoltage	Torque
1	1	1	20039	2	100
2	1	2	45387	2	120
3	1	3	99876	2	120
4	2.5	2.5	34547	2	150
5	2.5	2.5	98723	2	100
6	2.5	2.5	99876	2	120
7	2.5	2.5	34547	2	120
8	2.5	2.5	98723	2	150
9	2.5	2.5	20039	2	100
10	2.5	2.5	34547	2	120

QLDBを使った製品管理システム

https://github.com/aws-samples/amazon-qldb-product-management

ワークショップ

https://qldb-immersionday.workshop.aws/jp/

まとめ

- 台帳の管理は本当にそれでいいのか
- 中央集権 or 非中央集権
- QLDBの概要と使い所

Thank you!

Yuki Nakatake
Senor Solutions Architect

aws DevAx connect