AWS re:Invent

BLC209

Asset provenance ledger system based on Amazon QLDB: BMW's use case

Michael Labib
Principal SA, QLDB
Amazon Web Services

Emil Djerekarov
Technical Architect
BMW Group

AWS re:Invent

aws

Agenda

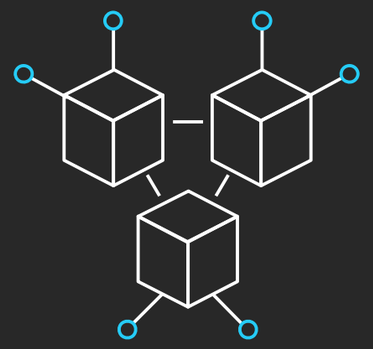
- Blockchain and purpose-built databases at AWS
- Amazon QLDB architecture and features
- BMW: Digital passport

What is blockchain

Ledgers

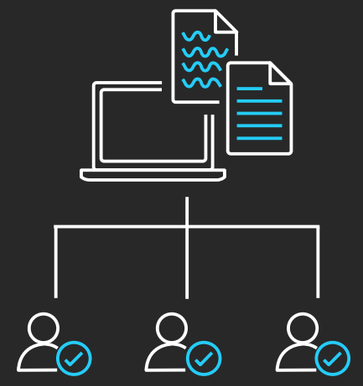
Immutable, append-only, cryptographically verifiable

Decentralization



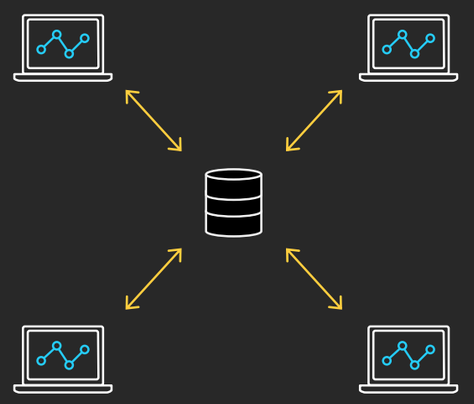
Distributed trust and data replication

Consensus algorithms



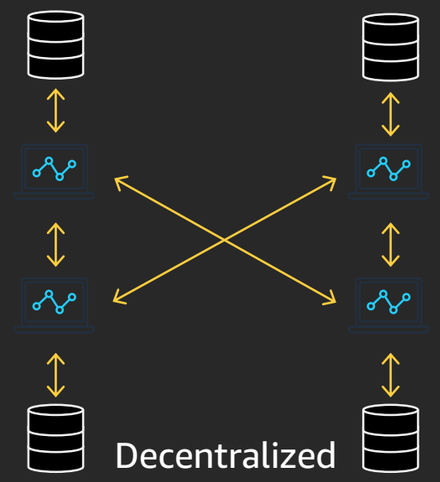
No intermediaries in decision process, support for smart contracts

Centralized versus decentralized



Centralized

- Owned by a single, trusted authority
- Addresses core need of an immutable and verifiable transactional log
- Fast—doesn't require consent from members to commit transactions



Decentralized

- No single owner of the ledger; joint ownership by multiple parties
- Addresses core need of enabling multiple parties to interact transparently with trust
- Removes intermediaries when a group of members needs to transact; can make business processes more efficient

Purpose-built databases at AWS

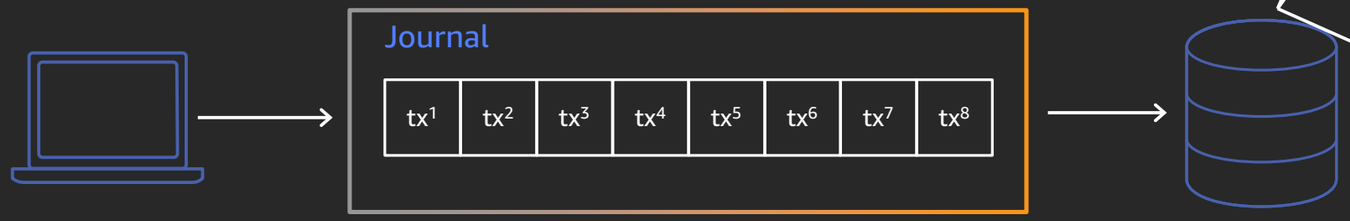
Relational	Key value	Document	In memory	Graph	Time series	Ledger
Referential integrity, ACID transactions, schema-on-write	High throughput, low-latency reads and writes, endless scale	Store documents and quickly access querying on any attribute	Query by key with microsecond latency	Quickly and easily create and navigate relationships between data	Collect, store, and process data sequenced by time	Complete, immutable, and verifiable history of all changes to application data
Lift and shift, ERP, CRM, finance	Real-time bidding, shopping cart, social, product catalog, customer preferences	Content management, personalization, mobile	Leaderboards, real-time analytics, caching	Fraud detection, social networking, recommendation engine	AWS IoT applications, event tracking	Systems of record, supply chain, healthcare, registrations, financial

Traditional database architecture: The log

- Typically an internal implementation
- Used for replicating data
- Difficult—or impossible—to directly access

Amazon QLDB: The journal is the database

- QLDB's journal has structural similarity to a database log
- All writes go to the journal—the journal determines state
- Journal handles concurrency, sequencing, cryptographic verifiability, and availability
- Accessible history of all transactions, document versions, document metadata



User #standard user data, the default

*select * from cars*

Manufacturer	Model	Year	VIN	Owner
Tesla	Model S	2012	123456789	Richard Roe

Committed #includes metadata

*select * from _ql_committed_cars*

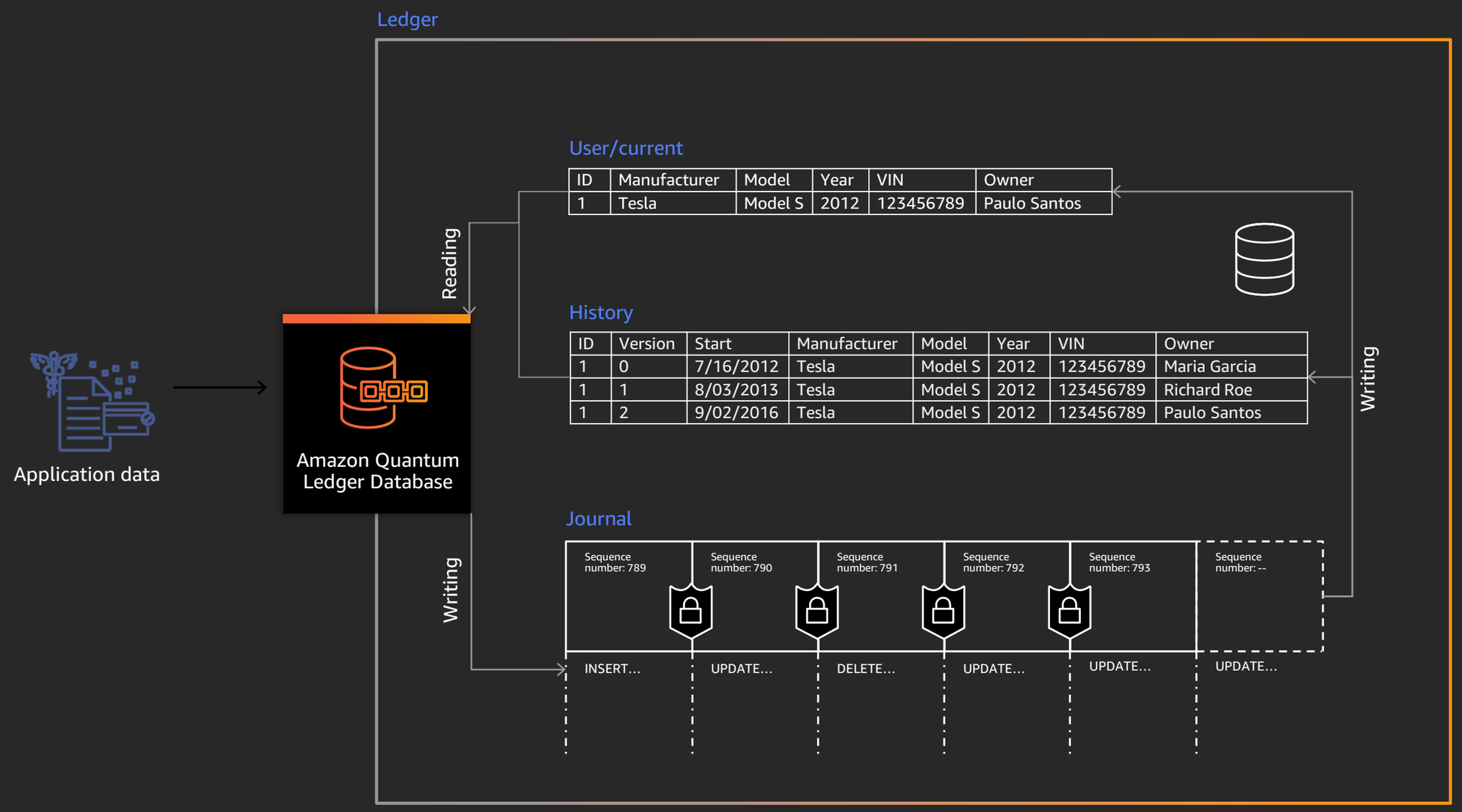
blockAddress	hash	data	metadata
{strandId:"IiPT4AnNeWpE6YdoFiAK0U",sequenceNo:139}	{{1fWeNgIm59pgnkdBol83exdL+t/BAvVfICrX8N9cYN4=}}	{Manufacturer:"Tesla",Model:"Model S",Year:2012,VIN:123456789,Owner:"Richard Roe"}	{id:"Cxlp6cA36YOHylSChJ7dzW",version:1,txTime:2019-10-28T12:20:25.979Z,txId:"1nmeGu0dy4nI3KHcsfC4tr"}

history() #function to query document history

*SELECT * FROM history(cars) WHERE:*

blockAddress	hash	data	metadata
{strandId:"IiPT4AnNeWpE6YdoFiAK0U",sequenceNo:136}	{{s6ytAZivsX2ukJNSwBVvYZ5KjnFQRJHRDrU0UKbYVRY=}}	{Manufacturer:"Tesla",Model:"Model S",Year:2012,VIN:123456789,Owner:"Maria Garcia"}	{id:"Cxlp6cA36YOHylSChJ7dzW",version:0,txTime:2019-10-28T12:19:50.098Z,txId:"JUJglL39zof3a0LKTXNRgX"}
{strandId:"IiPT4AnNeWpE6YdoFiAK0U",sequenceNo:139}	{{1fWeNgIm59pgnkdBol83exdL+t/BAvVfICrX8N9cYN4=}}	{Manufacturer:"Tesla",Model:"Model S",Year:2012,VIN:123456789,Owner:"Richard Roe"}	{id:"Cxlp6cA36YOHylSChJ7dzW",version:1,txTime:2019-10-28T12:20:25.979Z,txId:"1nmeGu0dy4nI3KHcsfC4tr"}

Amazon QLDB—Journal-first database



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

- Amazon QLDB supports the highest level of isolation
- There is no other mode for QLDB
- There is no risk that you'll see phantom reads, write skew, dirty reads, or other issues

Easy to use—Amazon Ion & PartiQL

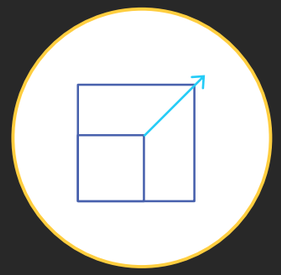
Amazon Ion

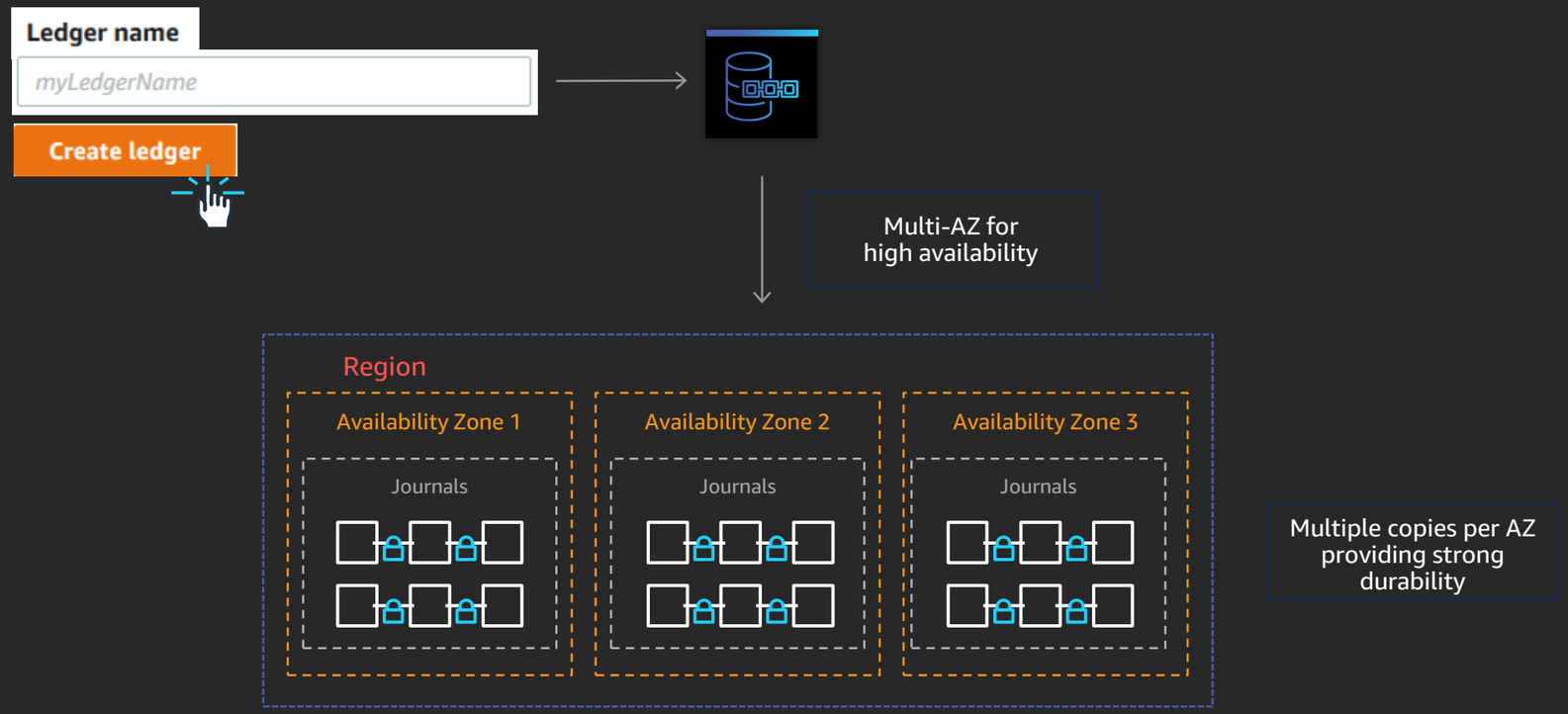
```
/* Ion supports comments.
This is a “Vehicle” document */
    {
    ‘VIN’ : ‘KM8SRDHF6EU074761’ ,
    ‘MfgDate’:  `2017-03-01T` ,
    ‘Type’:    ‘Truck’ ,
    ‘Mfgr’:    ‘Ford’ ,
    ‘Model’:   ‘F150’
    ‘Color”:    ‘Black’ ,
    ‘Specs’: {
                ‘EngSize’ : 3.3 ,(decimal)
           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

PartiQL

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

UPDATE cars SET owner = 'Richard Roe'

WHERE VIN = 123456789

SELECT * FROM cars
```

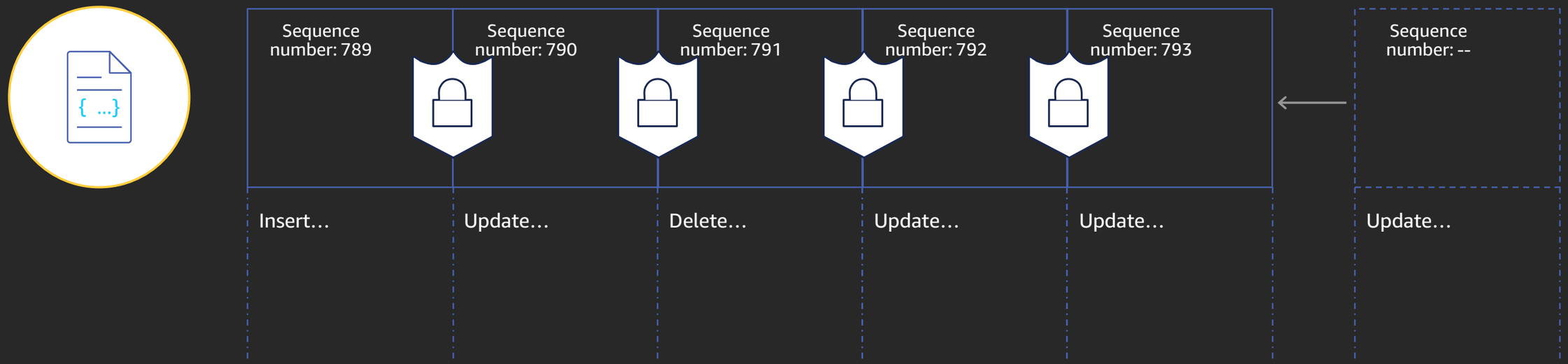
Serverless, scalable, highly available





Immutable

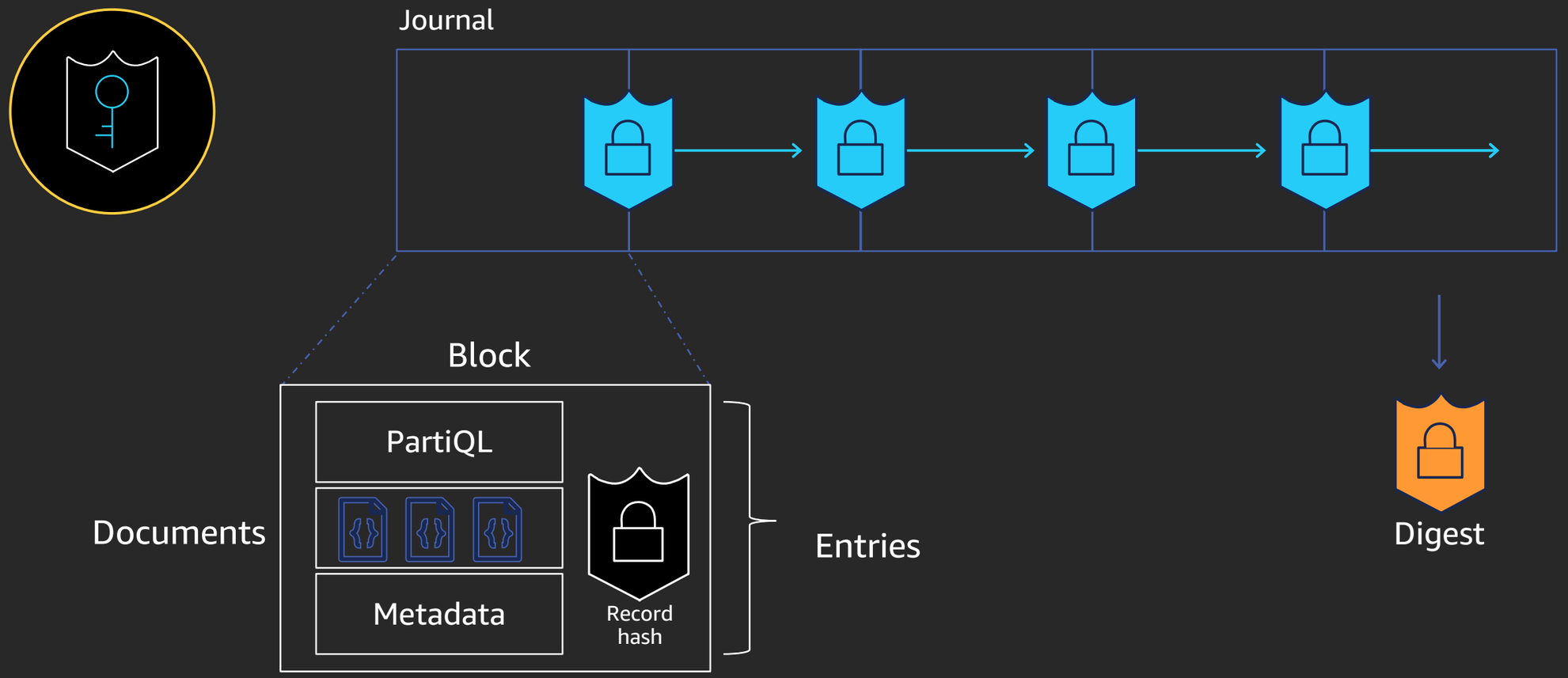
Records cannot be altered



- The journal is append only and sequenced
- There is no API or other method to alter committed data
- All operations, including deletes, are written to the journal

Cryptographic verification

Hash chaining using sha-256



Amazon QLDB summary

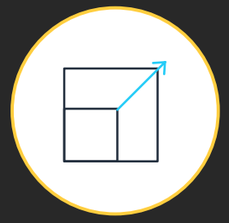
Immutable

Append only, sequenced

Cryptographically verifiable

Hash chaining provides data integrity

Highly scalable



Serverless, highly available

Easy to use

Familiar SQL operators

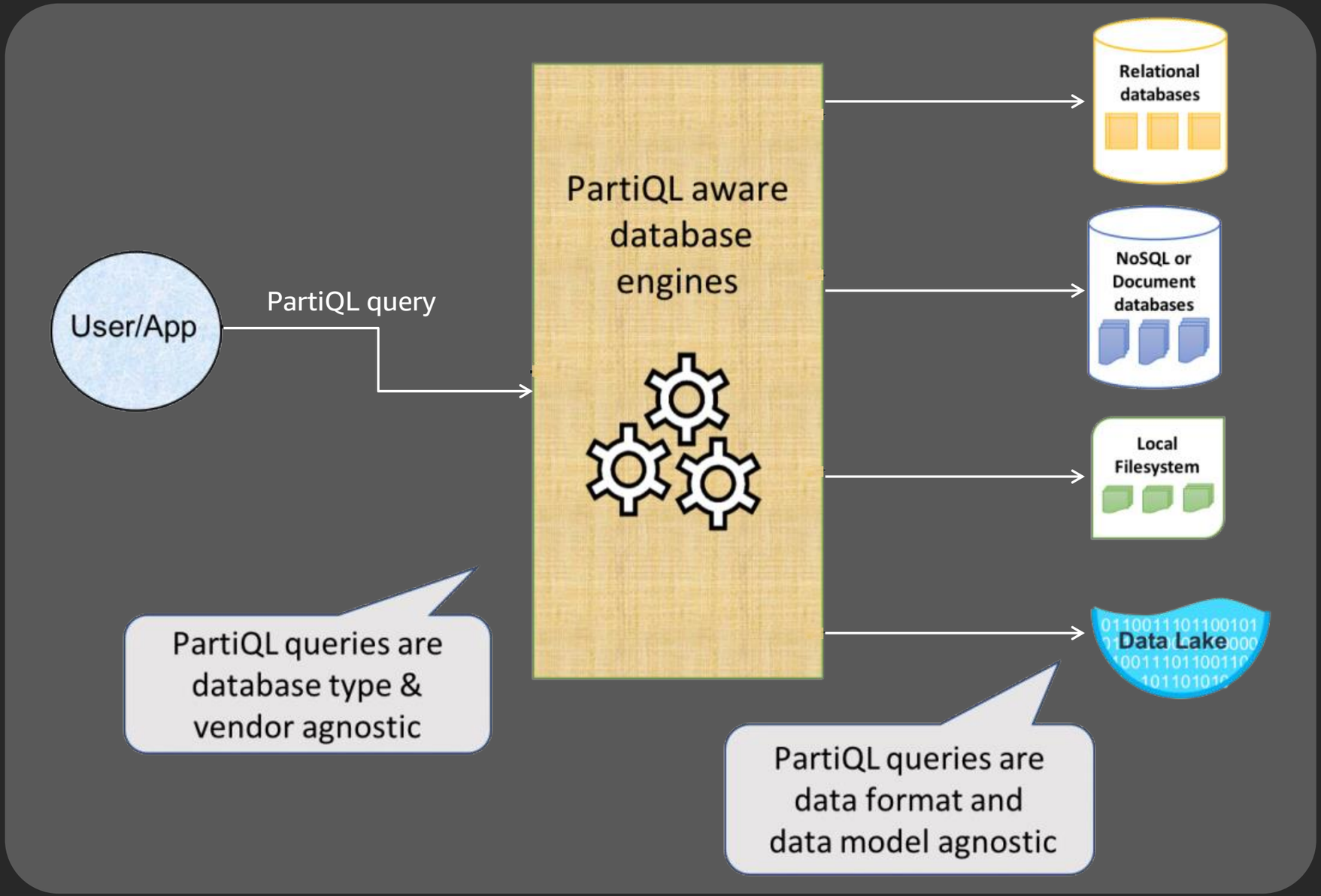
ACID transactions

Fully serializable isolation

Journal first

The journal is the database

PartiQL—one query language for all your data



PartiQL—fundamental design tenets

1. SQL compatibility
2. First-class nested data
3. Minimal extensions
4. Format independence
5. Data store independence

Basic PartiQL query

Input data:

```
{
  VIN: "1N4AL11D75C109151",
  LicensePlateNumber: "LEWISR261LL",
  State: "WA",
  City: "Seattle",
  PendingPenaltyTicketAmount: 90.25,
  ValidFromDate: 2017-08-21T,
  ValidToDate: 2020-05-11T,
  Owners: {
    PrimaryOwner: {
      PersonId: "C6XZswBSmTS0myD5MW34B9"
    },
    SecondaryOwners: [
      {PersonId:"5Ufgdlnj06gF5CWcOIu64s"},
      {PersonId:"HrBDqUrWTHD3lN12CJ605q"}
    ]
  }
}
```

Query:

```
SELECT
  s.VIN AS VIN
FROM
  VehicleRegistration AS s
WHERE
  s.City = 'Seattle'
```

Output:

```
{
  VIN: "1N4AL11D75C109151"
}
```

Nested collections

Input data:

```
{
  VIN: "1N4AL11D75C109151",
  LicensePlateNumber: "LEWISR261LL",
  State: "WA",
  City: "Seattle",
  PendingPenaltyTicketAmount: 90.25,
  ValidFromDate: 2017-08-21T,
  ValidToDate: 2020-05-11T,
  Owners: {
    PrimaryOwner: {
      PersonId: "C6XZswBSmTS0myD5MW34B9"
    },
    SecondaryOwners: [
      {PersonId:"5Ufgdlnj06gF5CWcOIu64s"},
      {PersonId:"HrBDqUrWTHD3lN12CJ605q"}
    ]
  }
}
```

Query:

```
SELECT
  s.VIN AS VIN,
  p.PersonId as PersonId
FROM
  VehicleRegistration AS s,
  s.Owners AS o,
  o.SecondaryOwners as p
WHERE
  p.PersonId = '5Ufgdlnj06gF5CWcOIu64s'
```

Output:

```
{
  VIN: "1N4AL11D75C109151",
  PersonId: "5Ufgdlnj06gF5CWcOIu64s"
}
```

Ion features

1. Value based
2. Supports primitive types and container types
3. Flexible and scalable
4. Hierarchy
5. Human readable
6. Efficient for machine

Ion types

Primitive types

bool, int, float, decimal, timestamp, string, symbol, blob, clob, null

Collection types

struct, list

- SQL Scalar types are covered by their Ion counterparts
- Ion's struct type is equivalent to an SQL tuple
- In addition to NULL, PartiQL has a MISSING type, meaning missing field

JSON versus Amazon Ion

JSON document

```
{
      “VIN” : “KM8SRDHF6EU074761”,
      “MfgDate” : “2017-03-01”,
      “Type : “Truck”,
      “Mfgr” : “Ford”,
      “Model” : “F150”,
      “Color” : “Black”,
      “Specs” : {
            “EngSize” : 3.3,
            “CurbWeight”: 4878,
            “HP”: 327,
            “BatterySize”: null
      }
}
```

Amazon Ion document

```
/* Ion supports comments. */
{
      VIN : “KM8SRDHF6EU074761”, // (String)
      MfgDate : 2017-03-01T, // (timestamp)
      Type : “Truck”,
      Mfgr : “Ford”,
      Model : “F150”,
      Color : “Black”,
      Specs: {
                EngSize : 3.3      // (decimal)
                CurbWeight : 4878 // (int)
                HP : 327           // (int)
                BatterySize : null.int
            }
}
```

https://github.com/amzn/ion-java

Mapping constructs between RDBMS & QLDB

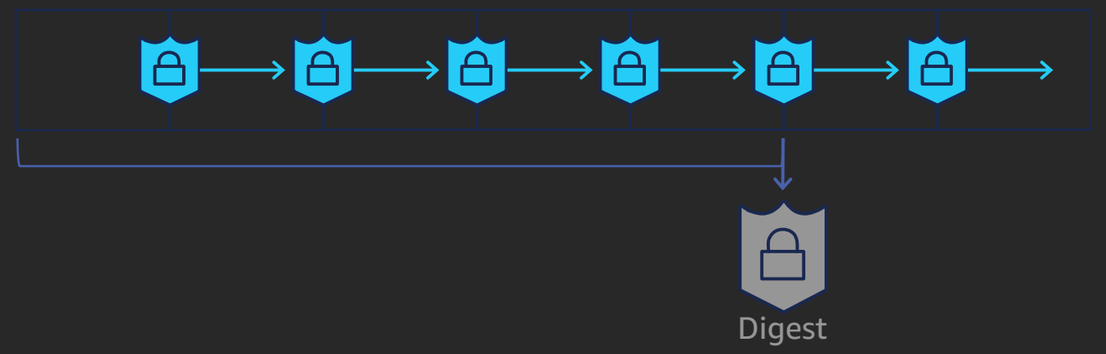
Relational	QLDB
Database	Ledger
Table	Table
Index	Index
Table row	Amazon Ion document
Column	Document attribute
SQL	PartiQL
Audit logs	Journal

Cryptographic verifiability

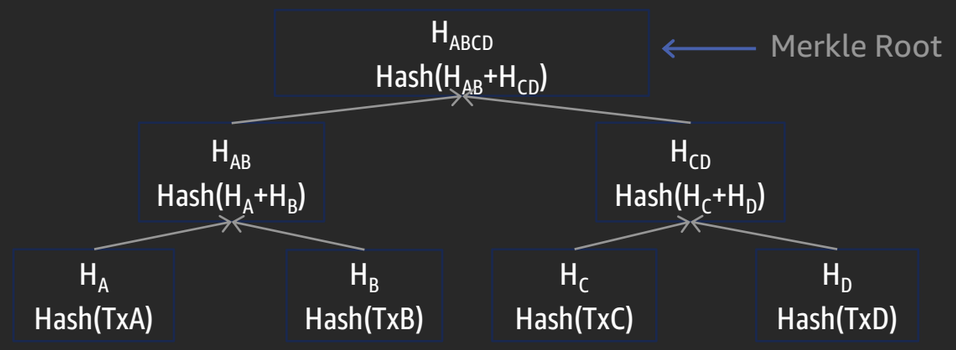
Four basic steps to seeing how QLDB's verifiability works

SHA256: Unique signature of a document

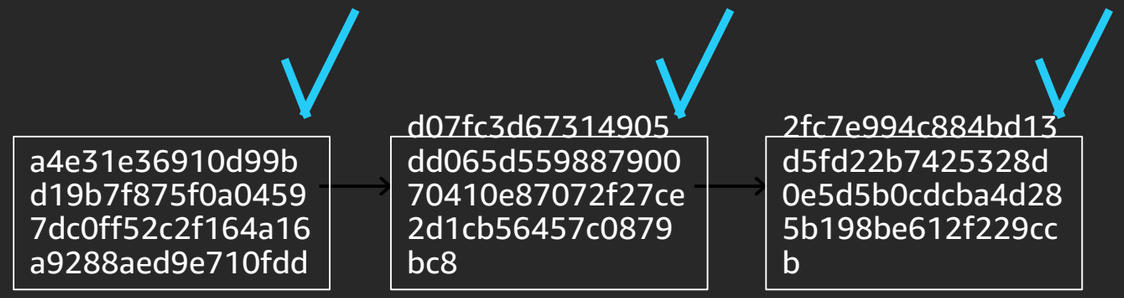
Digest: Periodic hash covering all history



Merkle trees: Chaining past hashes together



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



How it works

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

Cars

C

ID	Manufacturer	Model	Year	VIN	Owner

History

H

ID	Version	Start	Manufacturer	Model	Year	VIN	Owner

Journal

J

```
INSERT cars
ID:1
Manufacturer: Tesla
Model: Model S
Year: 2012
VIN: 123456789
Owner: Maria Garcia

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

$H(T_1)$

How it works

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

Cars

C

ID	Manufacturer	Model	Year	VIN	Owner
1	Tesla	Model S	2012	123456789	Maria Garcia

History

H

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

Journal

J

```
INSERT cars
ID:1
Manufacturer: Tesla
Model: Model S
Year: 2012
VIN: 123456789
Owner: Maria Garcia

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

$H(T_1)$

How it works

```
UPDATE cars SET owner = 'Richard Roe' WHERE
VIN = 123456789
```

C

Cars

ID	Manufacturer	Model	Year	VIN	Owner
1	Tesla	Model S	2012	123456789	Richard Roe

H

History

ID	Version	Start	Manufacturer	Model	Year	VIN	Owner
1	0	7/16/2012	Tesla	Model S	2012	123456789	Maria Garcia
1	1	8/03/2013	Tesla	Model S	2012	123456789	Richard Roe

Journal

J

```
INSERT cars
ID:1
Manufacturer: Tesla
Model: Model S
Year: 2012
VIN: 123456789
Owner: Maria Garcia

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

$H(T_1)$

```
UPDATE cars
ID:1
Owner: Richard Roe

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	Richard Roe

H

History

ID	Version	Start	Manufacturer	Model	Year	VIN	Owner
1	0	7/16/2012	Tesla	Model S	2012	123456789	Maria Garcia
1	1	8/03/2013	Tesla	Model S	2012	123456789	Richard Roe
1	2	9/02/2016	*Deleted*				

Journal

J

```
INSERT cars
ID:1
Manufacturer: Tesla
Model: Model S
Year: 2012
VIN: 123456789
Owner: Maria Garcia

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

$H(T_1)$

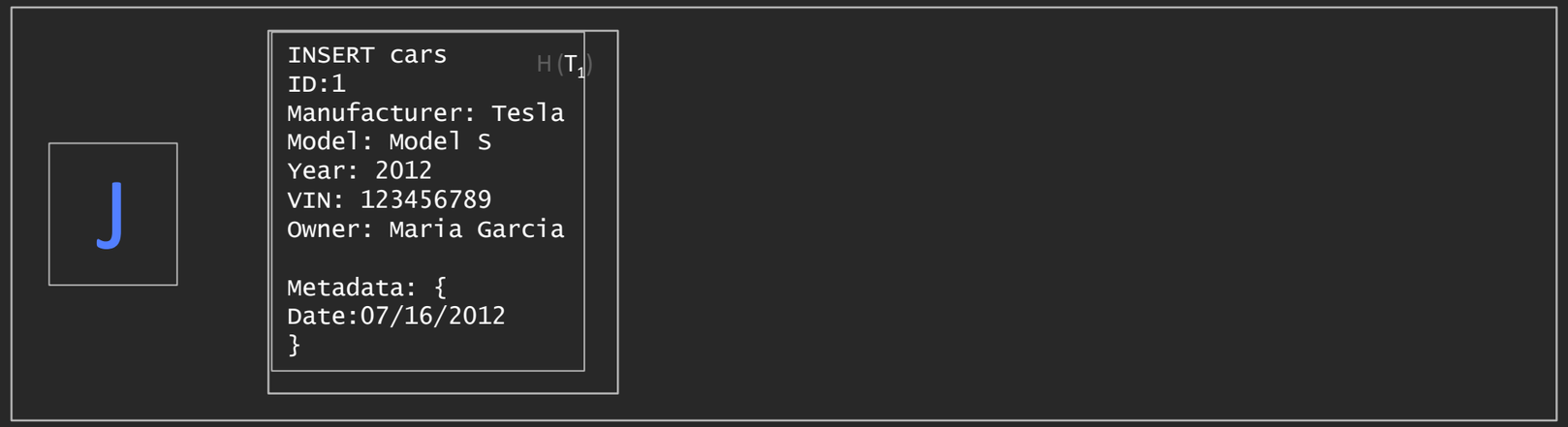
```
UPDATE cars
ID:1
Owner: Richard Roe

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

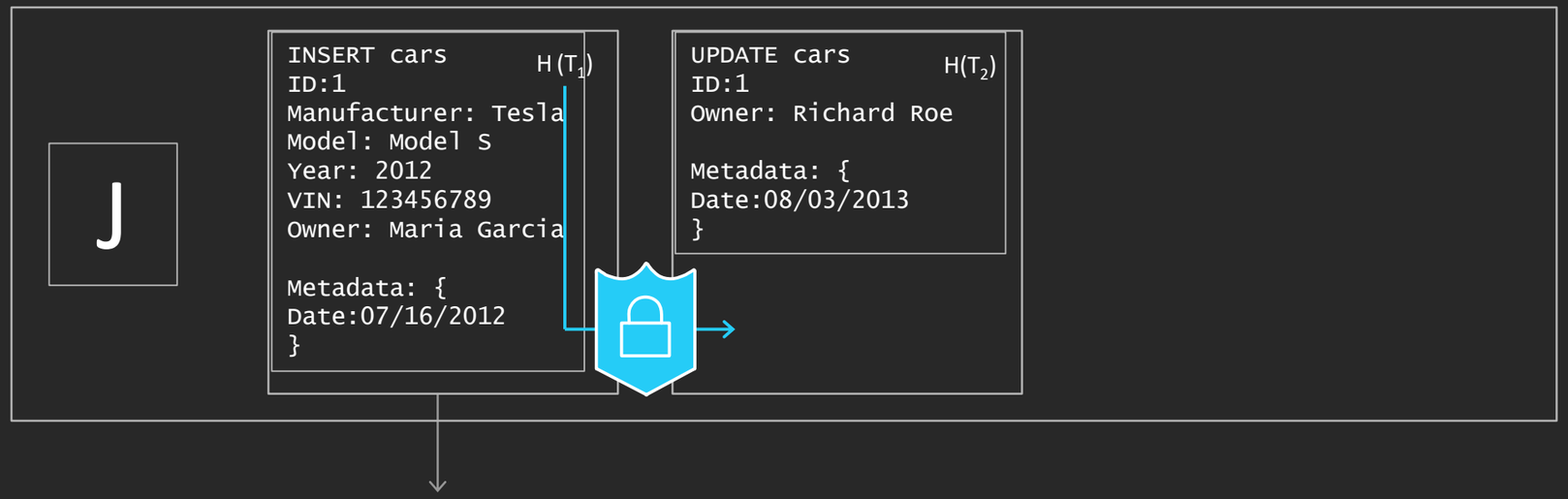



```
INSERT cars
ID:1
Manufacturer: Tesla
Model: Model S
Year: 2012
VIN: 123456789
Owner: Maria Garcia

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

SHA-256

$H(T_1)$ = 2526f16306c819d651af075934170d2430d246d9ab98d975d28a83baded47ca7

Hashing and chaining transactions



$H(T_1)$ = 2526f16306c819d651af075934170d2430d246d9ab98d975d28a83baded47ca7

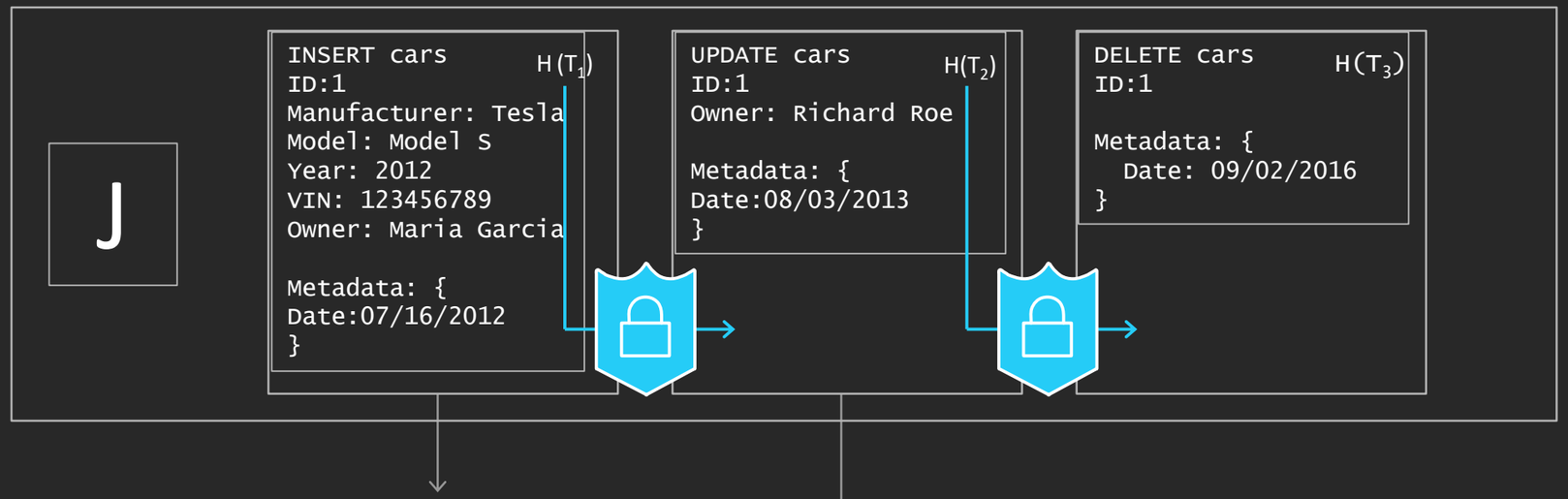
$H(T_1)$ +

```
UPDATE cars
ID:1
Owner: Richard Roe

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

SHA-256

$H(T_2)$ = 86a90e4166453d9423b84d47dcbd97c0e3099b1a1f0d7cfca6c191d8fd8994ff

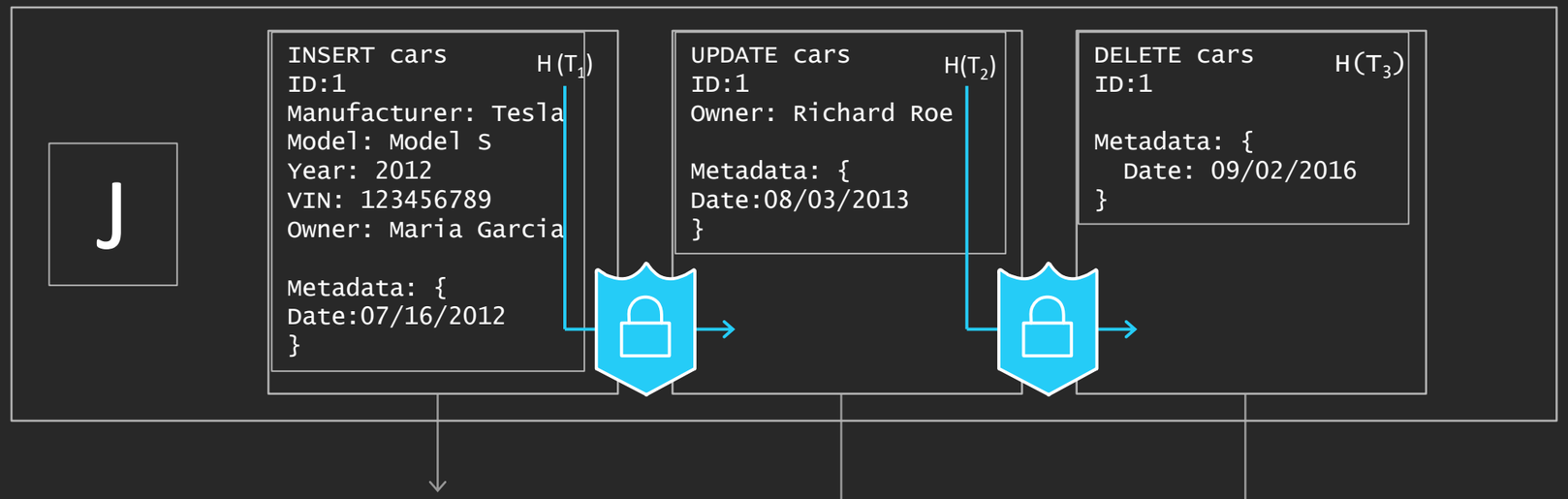
Hashing and chaining transactions



$H(T_1)$ = 2526f16306c819d651af075934170d2430d246d9ab98d975d28a83baded47ca7

$H(T_2)$ = 86a90e4166453d9423b84d47dcbd97c0e3099b1a1f0d7cfca6c191d8fd8994ff

Hashing and chaining transactions

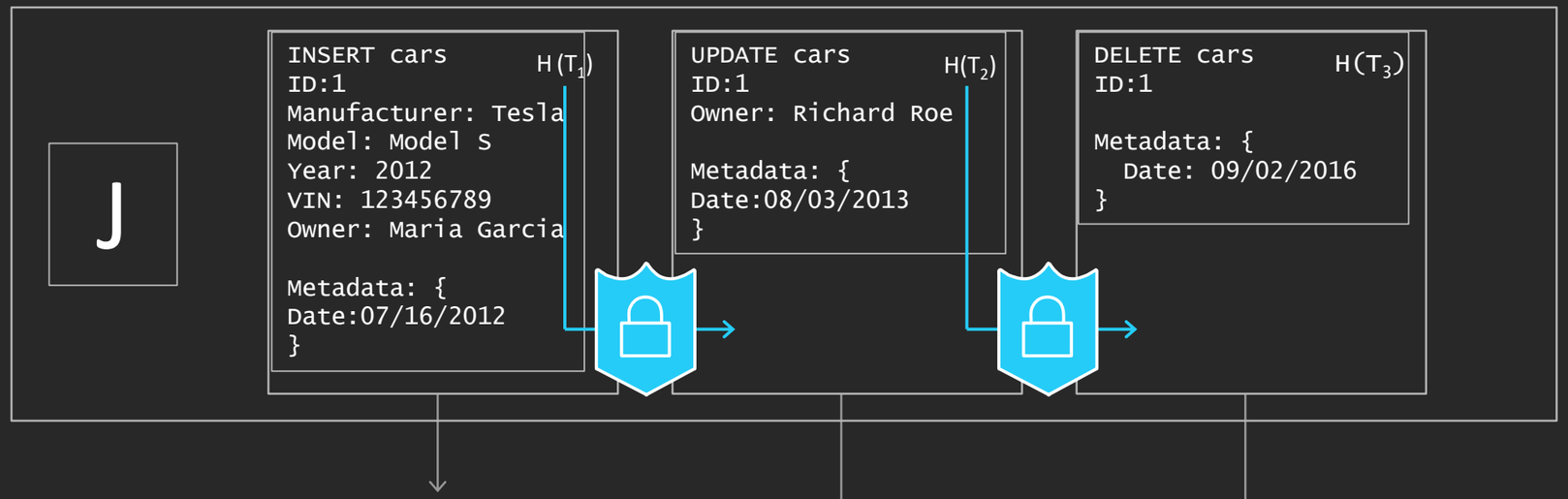


$H(T_1)$ = 2526f16306c819d651af075934170d2430d246d9ab98d975d28a83baded47ca7

$H(T_2)$ = 86a90e4166453d9423b84d47dcbd97c0e3099b1a1f0d7cfca6c191d8fd8994ff

$H(T_3)$ = ae2d64e562ec754ec3194c744eec72c9fdafffc6b559e0414d0e75bf96ca92ad

A digest is a hash value at a point in time

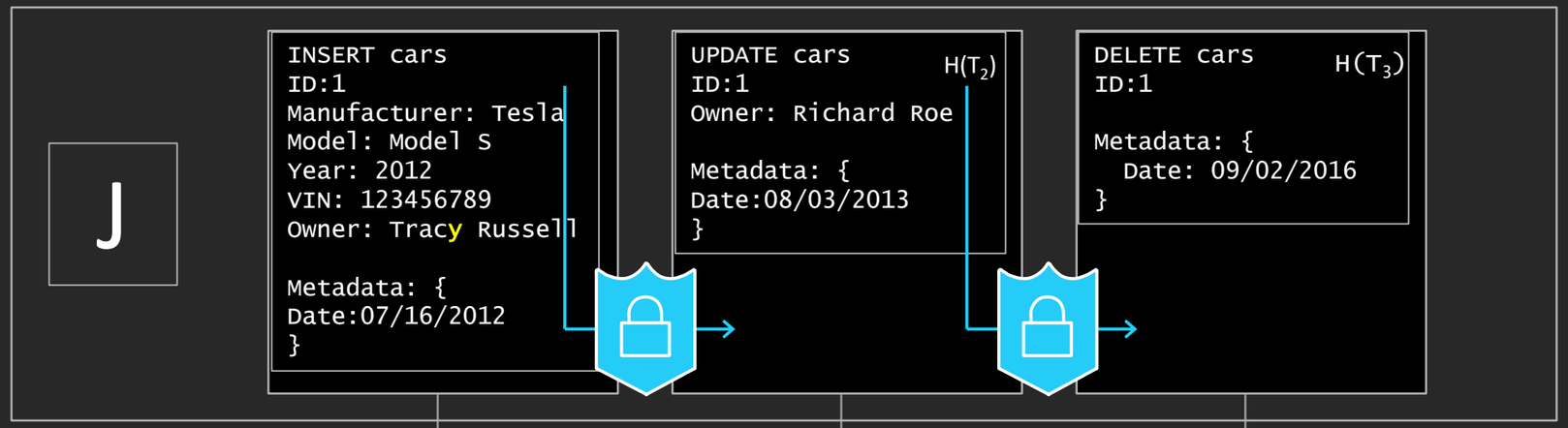


$H(T_1)$ = 2526f16306c819d651af075934170d2430d246d9ab98d975d28a83baded47ca7

$H(T_2)$ = 86a90e4166453d9423b84d47dcbd97c0e3099b1a1f0d7cfca6c191d8fd8994ff

$H(T_3)$ = ae2d64e562ec754ec3194c744eec72c9fdafffc6b559e0414d0e75bf96ca92ad

Changing committed data breaks the chain



$H(T_1)$ = ~~2526f16306c819d651af075934170d2430d246d9ab98d975d28a83baded47ca7~~

$H(T_1)$ = 25d0b44e6e8878151646ffc1fea4eb85c3e4bf4baec212a9fcf67b6d5a81e01a

$H(T_2)$ = ~~86a90c4166453d9423b84d47dcbd97c0e3099b1a1f0d7cfca6c191d8fd8994ff~~

$H(T_2)$ = a90a9898c7e4b1aab19c705b554afd9e0bf6539bb0346df19be362ff63001098

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

$H(T_3)$ = c6268578a24dbe0c7cfba07bd967411a35462b8c875d42f1991faad02c0ac93c

Challenges customers face

Building ledgers with traditional databases

- Resource intensive
- Difficult to manage and scale
- Error prone and incomplete
- Impossible to verify

Blockchain approaches

- Designed for a different purpose
- Adds unnecessary complexity

Leveraging Amazon QLDB as a trusted verifiable ledger for automotive data

Emil Djerekarov

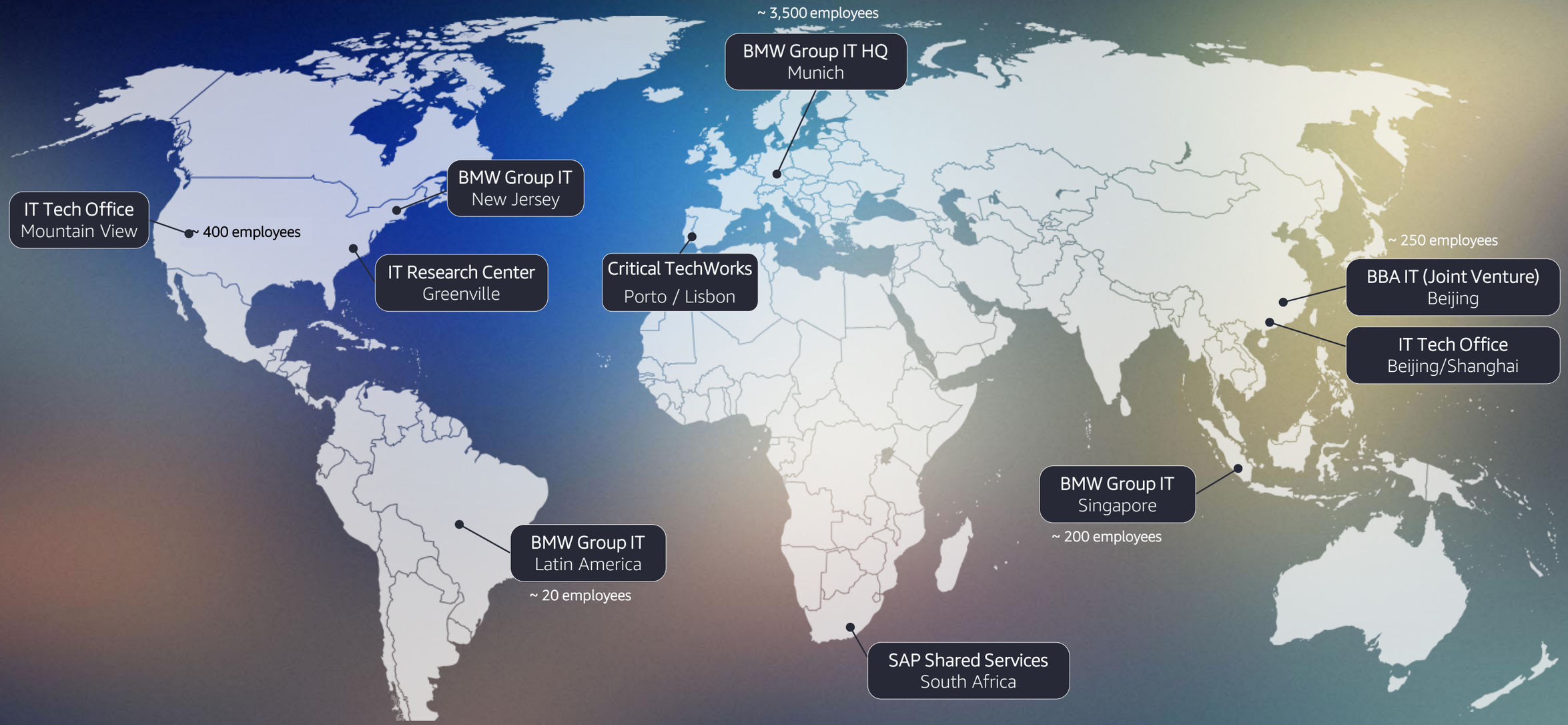
Technical Architect

BMW Group

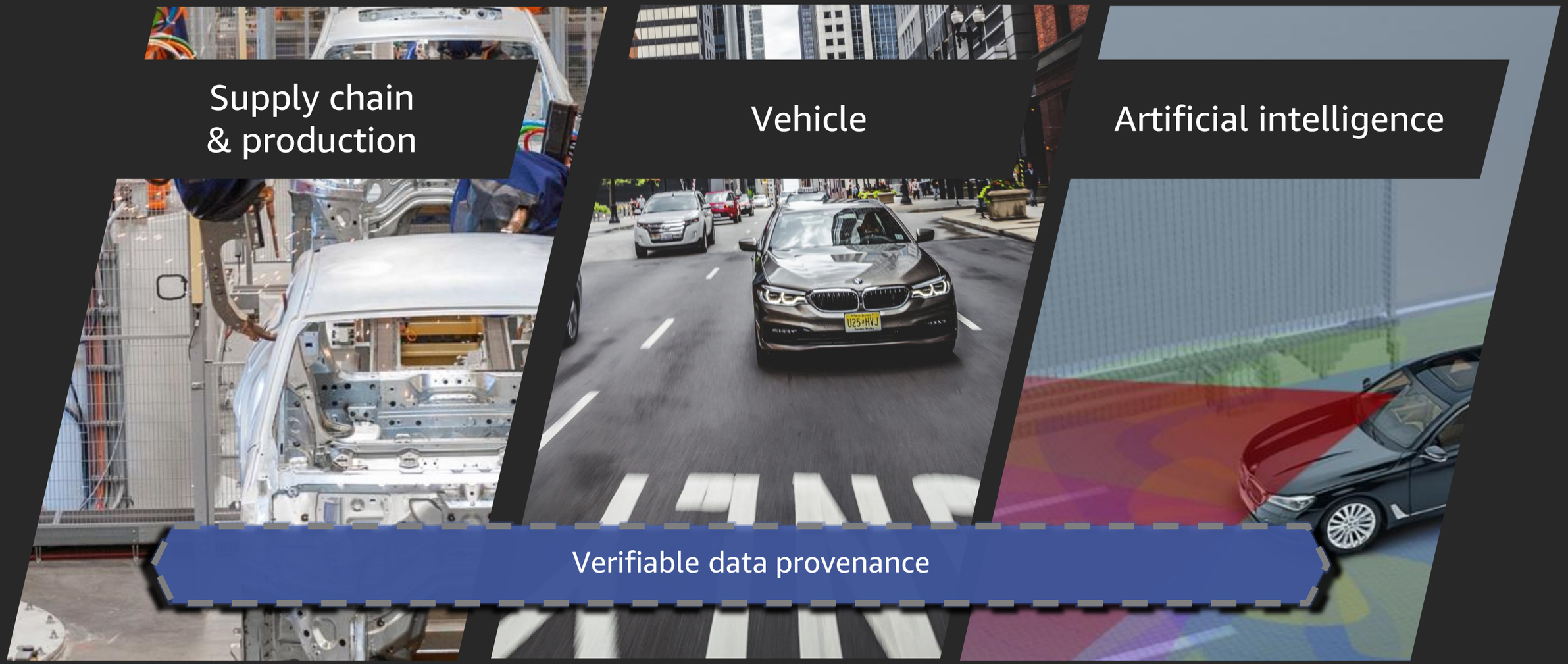
AWS re:Invent

aws

BMW Group IT


~ 3,500 employees
BMW Group IT HQ
Munich
BMW Group IT
New Jersey
IT Tech Office
Mountain View
~ 400 employees
IT Research Center
Greenville
Critical TechWorks
Porto / Lisbon
~ 250 employees
BBA IT (Joint Venture)
Beijing
IT Tech Office
Beijing/Shanghai
BMW Group IT
Singapore
~ 200 employees
BMW Group IT
Latin America
~ 20 employees
SAP Shared Services
South Africa

Ledger technologies within automotive industry



Ledger technologies enable new business models

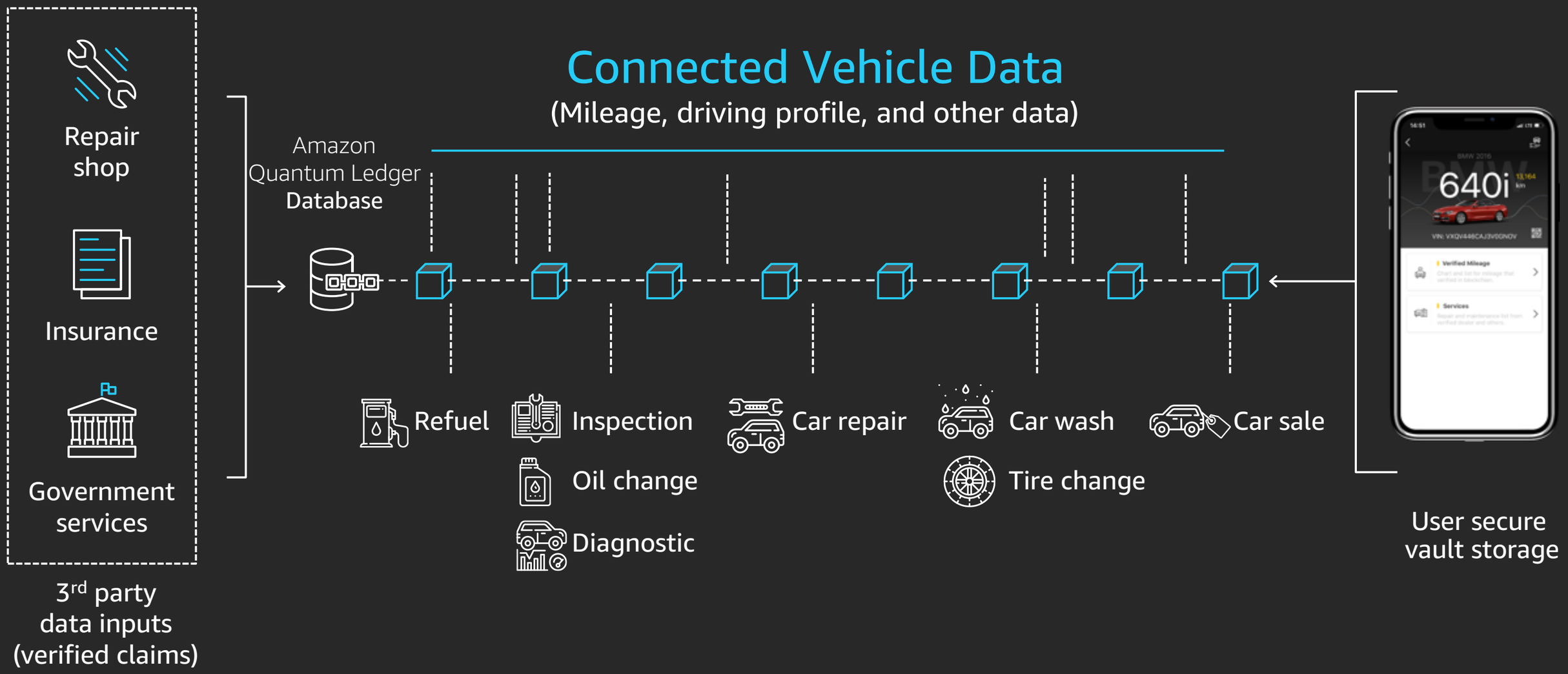
Data must be verified and auditable

Multiple organizations and partners involved

Ensure customer privacy

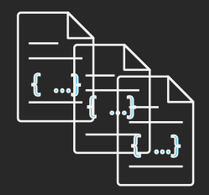
Cryptographic Ledger Technology enables end-to-end data pipelines with verification built in at every stage

Trusted vehicle data: The foundation of ecosystems
Digital Vehicle Passport Prototype



QLDB as enabler for automotive data ecosystems

Historical data



Journaled data revisions

Trusted, verifiable data

Cryptographic hashing / verification

Amazon Quantum Ledger Database

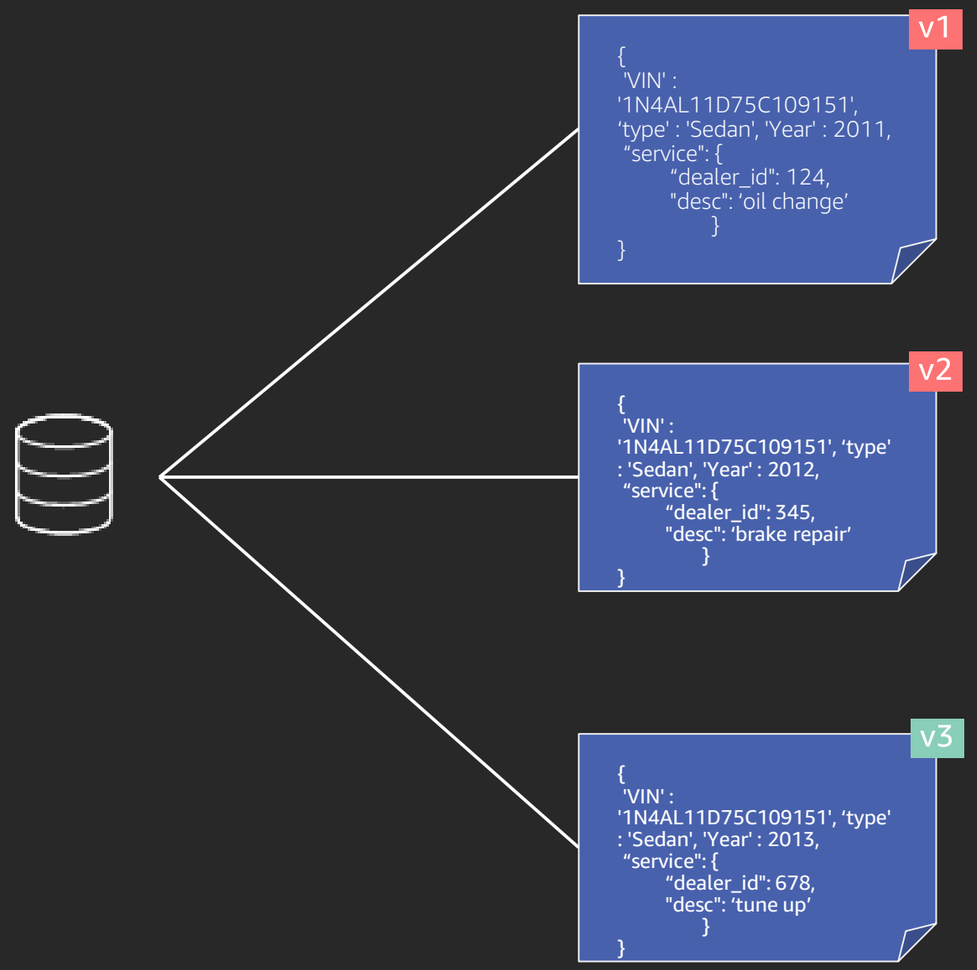
On-demand access

Higher throughput than decentralized ledgers

Ease of use

Schema-less design: SQL-like, ACID

The history / versioning problem



Enables:

- Data integrity protection
- Lock access to old document versions
- Disable deletion capability even for admin
- Manually keep indexes / tracking of doc versioning
- Allow external data verification

QLDB provides elegant solution to this problem out-of-the-box

Data model

```
{
  "document_id": "3Lb4dnu5c2Y3Uc04OM",
  "vid": "WBA123456789012345",
  "ownerID": "bmw.driver@bmw.com",

  "vehicleSpecification": {
    "brand": "BMW",
    "model": "I8",
    "productionDate": 2016-08-23,
    "color": "FLUID BLACK MIT AKZENT",
    "driveTrain": "BEV",
    …
  },

  "lastState": {
    "lastStateCallTime": 1570782292689,
    "geoPosition": "Fi6bnRHN10mCwcR",
    "mileage": "AzLG66DISvu2fGhssN"
  },

  "services": [
    "L2889EwdMA4jtnmR",
    "Tses284nfnwi246k",
    …
  ],
  …
}
```

Mileage doc

```
Doc_id: AzLG66DISvu2f5L3lGhssN
Rev_id: 1
-----------------------
{
 "mileage": {
  "value":2392,
  "time": "2019-04-19T
  }
}
```

```
Doc_id: AzLG66DISvu2f5L3lGhssN
Rev_id: 2
-----------------------
{
 "mileage": {
  "value":3598,
  "time": "2019-04-30
  }
}
```

```
Doc_id: AzLG66DISvu2f5L3lGhssN
Rev_id: 3
-----------------------
{
 "mileage": {
  "value":4844,
  "time": "2019-05-12T09:00:55.223Z"
  }
}
```

Services docs

```
Doc_id: L2889EwdMA4jtnmR
Rev_id: 1
-------
{
 "serv
  "de
  "ds
   …
 } …
```

```
Doc_id: L2889EwdMA4jtnmR
Rev_id: 2
-------
{
 "serv
  "de
  "ds
   …
 } …
```

```
Doc_id: L2889EwdMA4jtnmR
Rev_id: 3
-----------------------
{
 "service": {
   "dealer_id": 124,
   "dsc": "oil change"
   …
 } …
```

```
Doc_id: Tses284nfnwi246k
Rev_id: 1
-----------------------
{
 "service": {
   "date" : "2019-05-12T09:00:55.223Z"
   "dealer_id": 5790,
   "service_dsc": "tyre rotaion"
   …
 } …
```

Any doc (i.e., any new modules added in future)

```
Doc_id: ------------------
Doc_id: ------------------
Doc_id: ------------------
Rev_id: …
-----------------------
{…}
```

Data model with customer encryption

```
{
  "document_id": "3Lb4dnu5c2Y3Uc04OM",
  "vid": "WBA123456789012345",
  "ownerID": "bmw.driver@bmw.com",

  "vehicleSpecification": {
    "brand": "BMW",
    "model": "I8",
    "productionDate": 2016-08-23,
    "color": "FLUID BLACK MIT AKZENT",
    "driveTrain": "BEV",
    …
  },

  "lastState": {
    "lastStateCallTime": 1570782292689,
    "geoPosition": "Fi6bnRHN10mCwcR",
    "mileage": "AzLG66DISvu2fGhssN"
  },

  "services": [
    "L2889EwdMA4jtnmR",
    "Tses284nfnwi246k",
    …
  ],
  …
}
```

Mileage doc

```
Doc_id: AzLG66DISvu2f5L3lGhssN
Rev_id: 1
-----------------------
aWx0b24xETAPBgNVBAgMCFBlbWJ
QYDVQQGEwJCTTEPMA0GCSqGSIb3
GfMA0GCSqGSIb3DQEBAQUAA4GNA
```

```
Doc_id: AzLG66DISvu2f5L3lGhssN
Rev_id: 2
-----------------------
QEJARYAMIGfMA0GCSqGSIb3DQE
CBiQKBgQCJ9WRanG/fUvcQEJAR
SqGSIb3DQEBAQUAA4GNADCBiQK
```

```
Doc_id: AzLG66DISvu2f5L3lGhssN
Rev_id: 3
-----------------------
QEJARYAMIGfMA0GCSqGSIb3DQEBAQUAA4GNAD
CBiQKBgG/fUvcQEJARYAMIGfMA0GCSqGSIb3D
QEBAQUAA4GN
```

Services docs

```
Doc_id: L2889EwdMA4jtnmR
Rev_id: 1
-----------------
PMA0GC
fMA0GC
NADCBi
```

```
Doc_id: L2889EwdMA4jtnmR
Rev_id: 2
-----------------
PMA0GC
fMA0GC
NADCBi
```

```
Doc_id: L2889EwdMA4jtnmR
Rev_id: 3
-----------------
TEPMA0GCSqGSIb3DQEJARYAM
IGfMA0GCSqGSIb3DQEBAQUAA
4GNADCBiQKBgQ/CJ9WRa
```

```
Doc_id: Tses284nfnwi246k
Rev_id: 1
-----------------
0GCSq0GCSqGSIb3DQEBAQUAA4
GNADCBiQK3DQEJARYAMIGfMA0
GCSqGSIb3DQBgQCJ9WRanGGSI
b3DQEBAQUAA4GNADCBiQKBgQC
J9WRanG
```

Any doc (i.e., any new modules added in future)

```
Doc_id: ----------
R
-
{
```

```
Doc_id: ----------
R
-
{
```

```
Doc_id: ----------
Rev_id: …
-----------
{…}
```

Verifying a record

1.) Capture Docid+Blockaddress of the document

```
Doc_id: L2889EwdMA4jtnmR
Rev_id: 3
----------------------
TEPMA0GCSqGSIb3DQEJARYAM
IGfMA0GCSqGSIb3DQEBAQUAA
4GNADCBiQKBgQ/CJ9WRa
```

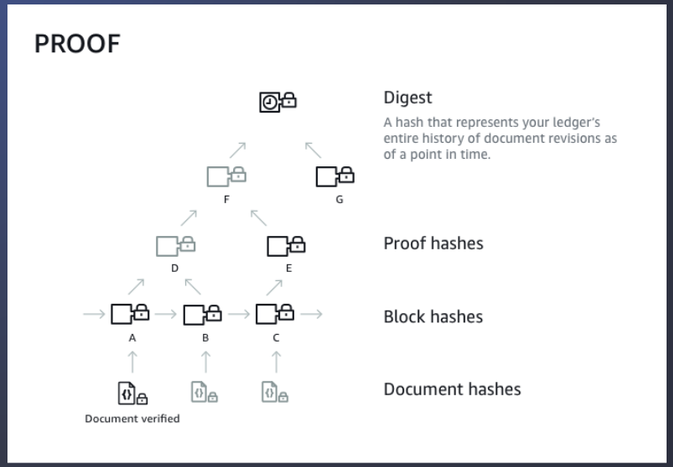
```
SELECT r.metadata.id, r.blockAddress FROM
_ql_committed_VehicleServices AS r
WHERE …
```

2.) Retrieve the QLDB Digest

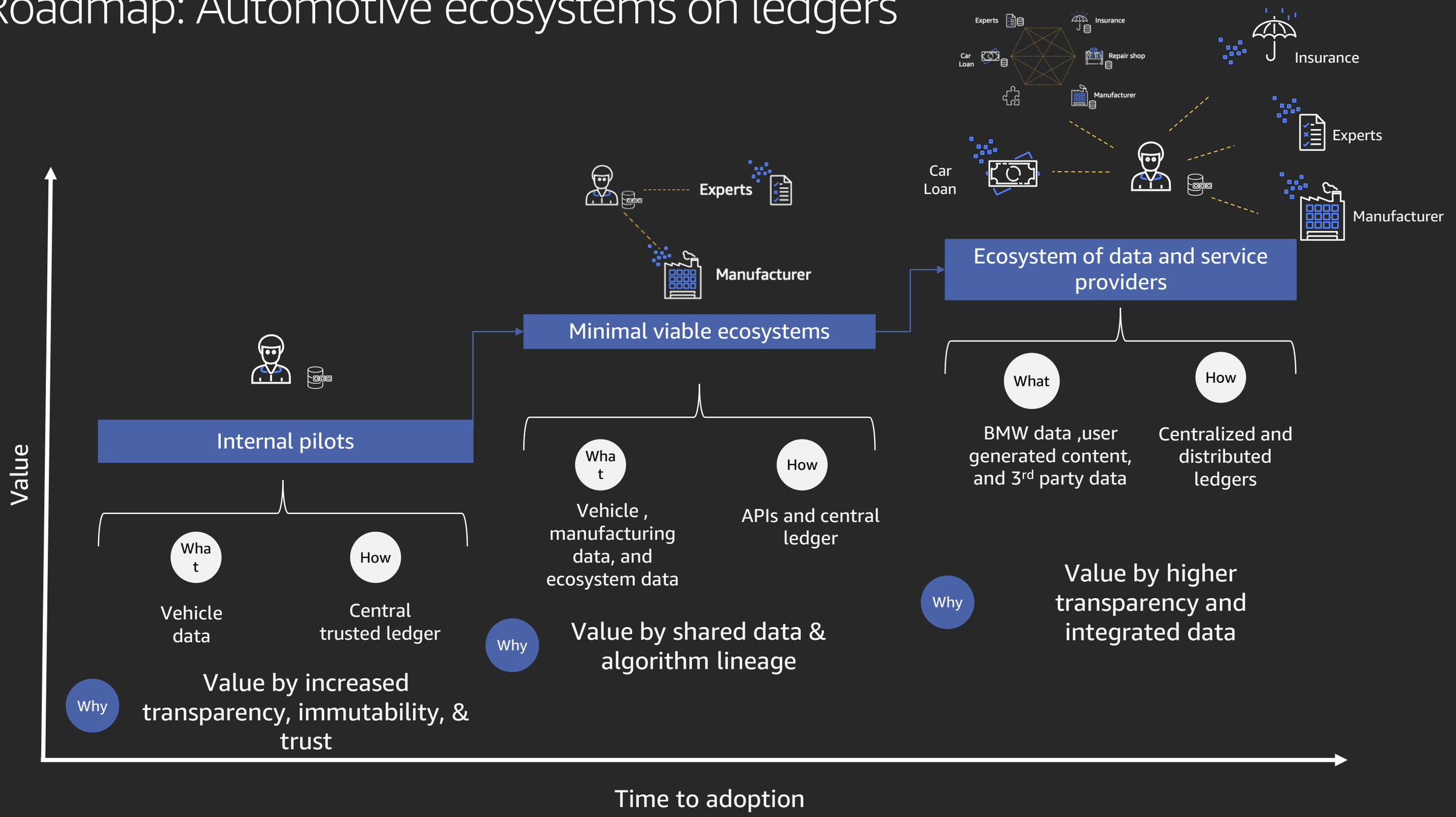

```
{ "digest": "42zaJOfV8iGuthD5Xb/5B9lScHnvxPXm9E=",
"digestTipAddress": "{strandId:"BlFTjlSXzOszcE3",sequenceNo:73}",
 "ledger": "my-ledger",
"date": "2019-04-17T16:57:26.749Z" }
```


3.) Execute the verification process (proof)

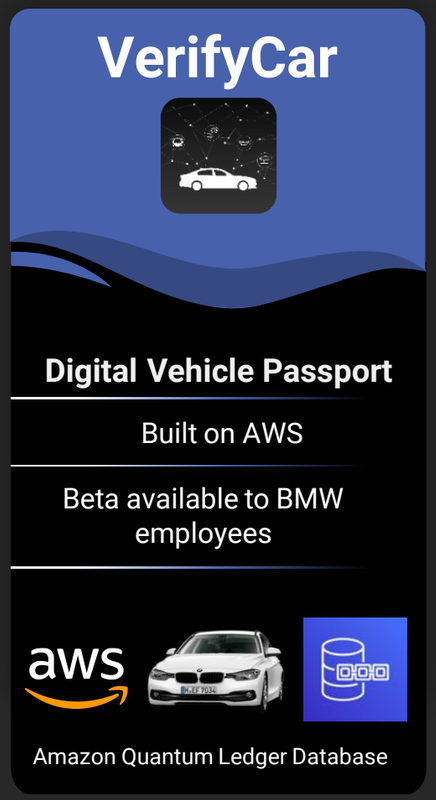
a4e31e36910d99bd19b7f875f0a04597dc0ff52c2f164a16a9288aed9e710fdd → d07fc3d67314905dd065d55988790070410e87072f27ce2d1cb56457c0879bc8 → 2fc7e994c884bd13d5fd22b7425328d0e5d5b0cdcba4d285b198be612f229ccb

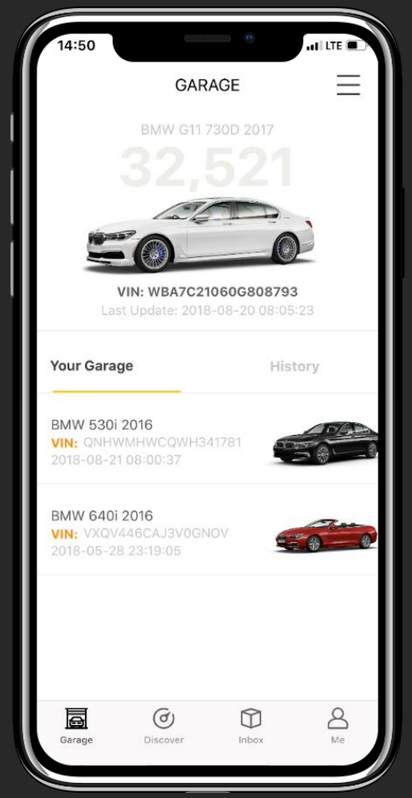


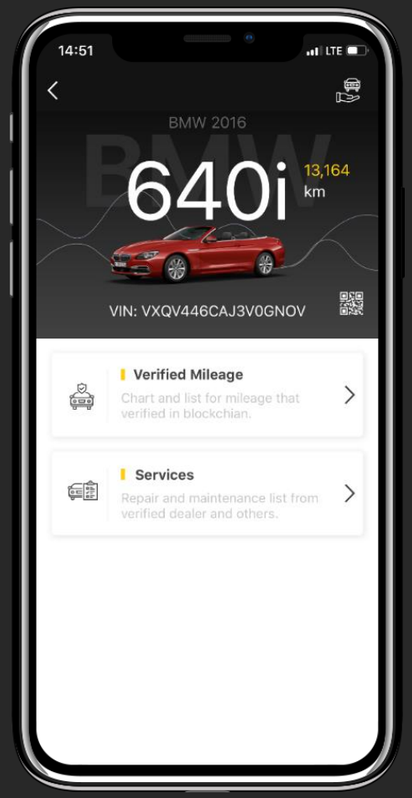
Roadmap: Automotive ecosystems on ledgers

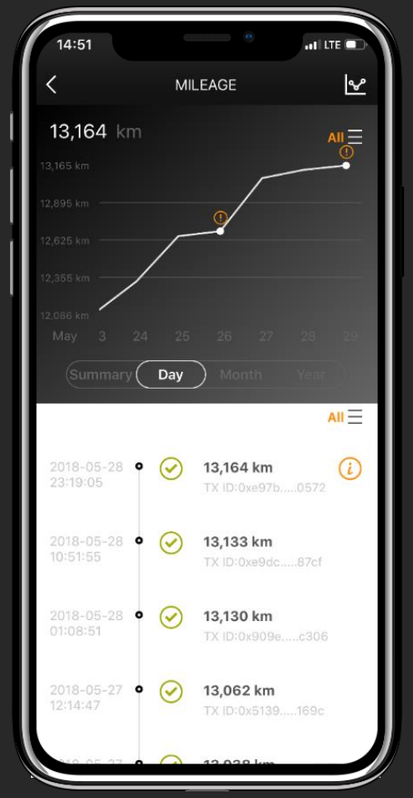


Digital vehicle passport application









Thank you!

AWS re:Invent

aws

Please complete the session survey in the mobile app.