CONFLUENT

ksqlDB: A Stream-Relational Database System

Matthias J. Sax | Software Engineer

 **@MatthiasJSax**

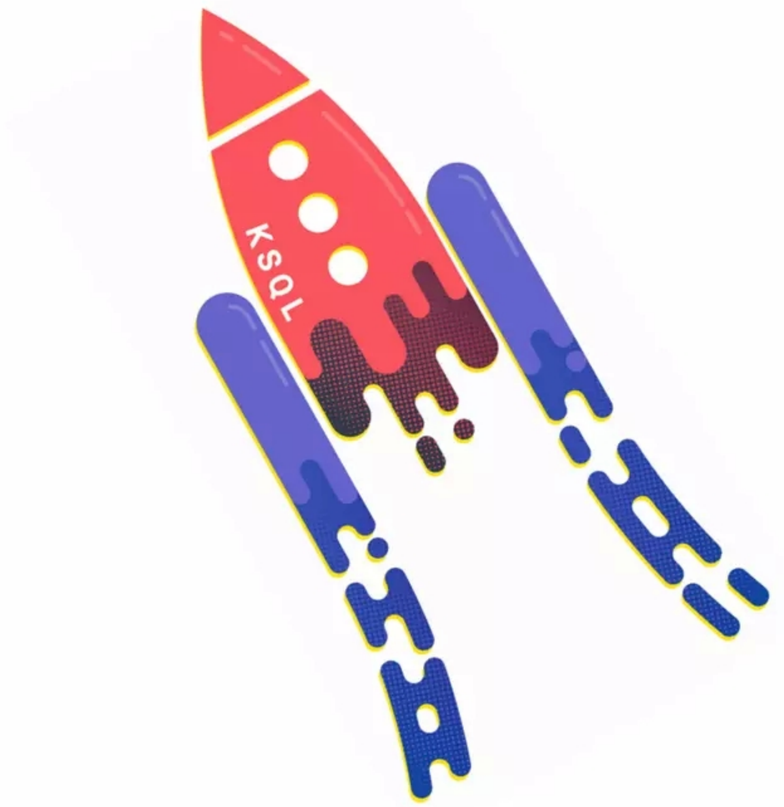
The Vision of a Streaming Database

SQL abstraction over mutable TABLEs and immutable append-only STREAMs

- OLTP?
  - transaction, concurrent updates
- OLAP?
  - complex ad-hoc queries

- Online Stream Processing:
  - Continuous queries over STREAMs and TABLEs
  - Simple state lookups into TABLEs
  - Subscribe to data streams

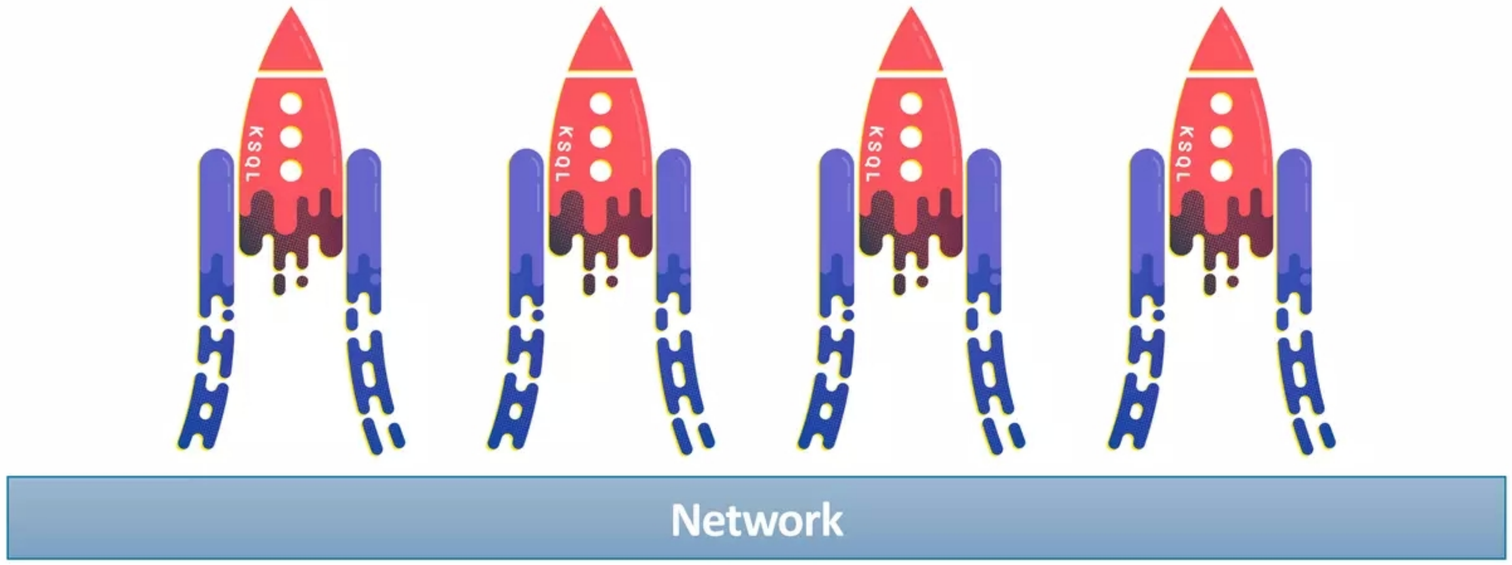
  - *Streaming ETL*
  - *Materialized View maintenance*



Apache Kafka and ksqlDB

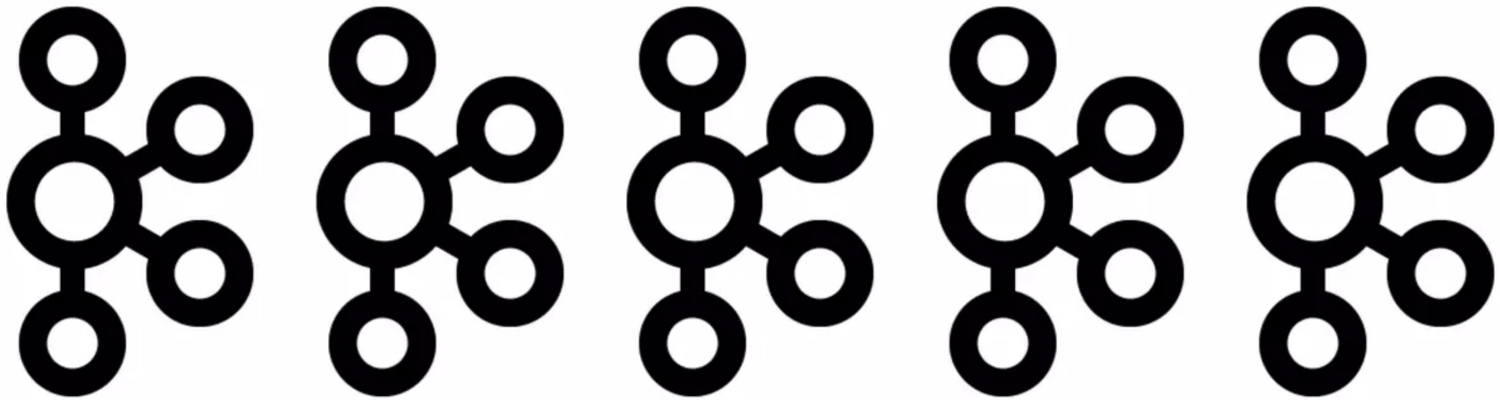
ksqlDB Cluster

"Compute Layer"



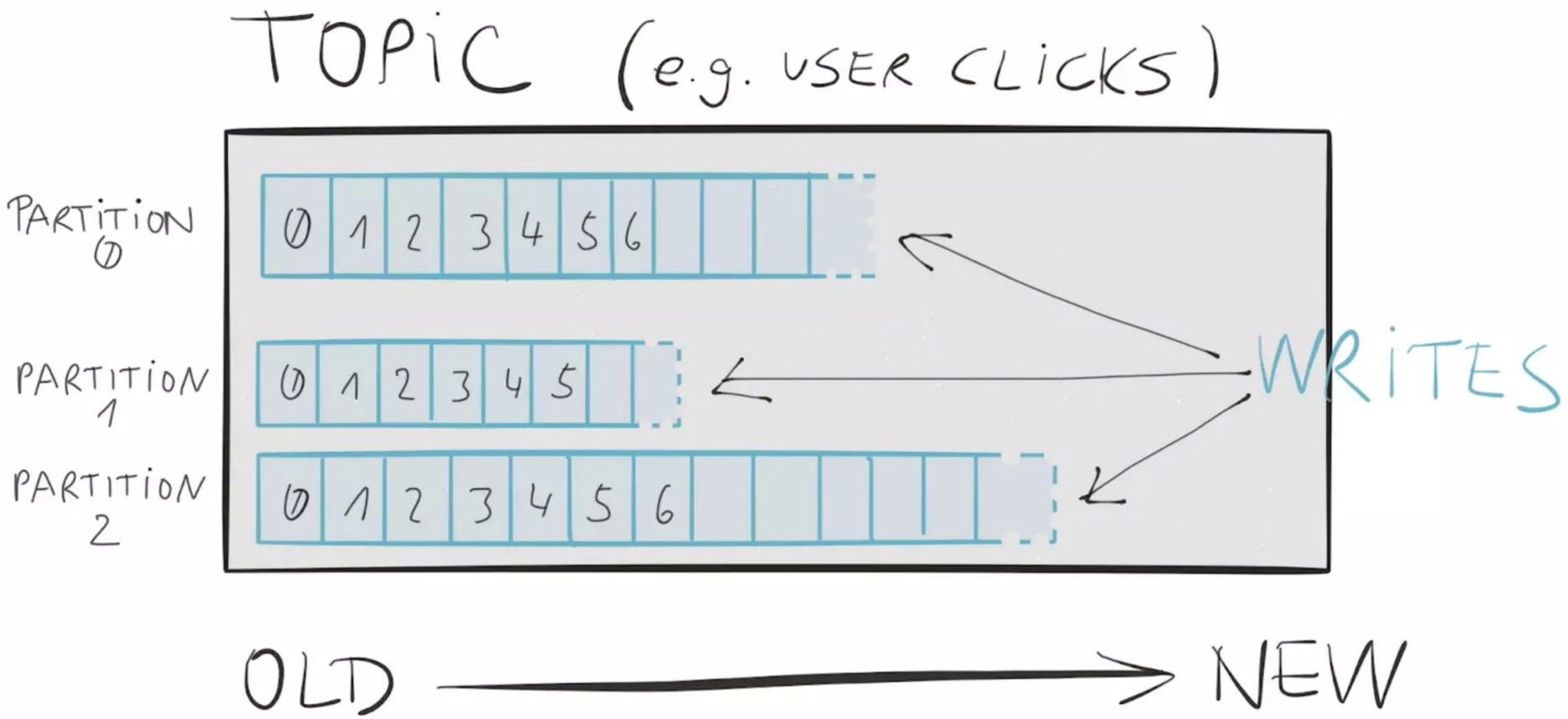
Broker Cluster

"Storage Layer"



Apache Kafka: Data Model

Replicated & partitioned, immutable append-only log of messages



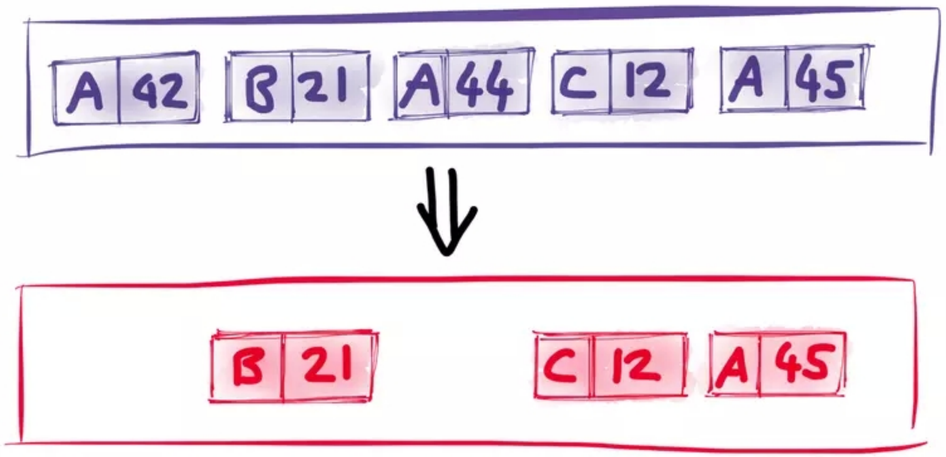
Apache Kafka: Data Model (cont.)

Messages:

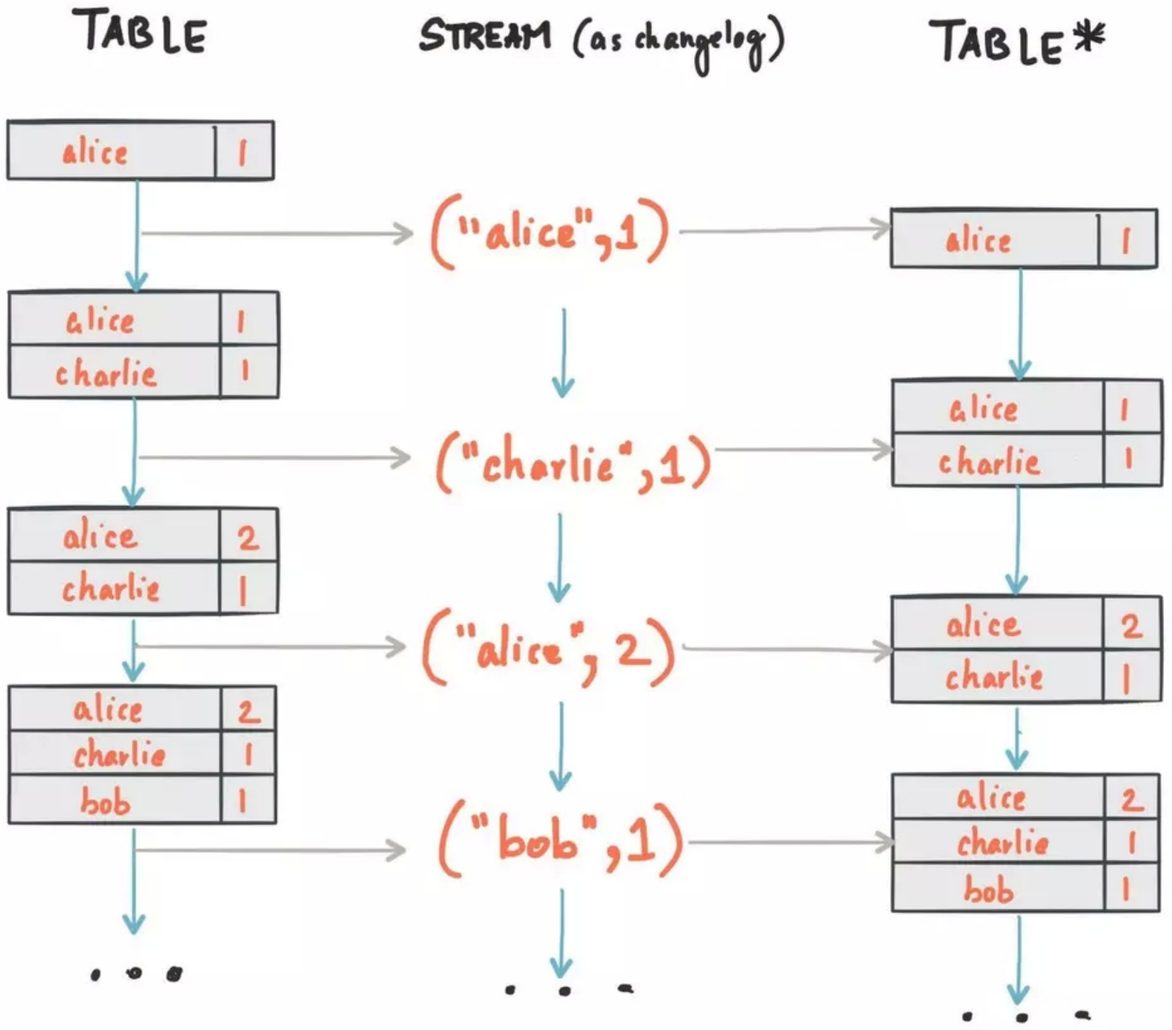
- plain bytes: ***key*** (for partitioning), ***value***
- long ***timestamp***

Topic cleanup policies:

- ***retention:*** purge old messages (time or size based)
- ***compaction:*** keep the latest message per key



The Stream-Table Duality



ksqlDB: Data Model

Streams and tables of structured records

```
CREATE STREAM clickstream (
  time    BIGINT,
  url     VARCHAR,
  status  INTEGER,
  bytes   INTEGER,
  user_id VARCHAR,
  agent   VARCHAR)
WITH (
  kafka_topic = 'cs_topic',
  value_format = 'JSON'
);
```

```
CREATE TABLE users (
  user_id       INTEGER PRIMARY KEY,
  registered_at LONG,
  username      VARCHAR,
  name          VARCHAR,
  city          VARCHAR,
  level         VARCHAR)
WITH (
  kafka_topic = 'users_topic',
  value_format = 'AVRO'
);
```

ksqlDB: Queries

Persistent continuous queries (CQ)

- Take one or more input STREAMs/TABLEs and compute a result STREAM or TABLE
- Deployed in the ksqlDB servers
- Executed in a data-parallel manner
- Scalable
- Fault-tolerant

```
CREATE STREAM resultStream
  AS SELECT...
```

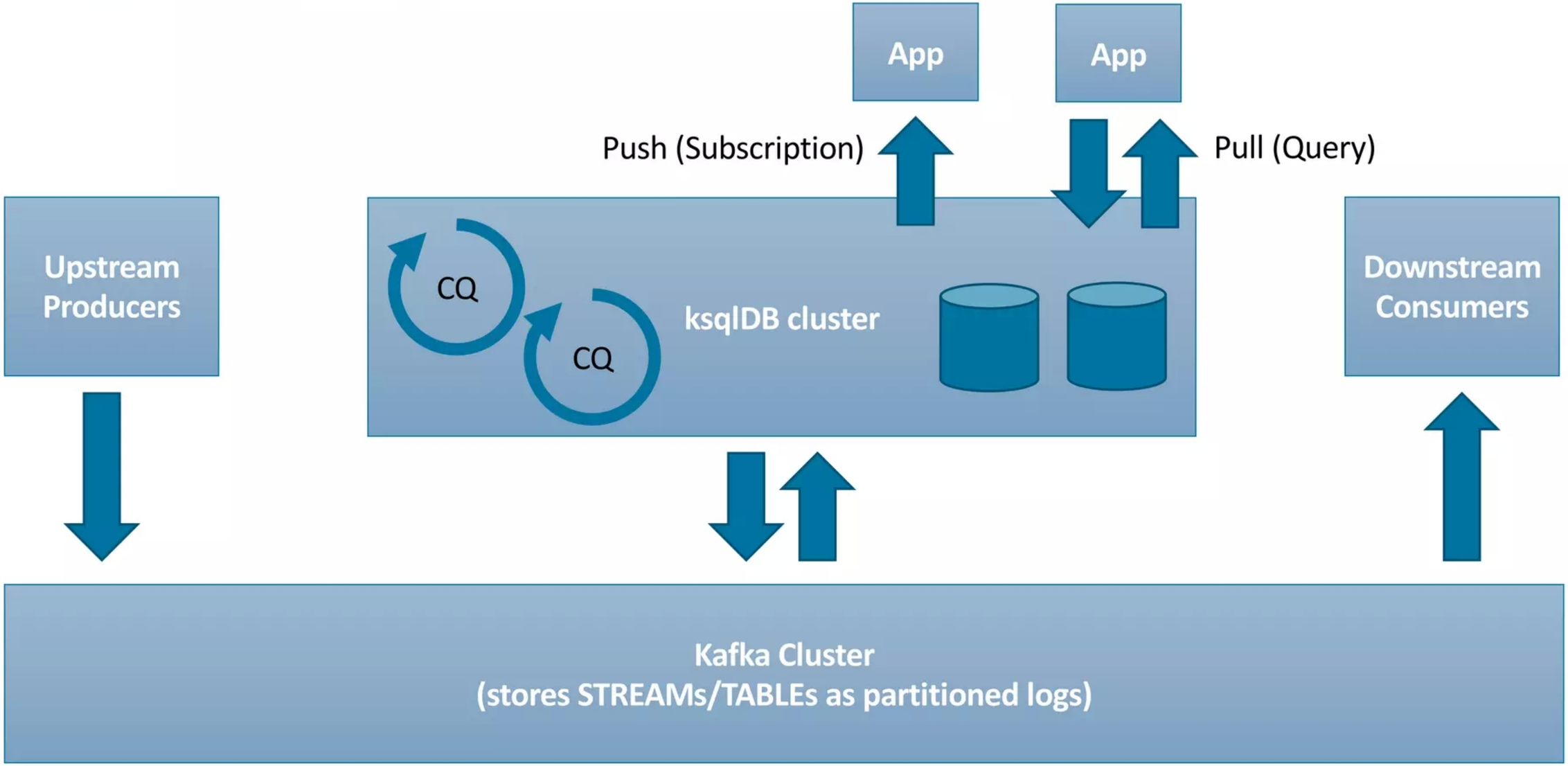
```
CREATE TABLE resultTable
 AS SELECT...
```

ksqlDB: Queries (cont.)

Transient client queries

- CLI / Java client / etc.
- Pull queries: simple state lookups against TABLEs
  - "classic" queries
  - Limited in ksqlDB (no aggregations or joins)
- Push queries: ***subscription*** to a result STREAM

ksqlDB: Overview



Querying

Queries Semantics

Streaming queries have *temporal semantics* based on *event-time*

TABLE queries:

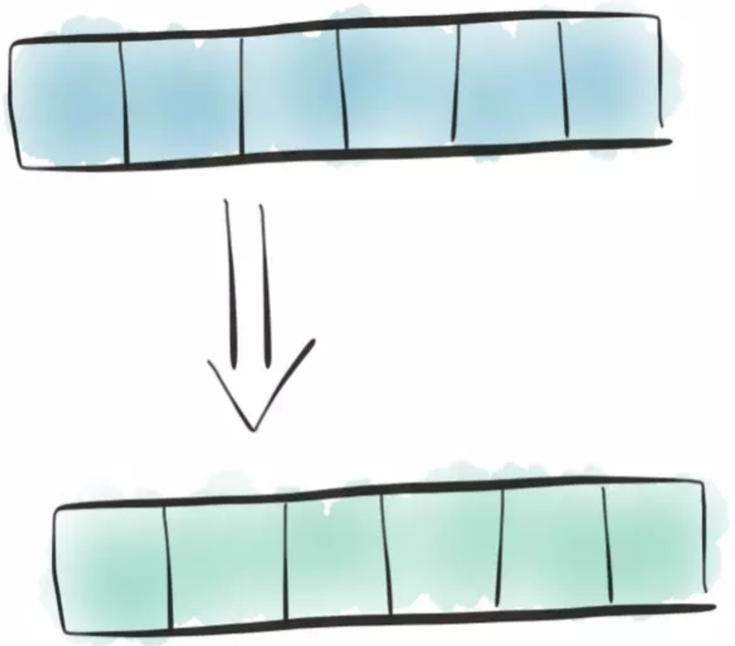
- Like regular SQL
- However: TABLEs are treated as “versioned” tables that evolve over time

How to query a STREAM?

Simple Stream Queries

Filter, projection etc (stateless)

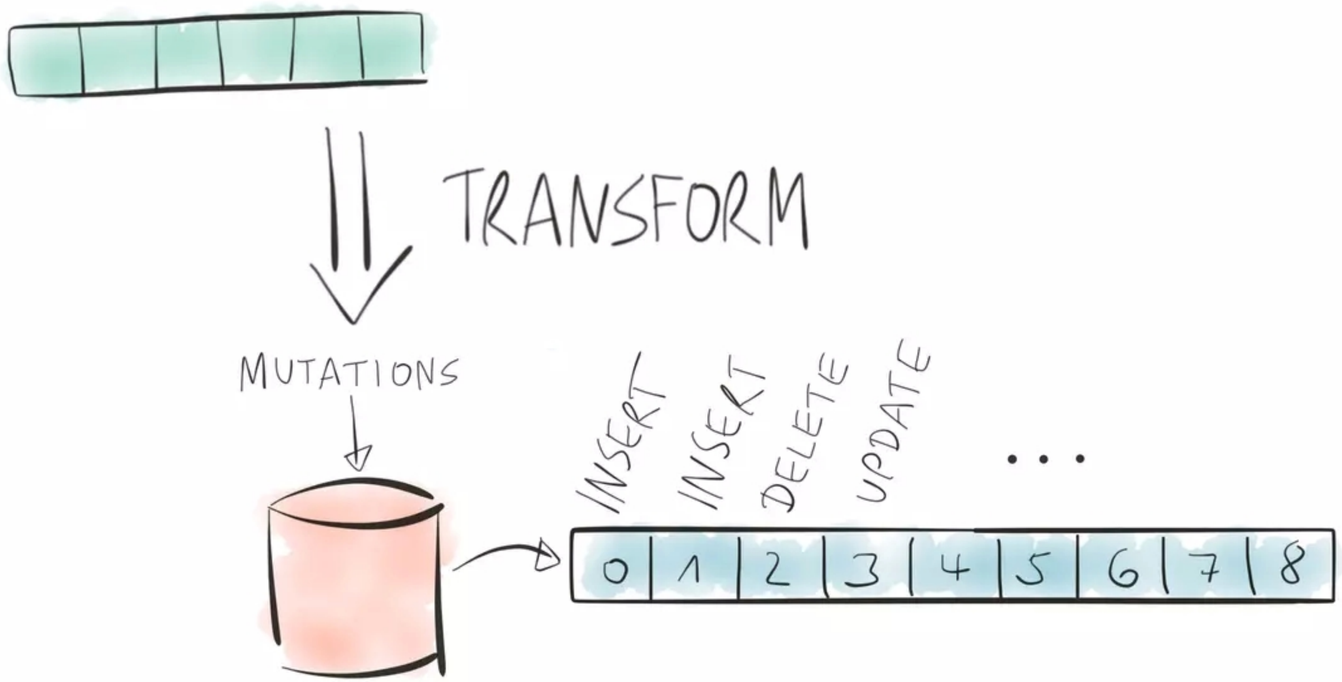
```
CREATE STREAM user_clicks AS
  SELECT user_id, status, ucase(agent)
  FROM clickstream
  WHERE user_id = 'mjsax';
```



Stream Aggregation

Windowed / non-windowed: returns a *continuously updating* TABLE

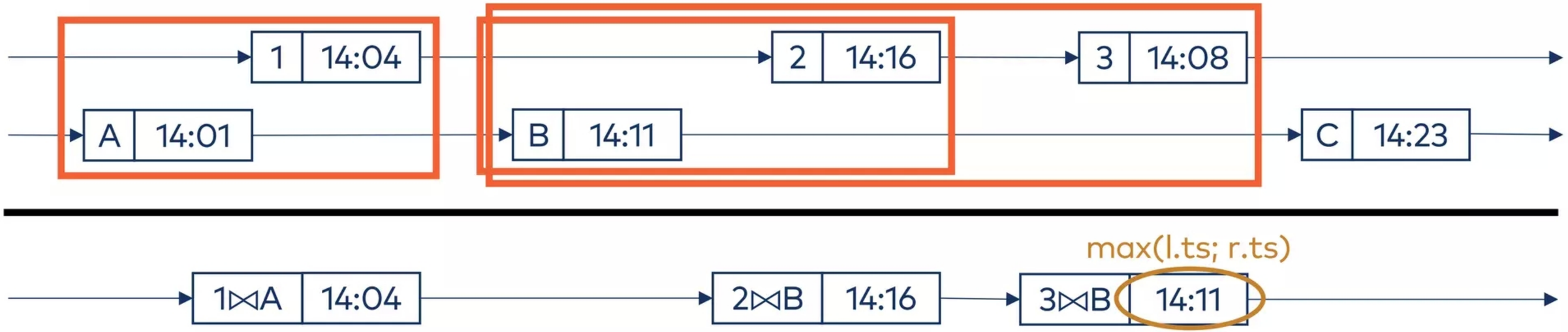
```
CREATE TABLE clicks AS
  SELECT user_id, COUNT(url)
  FROM clickstream
    WHERE bytes > 1024
    WINDOW TUMBLIND
      (size 30 seconds)
    GROUP BY user_id
    HAVING COUNT(url) > 20;
```



Stream-Stream Join

Sliding-window join:

```
CREATE STREAM joinedStream AS
  SELECT *
  FROM leftStream AS l JOIN rightStream AS r
    WITHIN 5 minutes ON l.id = r.id;
```

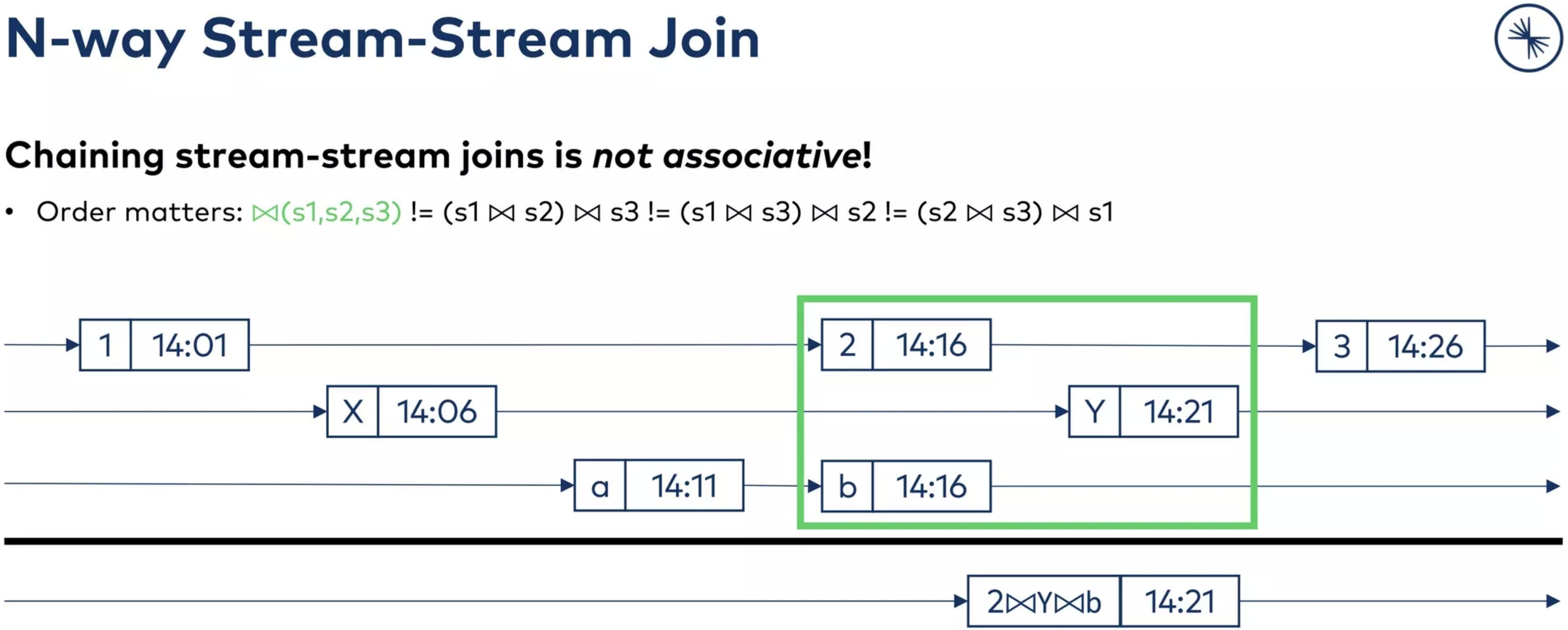


N-way Stream-Stream Join

Chaining stream-stream joins is *not associative*!

- Order matters: ⋈(s1,s2,s3) != (s1 ⋈ s2) ⋈ s3 != (s1 ⋈ s3) ⋈ s2 != (s2 ⋈ s3) ⋈ s1

N-way Stream-Stream Join



Chaining stream-stream joins is *not associative*!

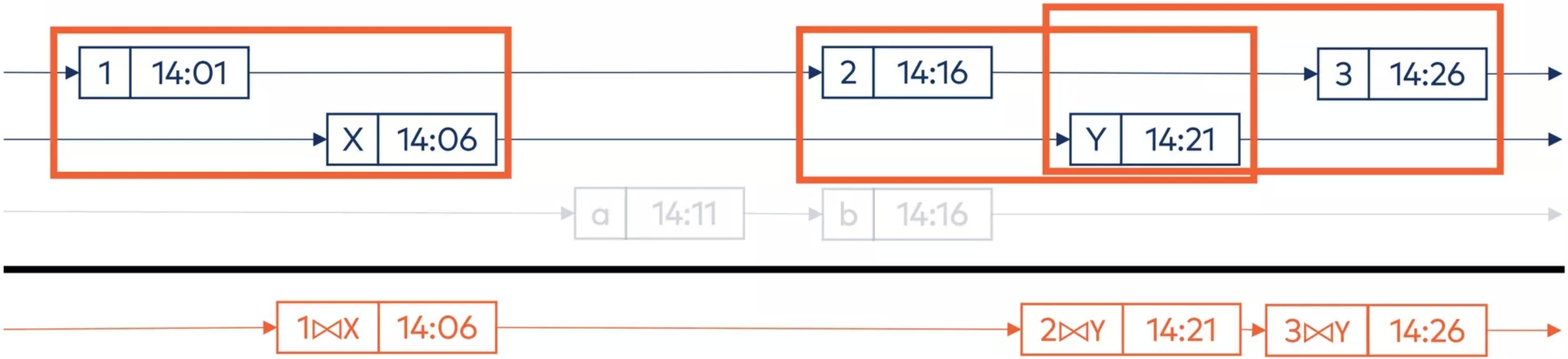
- Order matters: ⋈(s1,s2,s3) != (s1 ⋈ s2) ⋈ s3 != (s1 ⋈ s3) ⋈ s2 != (s2 ⋈ s3) ⋈ s1

* window size=5min

N-way Stream-Stream Join

Chaining stream-stream joins is *not associative*!

- Order matters: ⋈(s1,s2,s3) != (s1 ⋈ s2) ⋈ s3 != (s1 ⋈ s3) ⋈ s2 != (s2 ⋈ s3) ⋈ s1

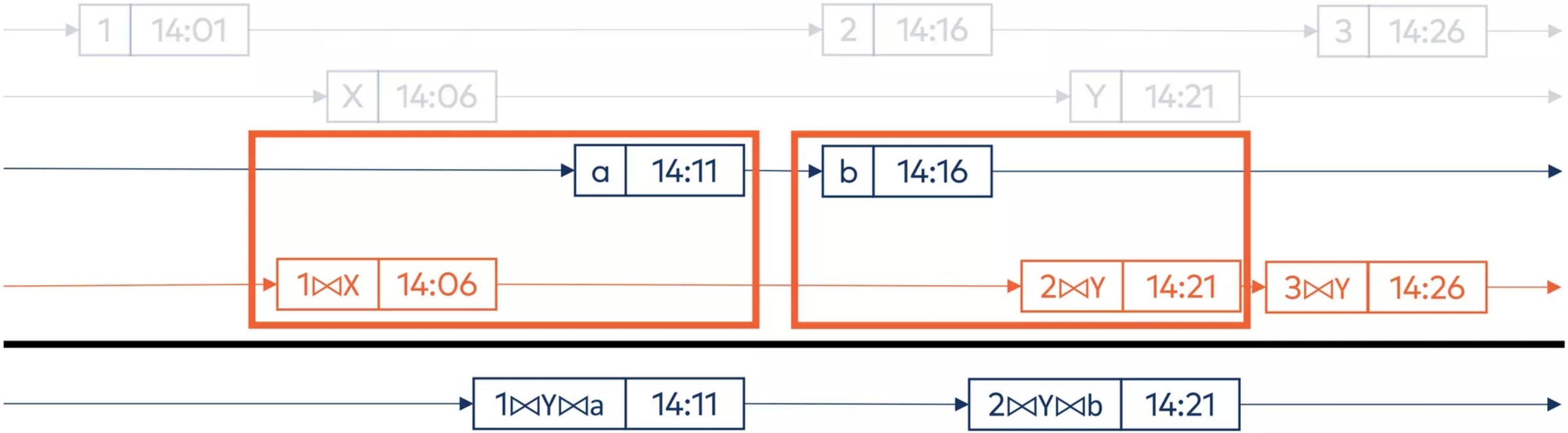


* window size=5min

N-way Stream-Stream Join

Chaining stream-stream joins is *not associative*!

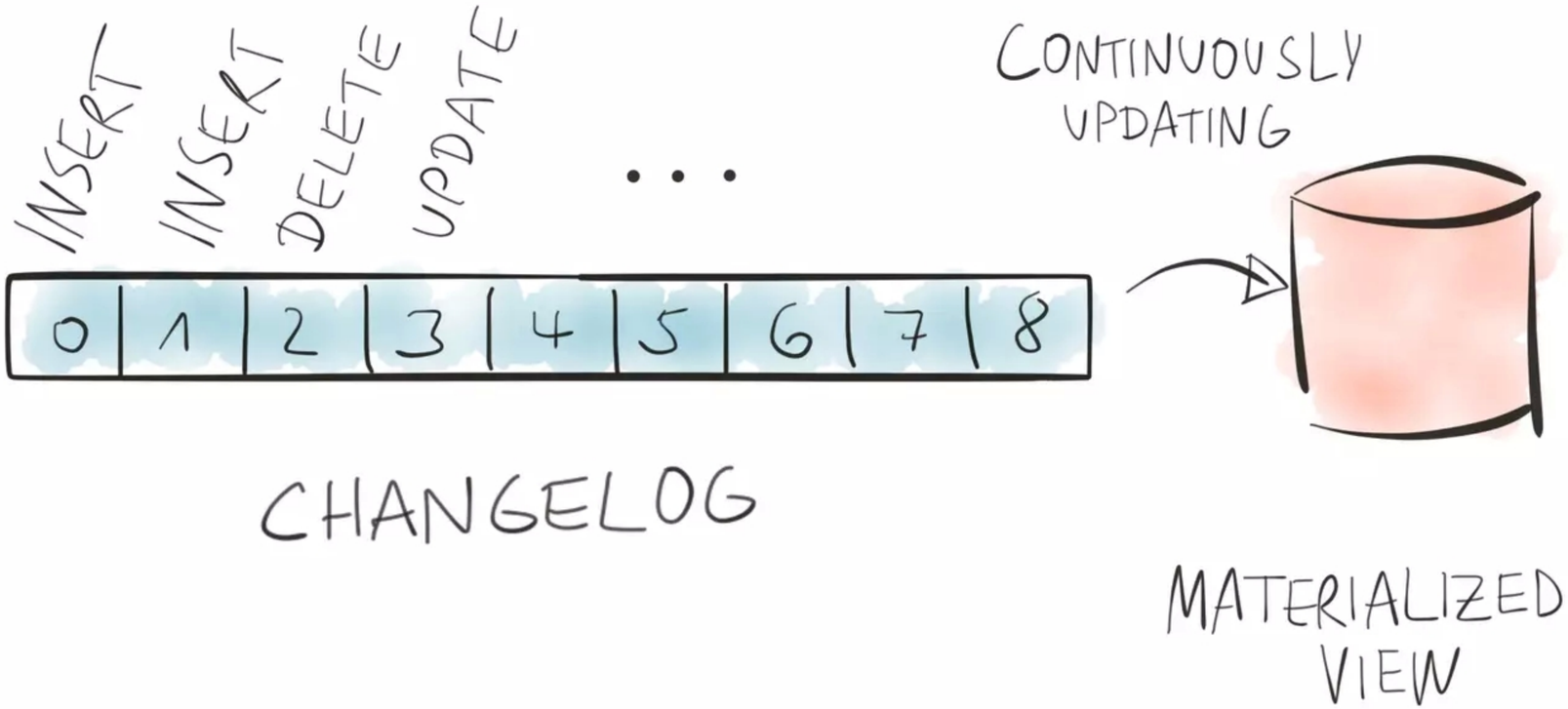
- Order matters: ⋈(s1,s2,s3) != (s1 ⋈ s2) ⋈ s3 != (s1 ⋈ s3) ⋈ s2 != (s2 ⋈ s3) ⋈ s1



* window size=5min

Table Queries

As in regular SQL, however…



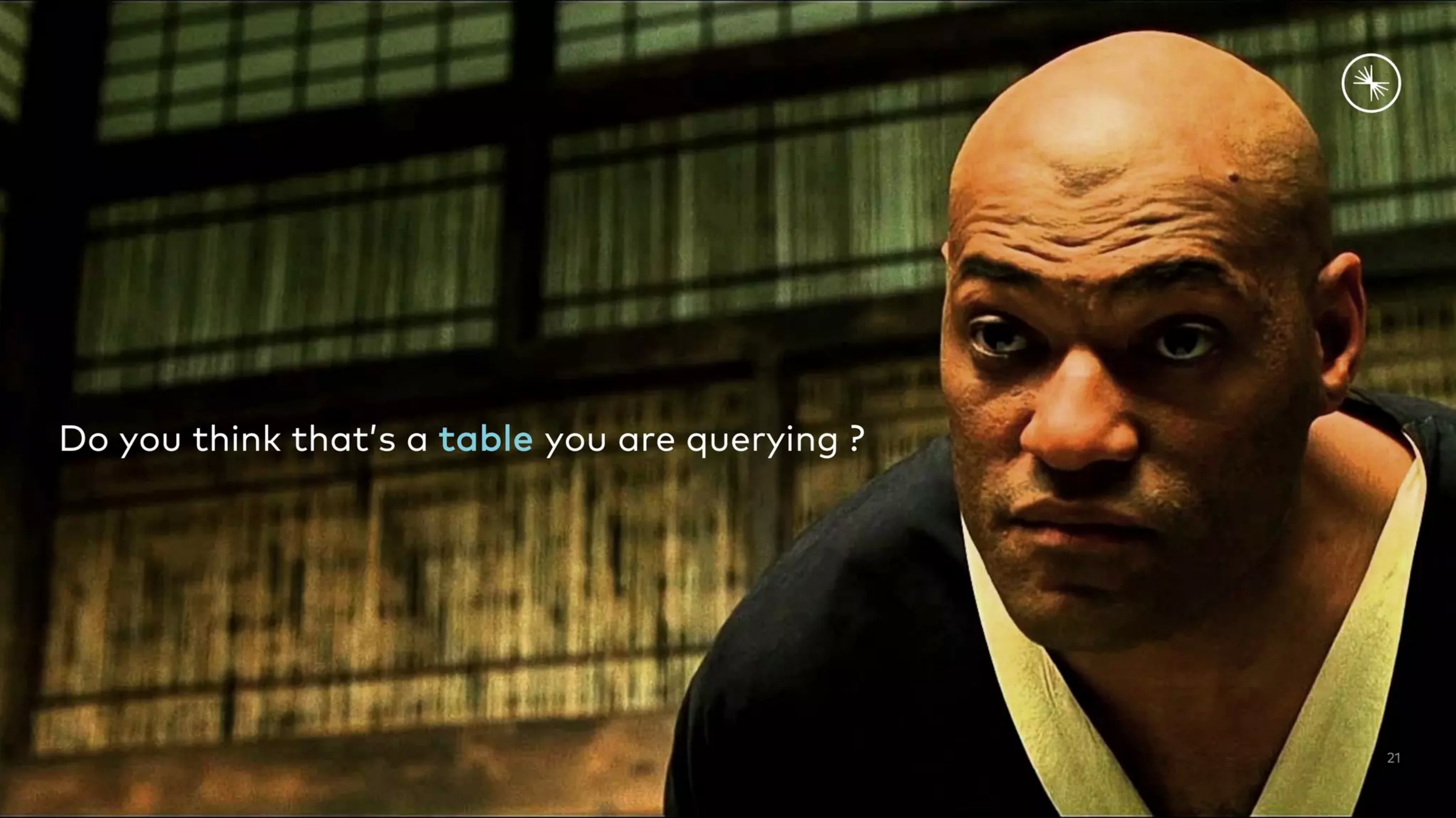
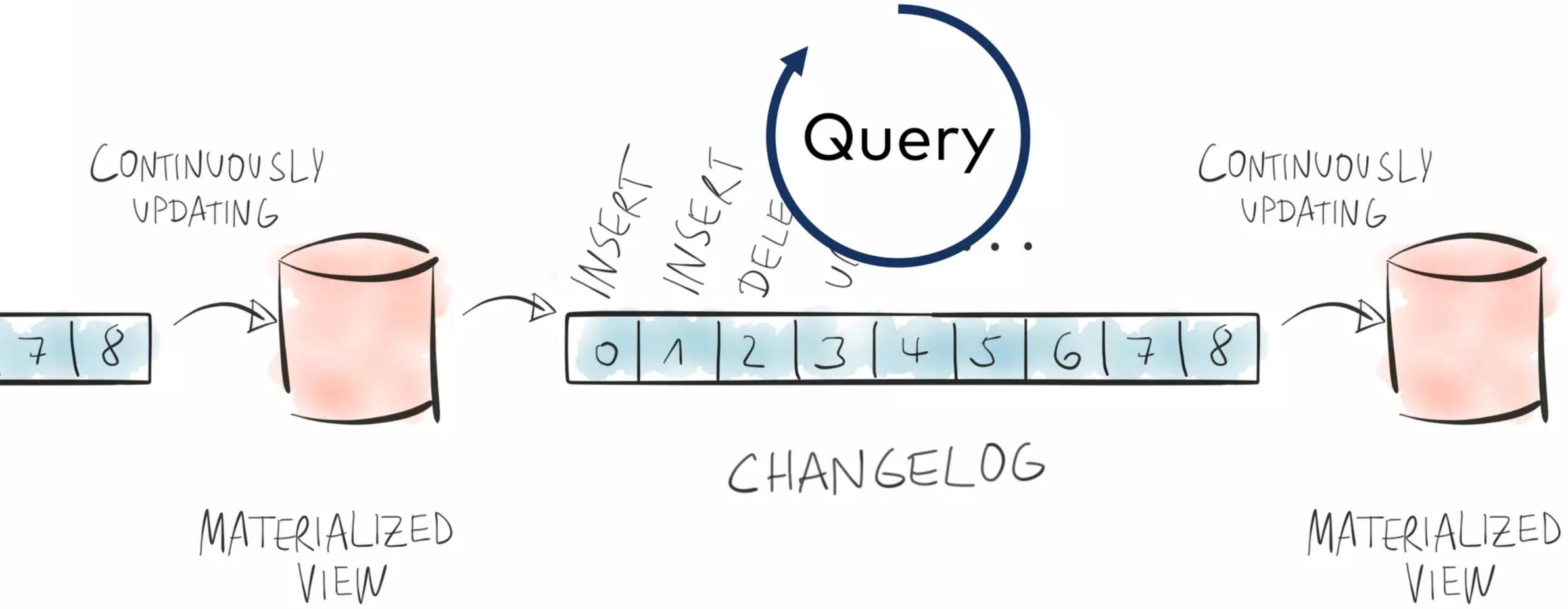
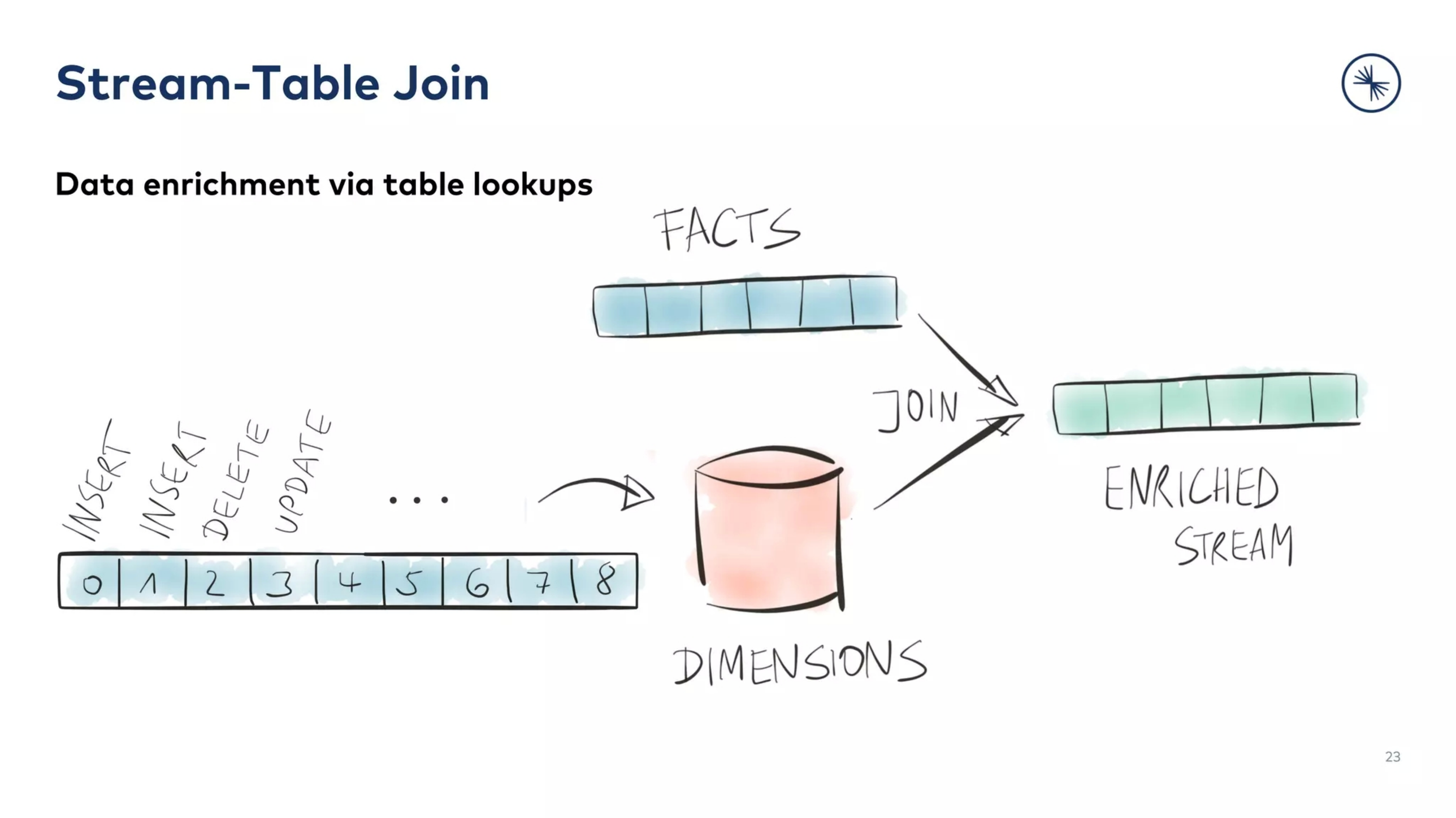

Do you think that's a table you are querying ?

Table Queries

Filters, projections, and aggregations are executed on changelog streams

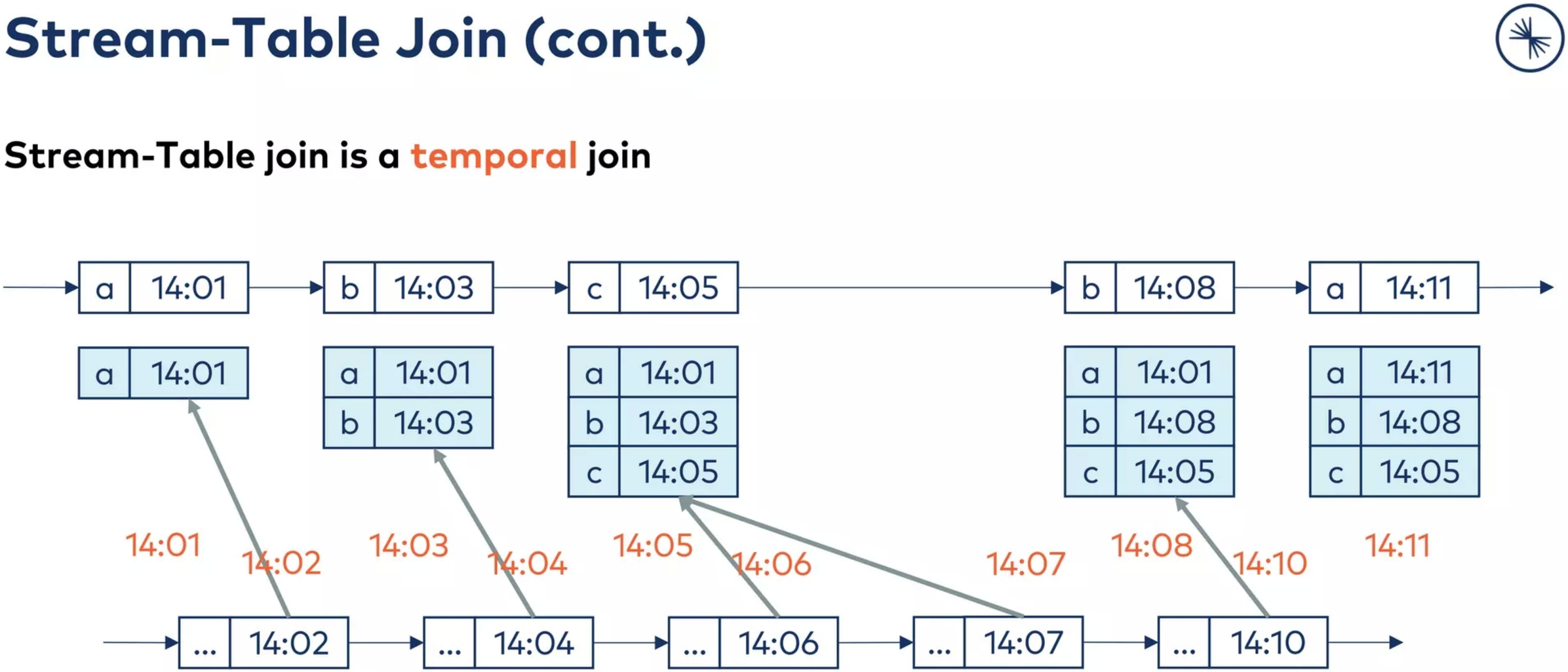


Stream-Table Join



Data enrichment via table lookups

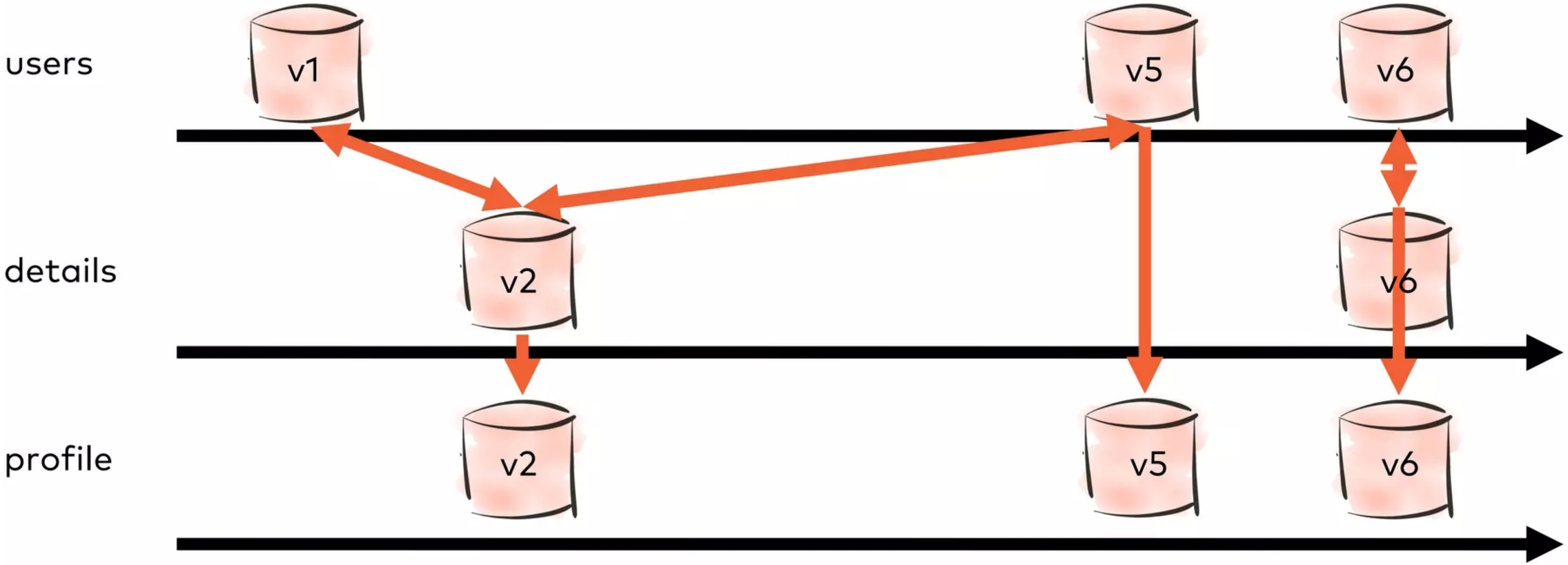
Stream-Table Join (cont.)



Stream-Table join is a temporal join

Table-Table Join

Table-Table join is a temporal join



Runtime: Kafka Streams

Persistent Queries

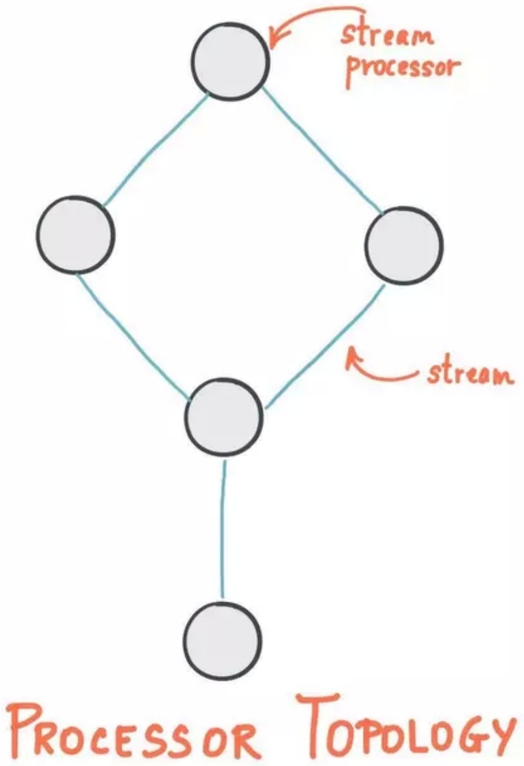
Executed as Kafka Streams programs

Kafka Streams:

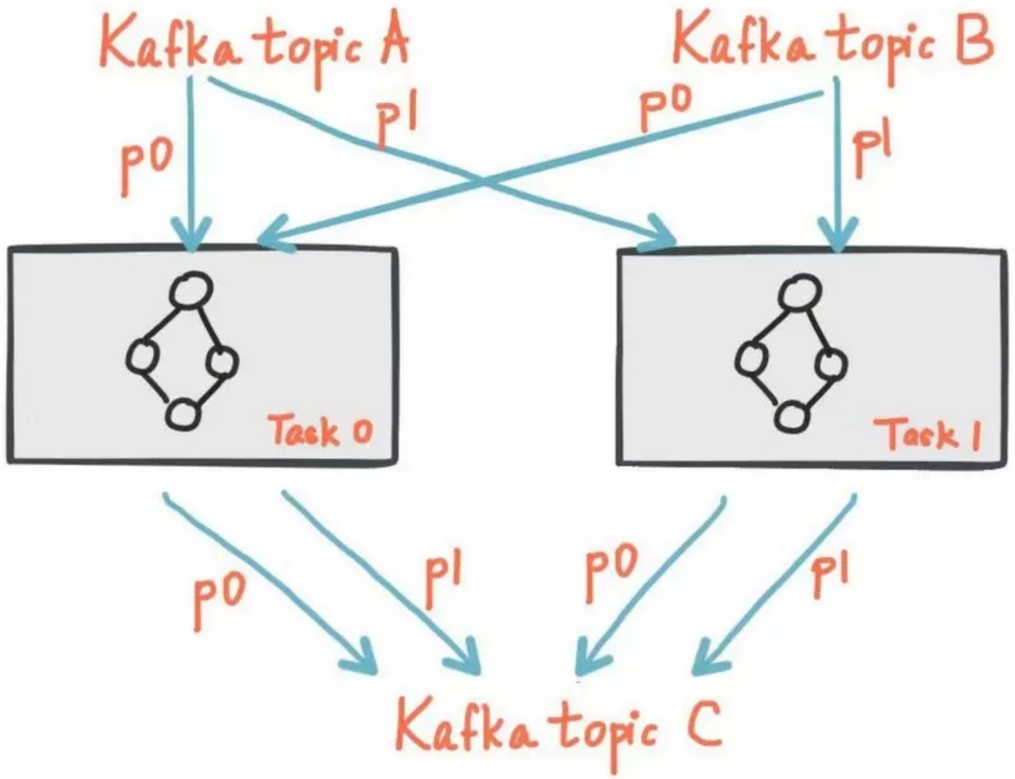
- Java client library (part of the Apache Kafka project)
- High level DSL to process Kafka topics as *KStreams* and *KTables*
  - Executes a dataflow program, represented as a directed graph of operators
  - filter()/map()
  - groupBy()/windowedBy()
  - aggregate()
  - join()
  - Etc.
- Scalable
- Fault-Tolerant

Persistent Queries

Kafka Stream *topologies*:

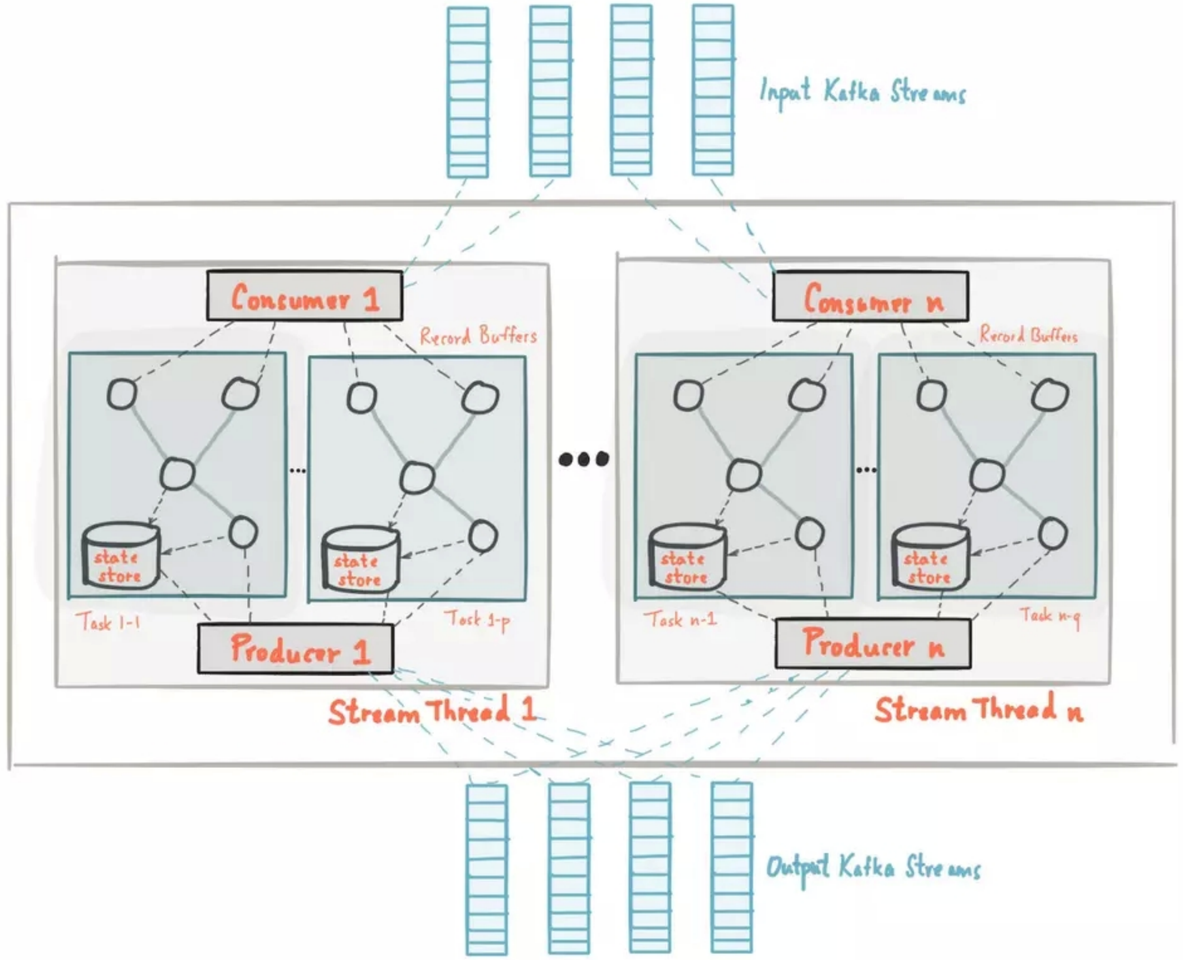


Parallel execution:



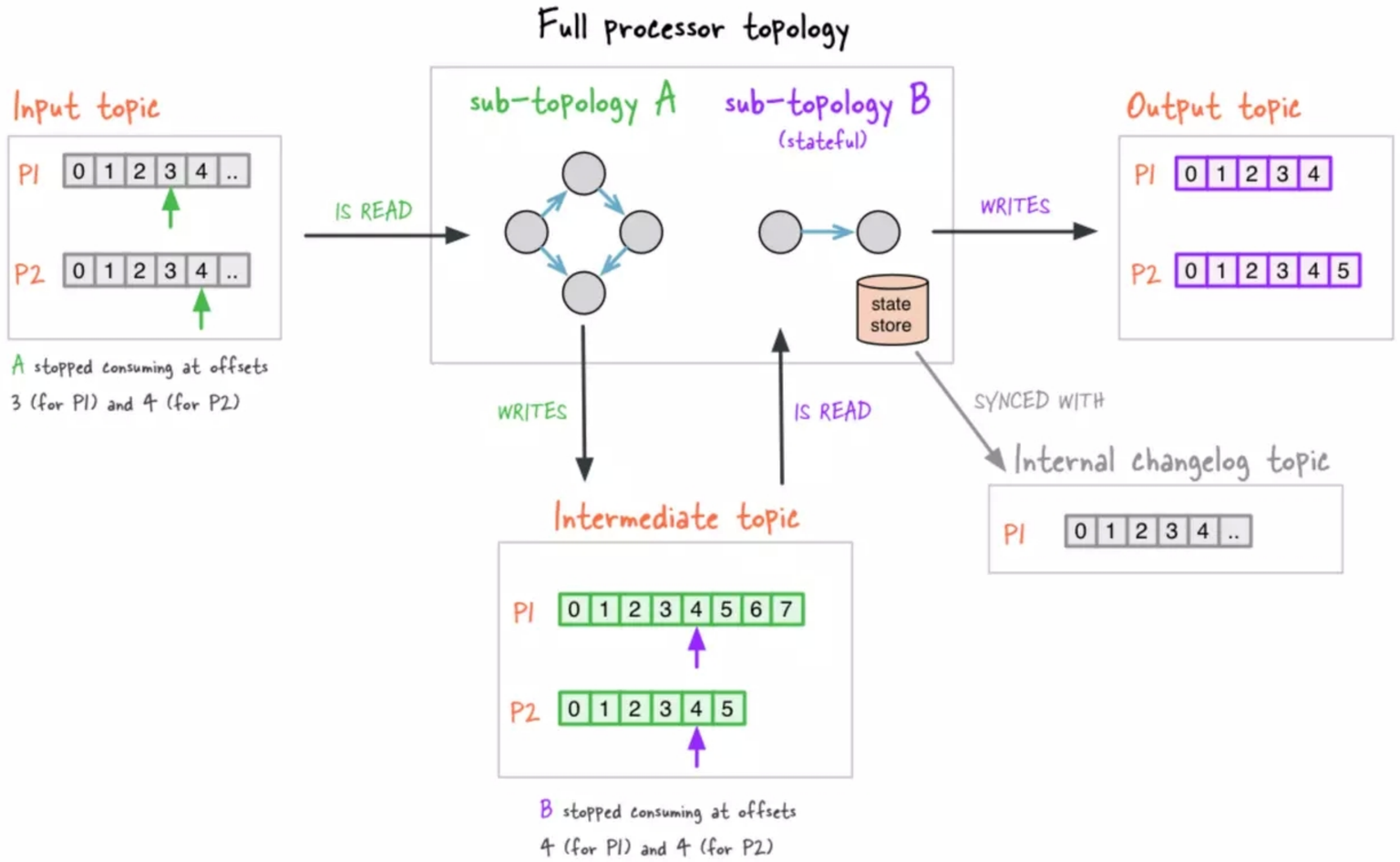
Kafka Streams Internals

Consumers, Producers, and RocksDB



Kafka Streams Internals (cont.)

Data repartitioning and fault-tolerance



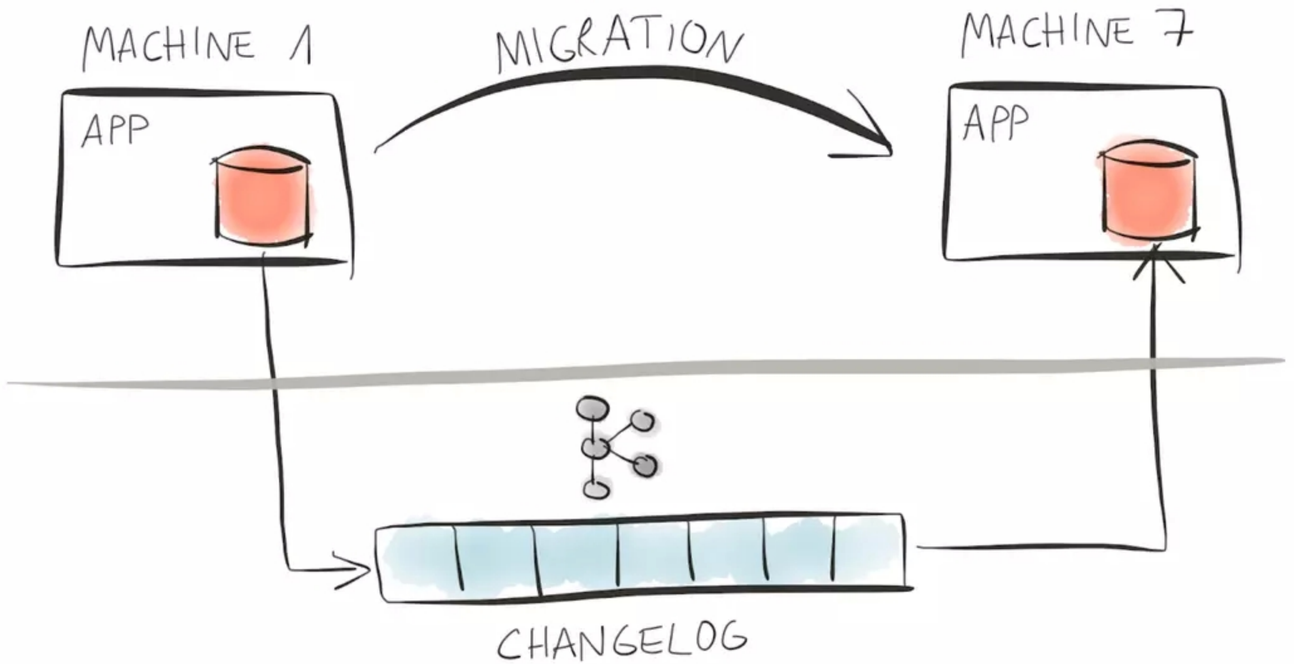
Fault-Tolerance and Hot Standbys

Recovery: replying the changelog topic

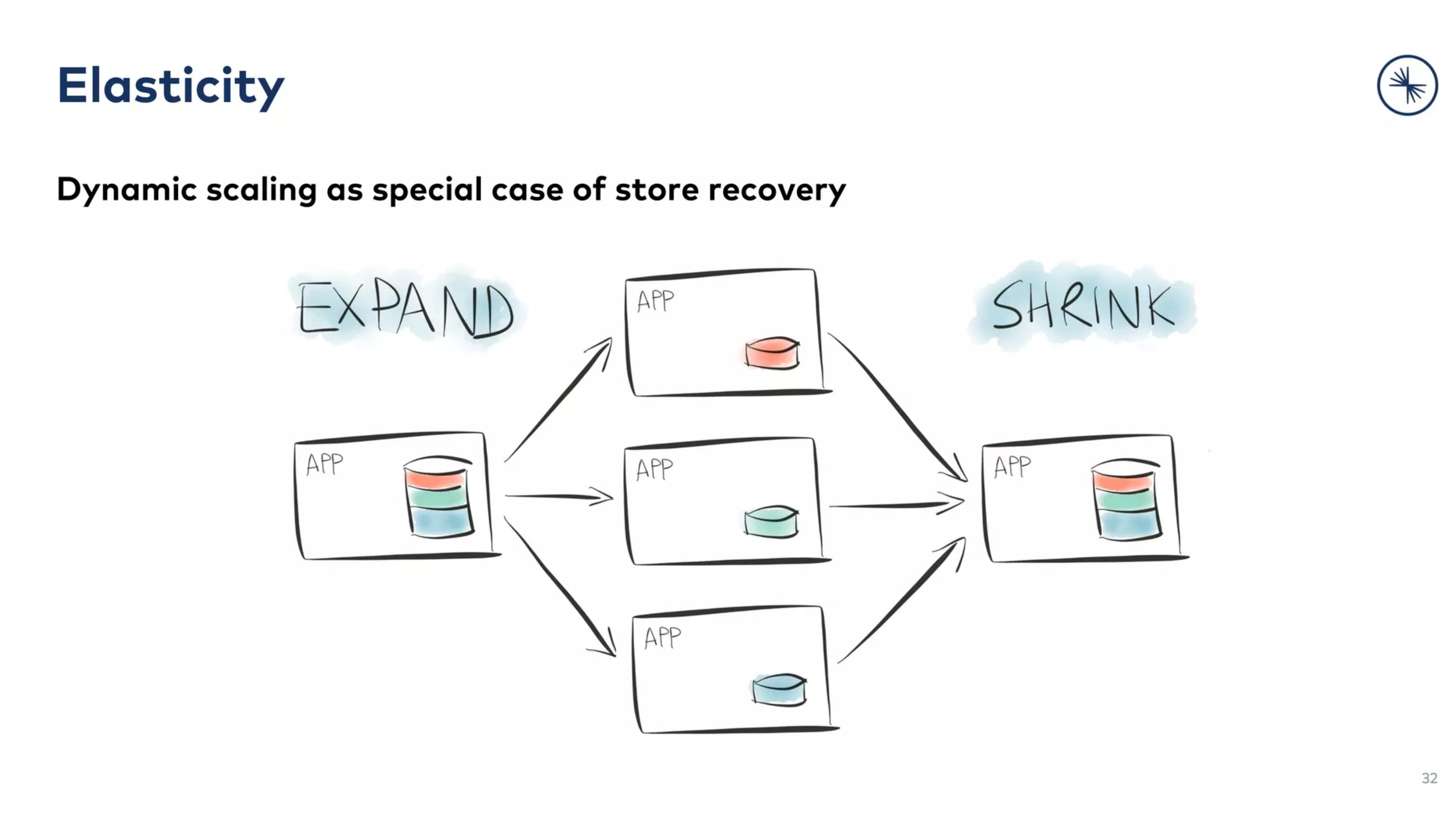
Hot standbys: eagerly replaying the changelog topic

Hot standbys allow for instant fail-over.

Hot standbys allow for HA pull queries.



Elasticity



Dynamic scaling as special case of store recovery

Challenges

Work in Progress

Streaming SQL

- Concise and more powerful language
- Improved time/operator semantics

Consistency guarantees

- ksqlDB is an async system: vector clock approach for improved time tracking

Applications and transient queries

- More efficient and more scalable pull/push queries
- Improved INSERT/UPDATE/DELETE support?

Future Work

Query optimization

- Currently state: rule based
  - filter push down
  - merging of repartition topic
  - merging of input/output/changelog topics
- Streaming cost model and cost-based optimization
- Adaptive re-optimization at runtime
- Query merging/splitting

Runtime improvements:

- Internal and optimized (binary) data format to avoid expensive (de)serialization costs
- Task assignment: load balancing vs stickyness

Richer SQL:

- Sub-query support

References

- KSQL: Streaming SQL Engine for Apache Kafka
  Hojjat Jafarpour, Rohan Desai, Damian Guy
  EDBT '19: Proceedings of the 22nd International Conference on Extending Database Technology, 2019
- Streams and Tables: Two Sides of the Same Coin
  Matthias J. Sax, Guozhang Wang, Matthias Weidlich, Johann-Christoph Freytag
  BIRTE '18: Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics, 2018
- https://kafka.apache.org/books-and-papers
- https://ksqldb.io/

CONFLUENT

Thanks! We are hiring!

@MatthiasJSax

matthias@confluent.io | mjsax@apache.org