OSA CON 24

Low latency Change Data Capture (CDC) to your data lake, using Apache Flink and Apache Paimon

Ali Alemi, Subham Rakshit

alialem@amazon.com, rakssubh@amazon.co.uk

November 19-21, 2024

Subham Rakshit

(he/him)
Senior Streaming Architect
Amazon Web Services

Ali Alemi

(he/him)
Senior Streaming Architect
Amazon Web Services

Agenda

Change Data Capture (CDC) use-cases

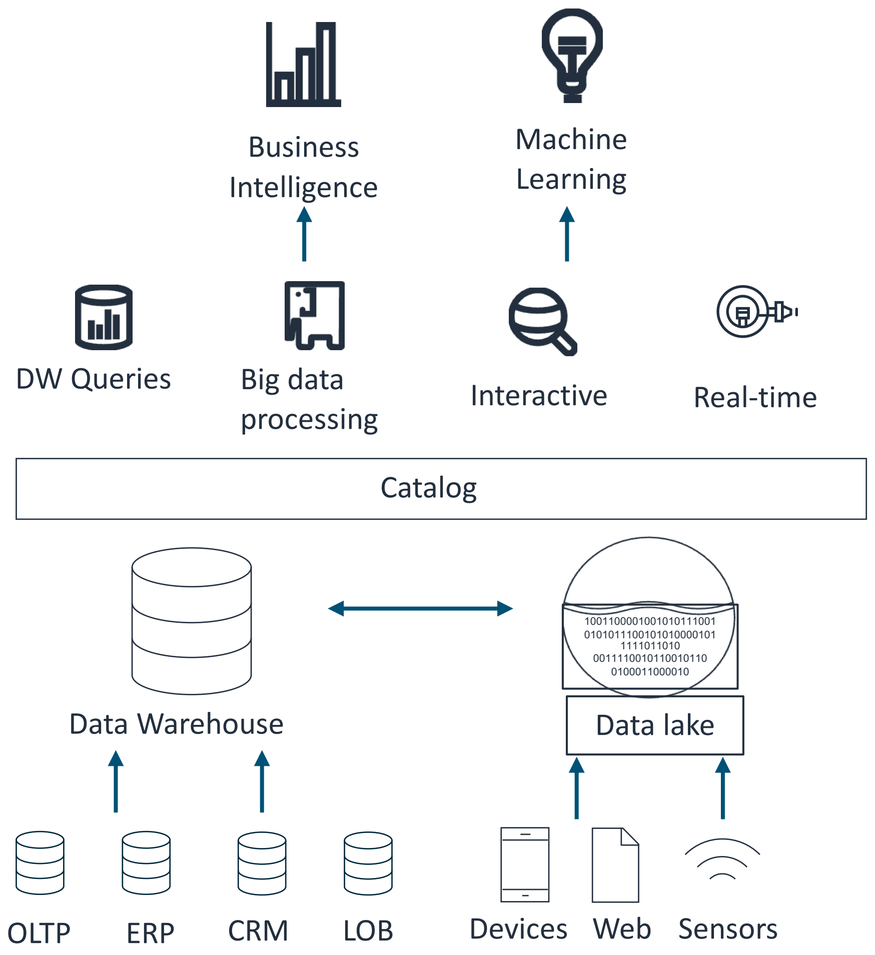
Challenges when handling CDC

Apache Flink CDC and Apache Paimon

How Apache Paimon solves data pipeline challenges?

Conclusion

Data Lakehouse



Data lakes provide:

Relational and non-relational data

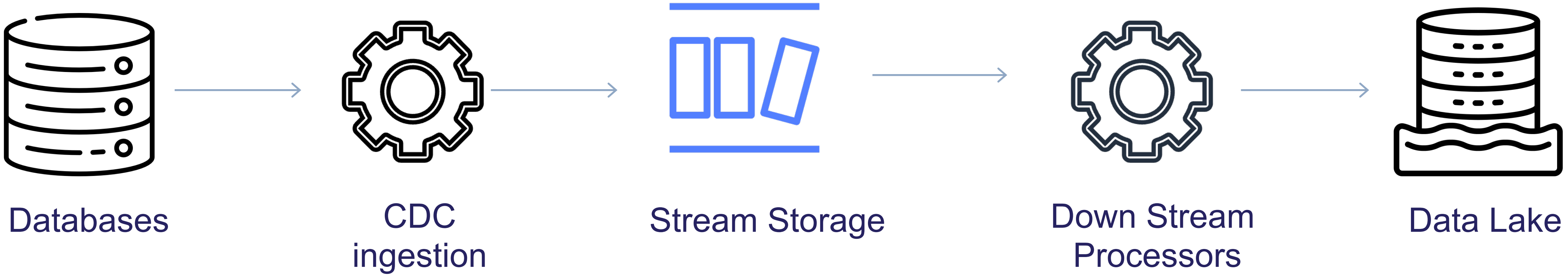
Scale-out to EBs

Diverse set of analytics and machine learning tools

Work on data without any data movement

Designed for low cost storage and analytics

CDC to a message queue



Challenges when handling CDC data

Data volume

Complexity

Data Quality

Integrating CDC data with other Data

Consistency

Data security and compliance

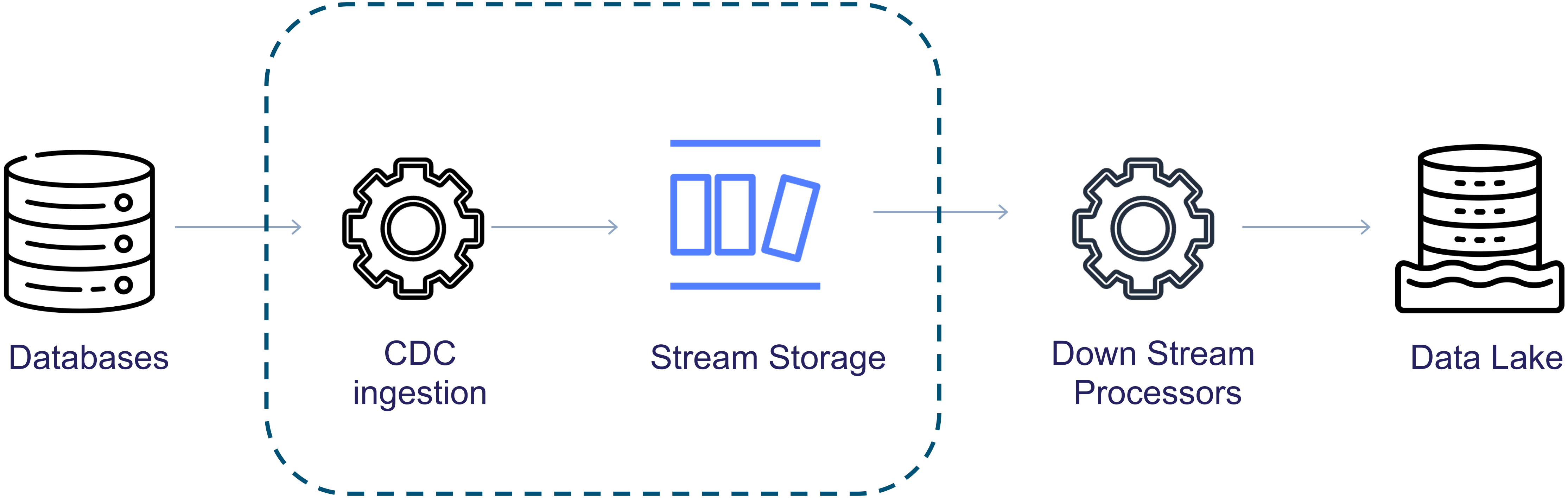
Why Apache Flink for CDC?

Apache Flink

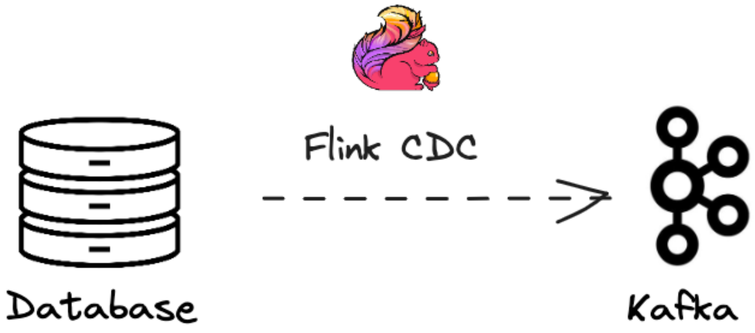
- Can consume changelog from various sources across variety of database engines: (MySQL, PostgreSQL, SQL Server, and more)
- Reading snapshots, and transaction logs
- Provides Dynamic Table as one abstract for:
  - Temporal processing
  - CDC data operations (Insert, Update, Delete)
  - Bounded Unbounded
- Exactly-once semantics
- Data delivery to various targets, including file system (including: Data lake)

CDC Patterns

CDC to a message queue



CDC to Apache Kafka topic



1. CREATE TABLE source_table… AS (…)
2. CREATE TABLE target_table…AS (…)
3. INSERT INTO target_table SELECT * from

   source_table;

Pros –

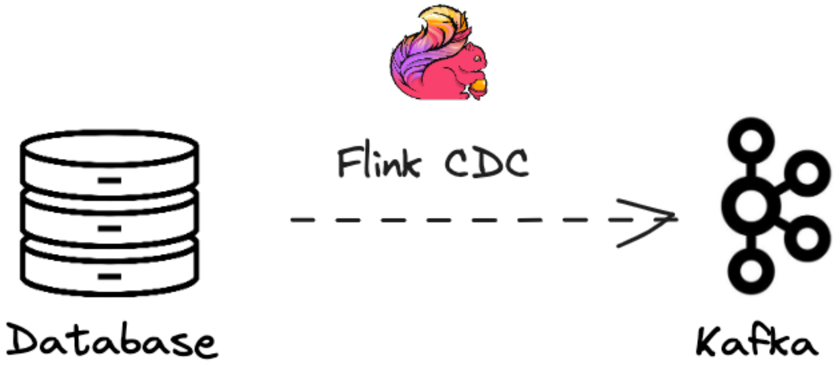
- Low Latency
- Horizontal scalability with Kafka

Challenges –

- Schema Evolution
- Kafka Storage
- Kafka capacity during backfill
- Retention

KAFKA is a registered trademark of The Apache Software Foundation and has been licensed for use by AWS. AWS has no affiliation with and is not endorsed by The Apache Software Foundation.

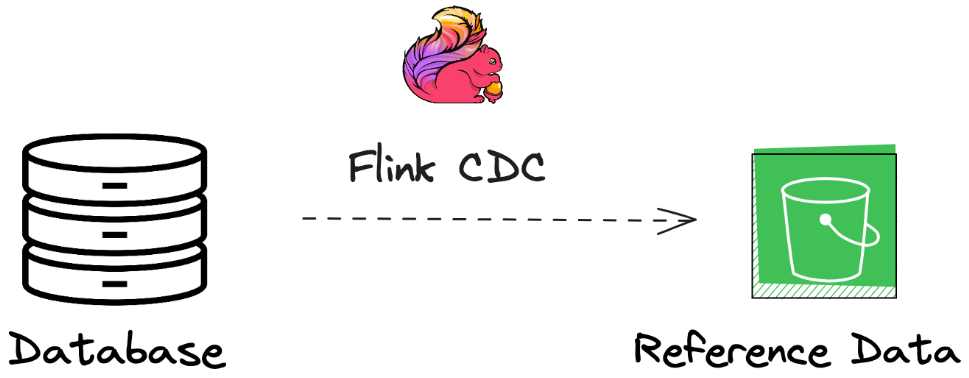
CDC to Apache Kafka topic



- New snapshot when Flink state is lost
- Data inconsistency between source, and Kafka
- Standard topic deletes data after retention
- Compacted topics keeps data forever
- Spike in storage during the backfill
- Separate topic per each table

KAFKA is a registered trademark of The Apache Software Foundation and has been licensed for use by AWS. AWS has no affiliation with and is not endorsed by The Apache Software Foundation.

More cost effective, scalable approach



1. CREATE TABLE source_table... AS (...)
2. CREATE TABLE target_table...AS (...)
3. INSERT INTO target_table SELECT * from source_table;

Pros –

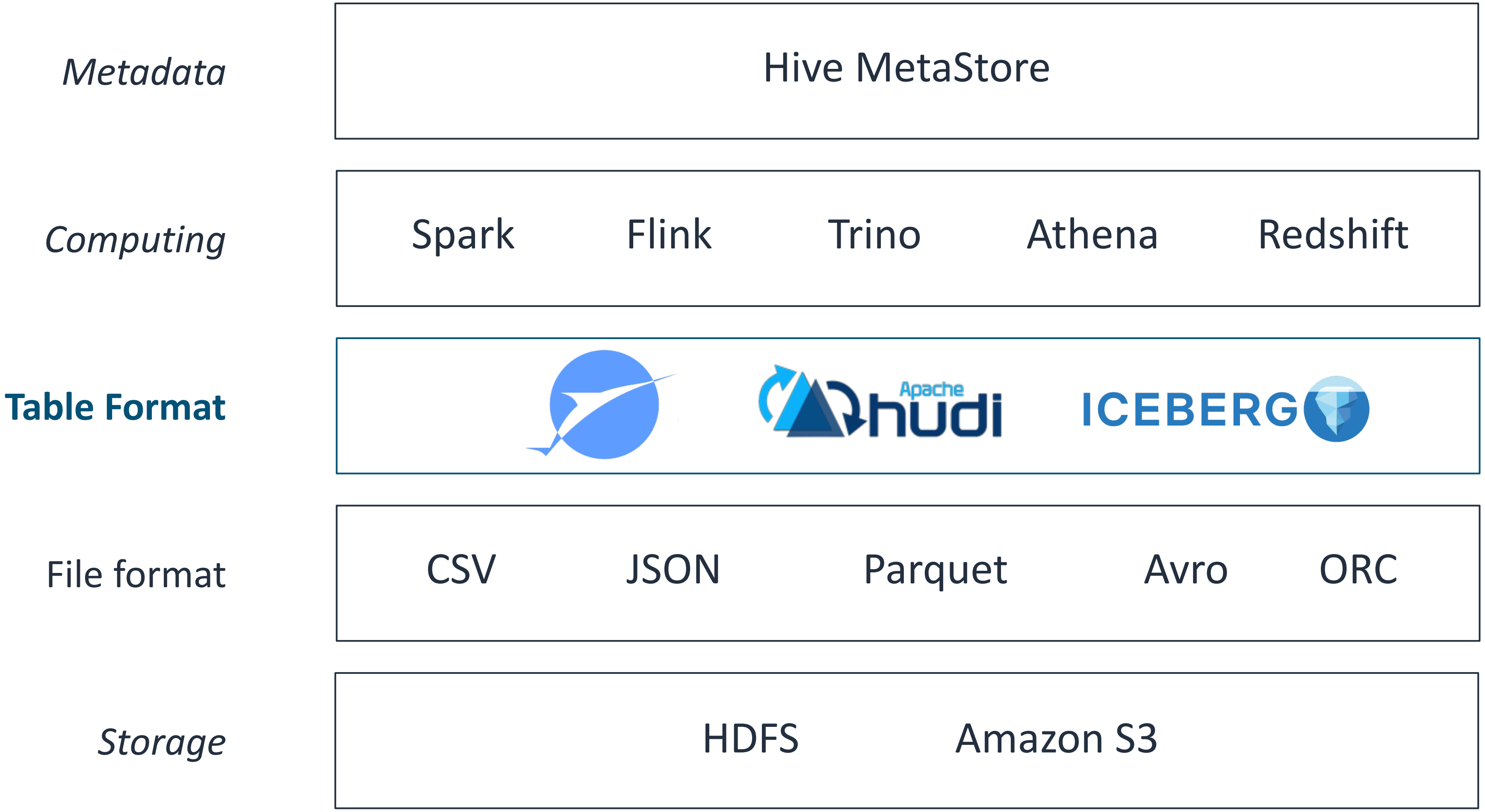
- Scalable storage
- Reduce cost
- Infinite retention

Challenges -

- Higher Latency compared to Kafka
- UPSERT / Delete
- Streaming read
- Schema evolution

Open Table Formats

Modern Data lakes use Open Table Format



Metadata

Hive MetaStore

Computing

Spark Flink Trino Athena Redshift

Table Format

Apache hudi ICEBERG

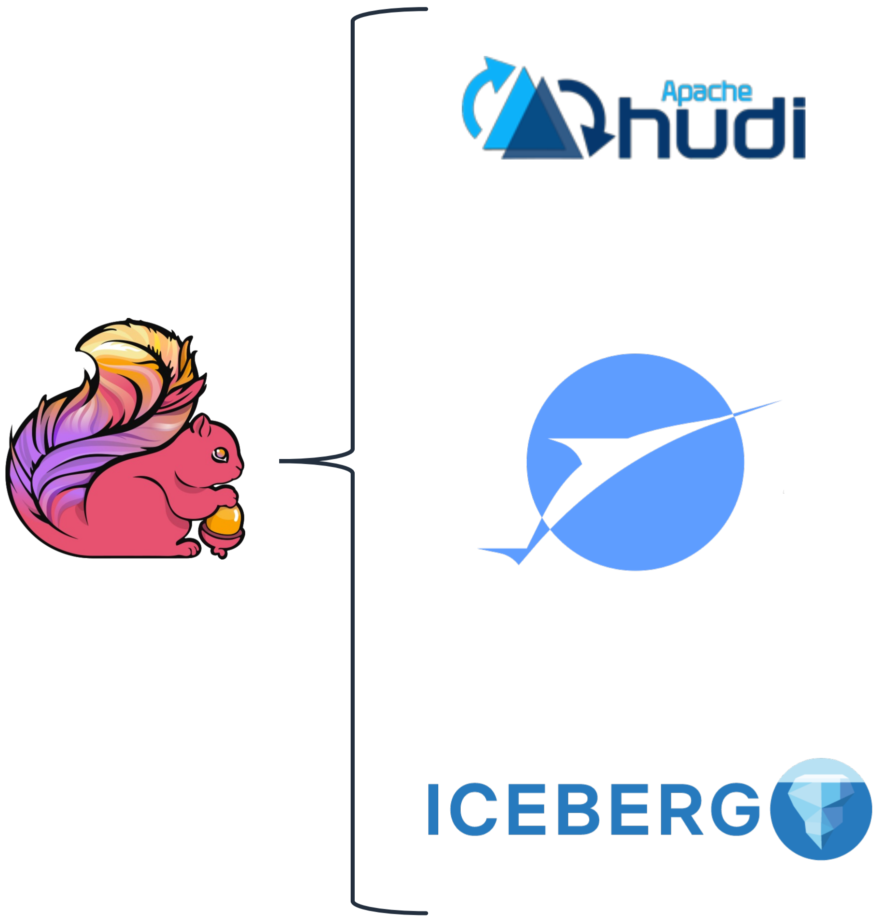
File format

CSV JSON Parquet Avro ORC

Storage

HDFS Amazon S3

CDC to data lake using Apache Flink



```
INSERT INTO “target_table”
SELECT * from “source_table”;
```

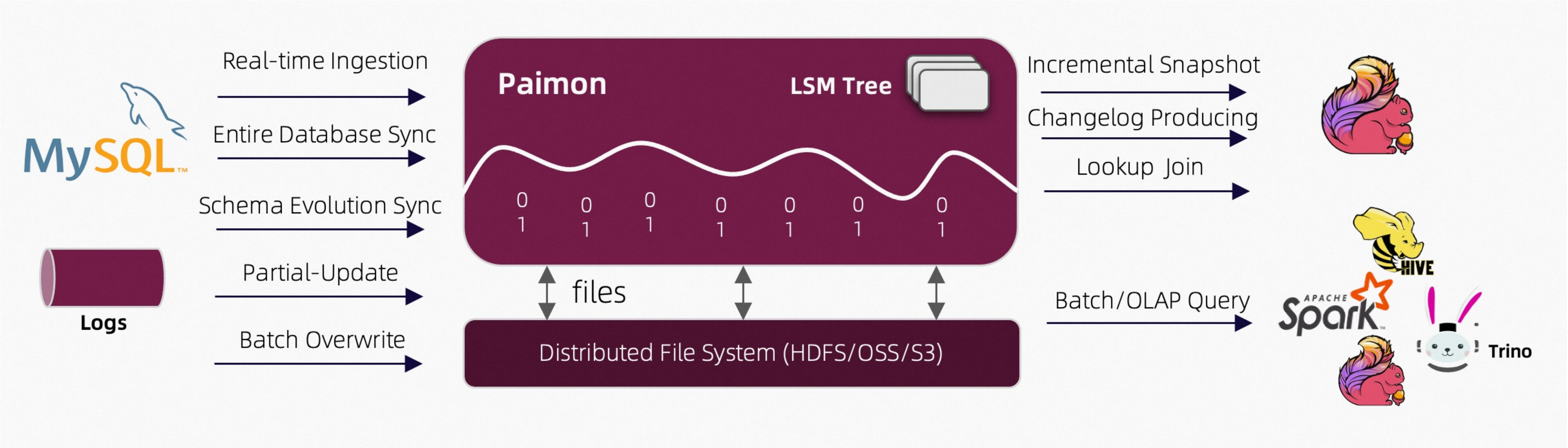
```
INSERT INTO “target_table”
SELECT * from “source_table”;
```

```
INSERT INTO “target_table”
SELECT * from” “source_table”
/*+ OPTIONS('upsert-enabled'='true') */
```

What is Apache Paimon ?

Apache Paimon

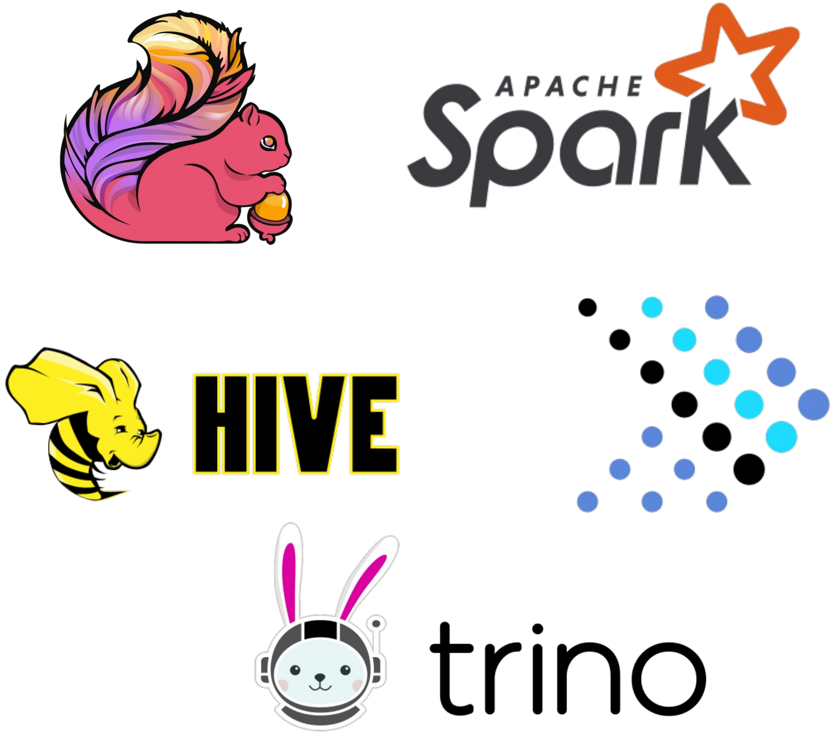
Apache Paimon is a streaming data lake platform that supports high-speed data ingestion, change data tracking and efficient real-time analytics.



Apache Paimon supports…

File Systems

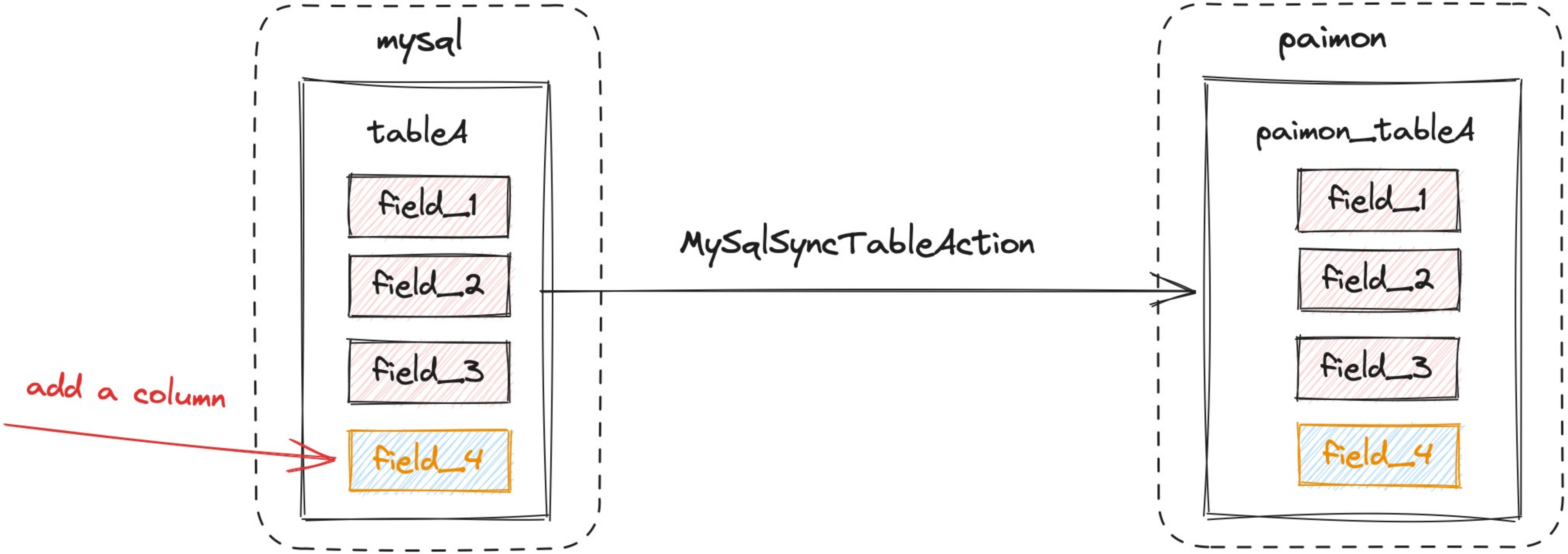
Engines



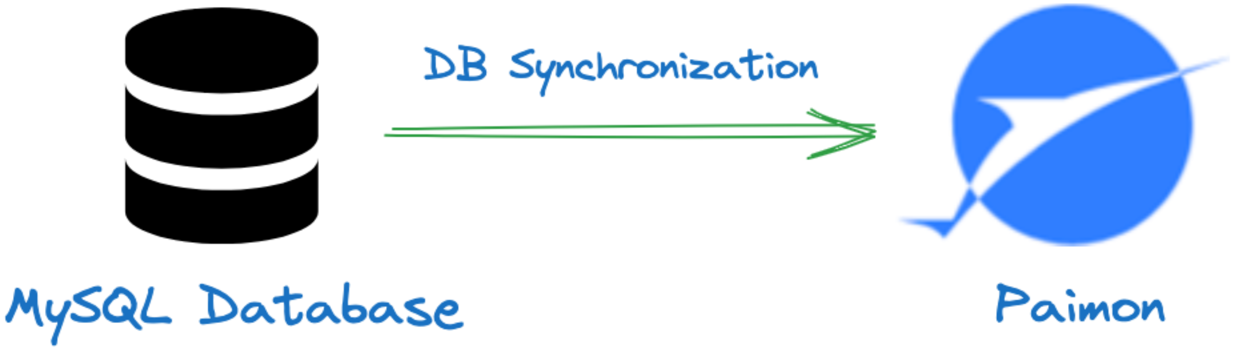
ICEBERG Readers

How do we handle schema changes ?

What is schema evolution?



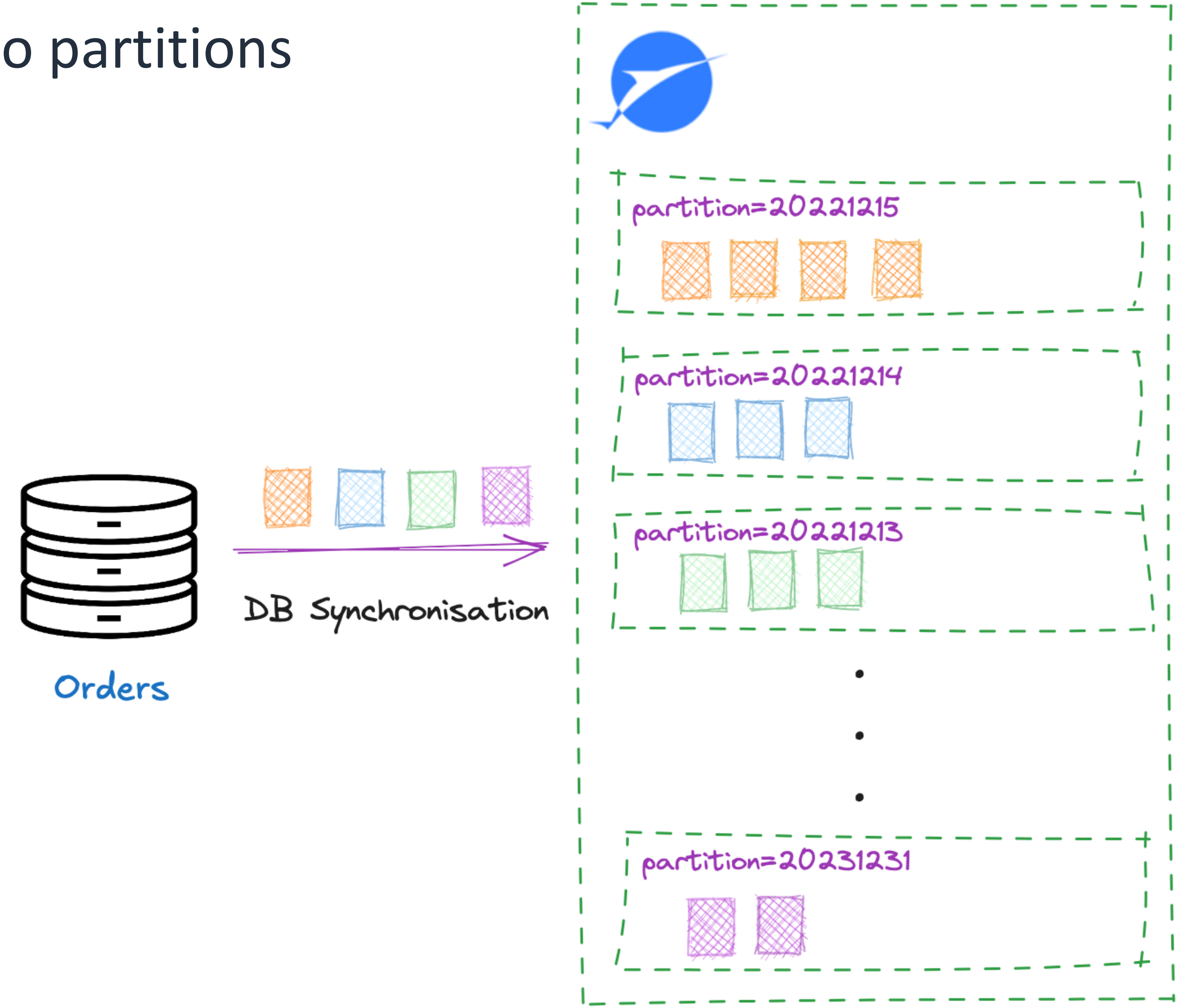
Apache Paimon Actions



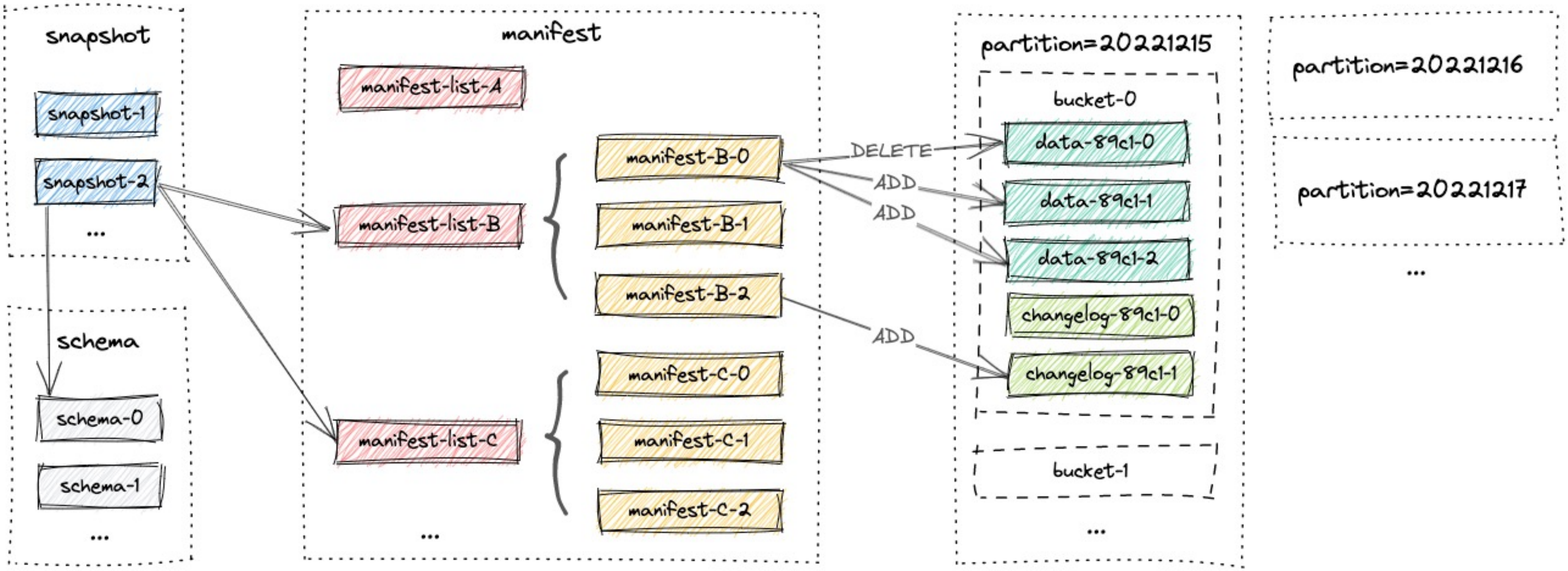
- Synchronisation from MySQL, Kafka, Pulsar and MongoDB
- Supports table/collection/database synchronization
- One Flink Job sinks all table
- Support Schema Evolution
- New tables are created and synced automatically
- Supports UPSERT for PK table
- Low cost for large number of small tables

CDC requires small file writes, random file reads, and many re-writes

CDC write to partitions

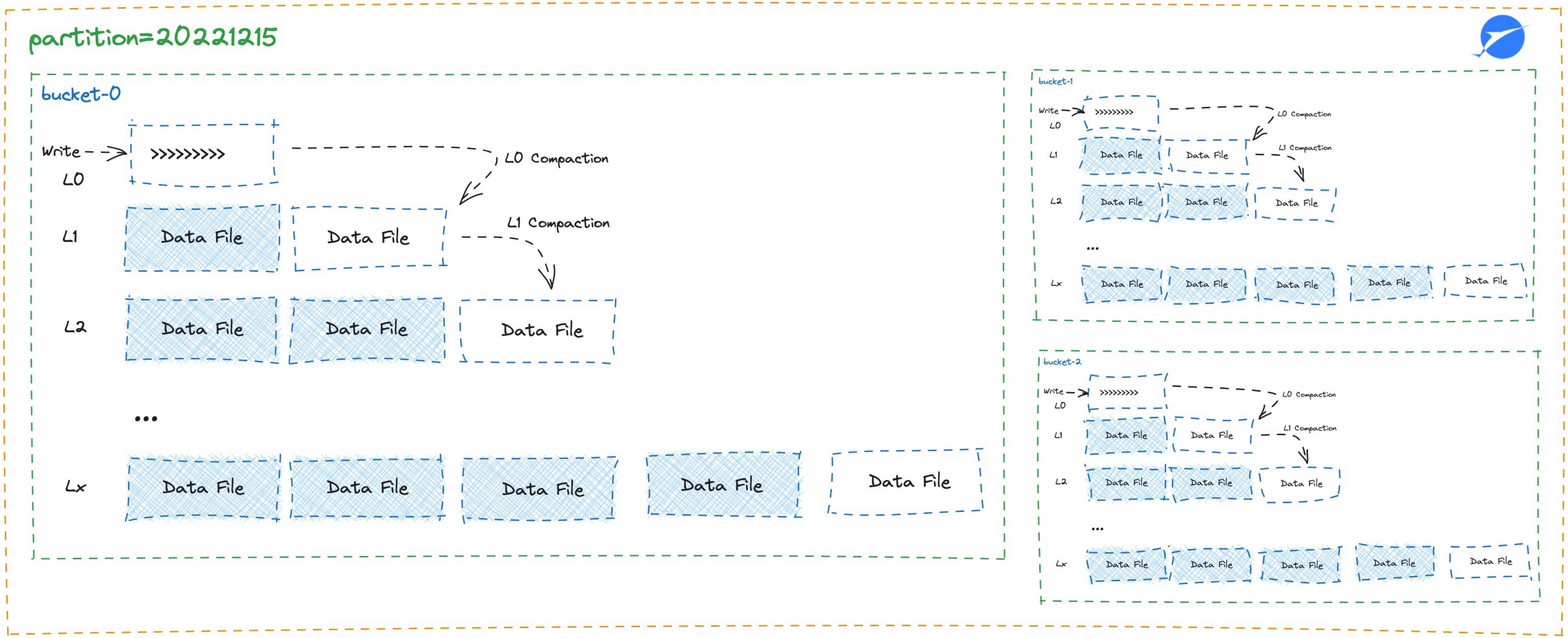


Apache Paimon File Layouts

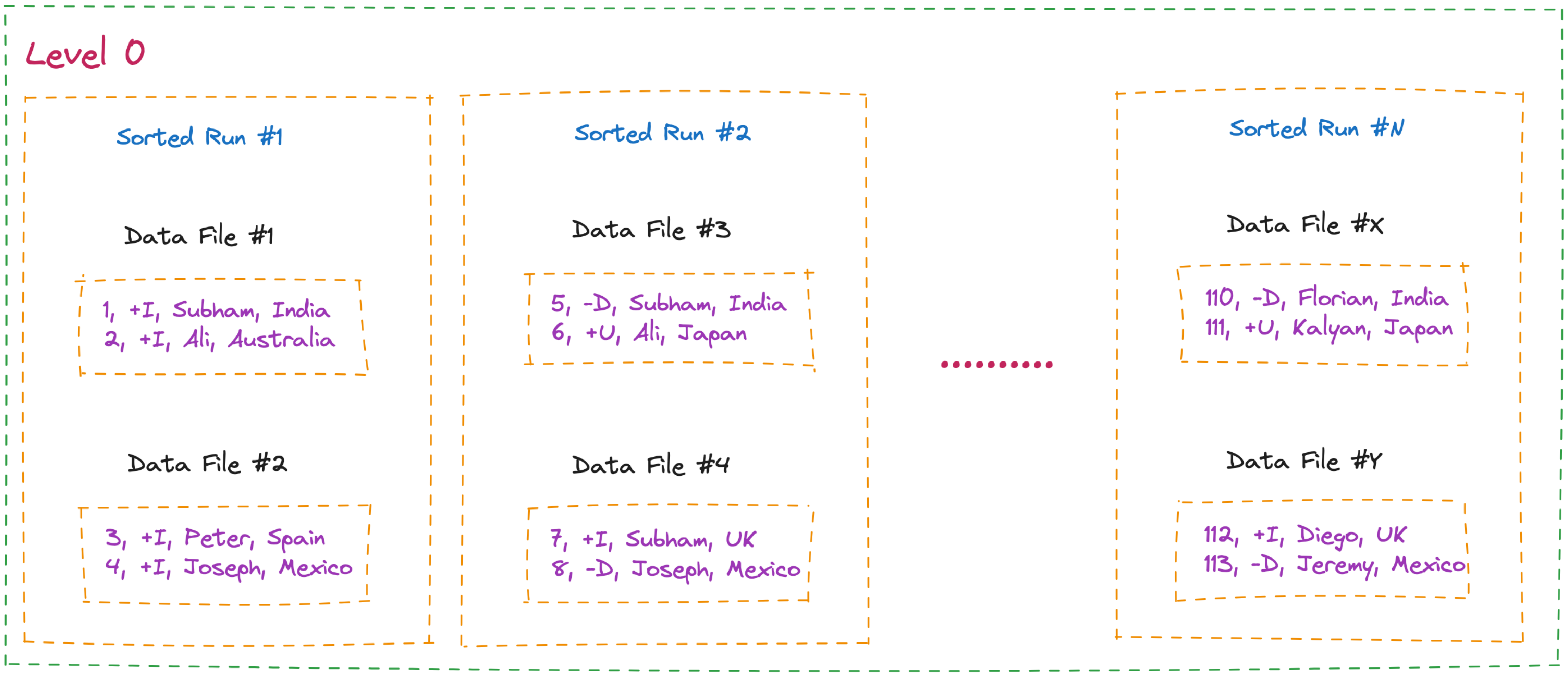


https://paimon.apache.org/docs/master/concepts/basic-concepts/

Partition and Bucket



Level 0 Sorted Runs



Level 0

Sorted Run #1

Data File #1

1, +I, Subham, India
2, +I, Ali, Australia

Data File #2

3, +I, Peter, Spain
4, +I, Joseph, Mexico

Sorted Run #2

Data File #3

5, -D, Subham, India
6, +U, Ali, Japan

Data File #4

7, +I, Subham, UK
8, -D, Joseph, Mexico

....

Sorted Run #N

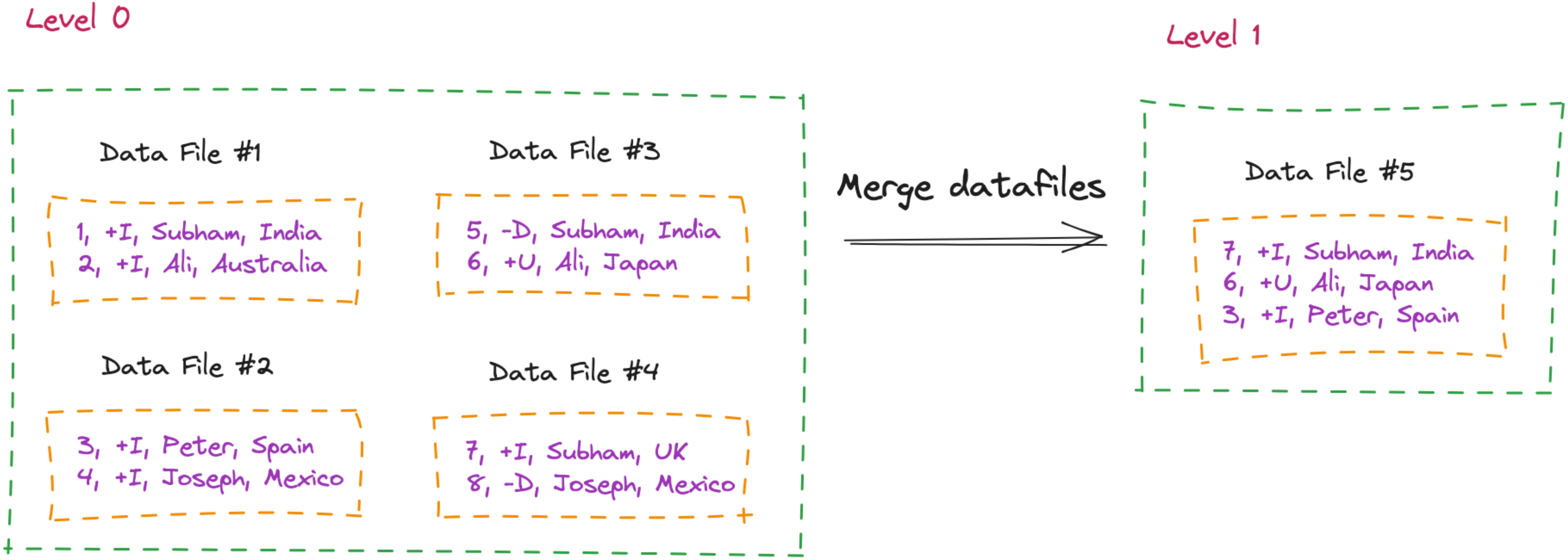
Data File #X

110, -D, Florian, India
111, +U, Kalyan, Japan

Data File #Y

112, +I, Diego, UK
113, -D, Jeremy, Mexico

Compaction



Level 0

Data File #1

1, +I, Subham, India
2, +I, Ali, Australia

Data File #2

3, +I, Peter, Spain
4, +I, Joseph, Mexico

Data File #3

5, -D, Subham, India
6, +U, Ali, Japan

Data File #4

7, +I, Subham, UK
8, -D, Joseph, Mexico

Merge datafiles

Level 1

Data File #5

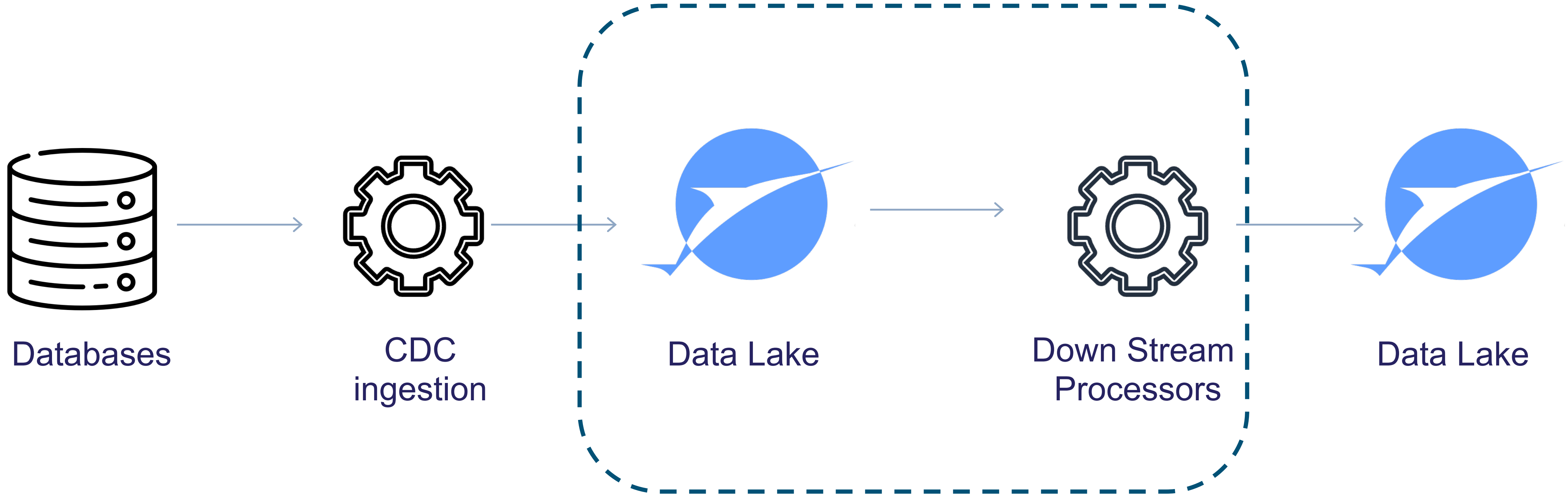
7, +I, Subham, India
6, +U, Ali, Japan
3, +I, Peter, Spain

Compaction

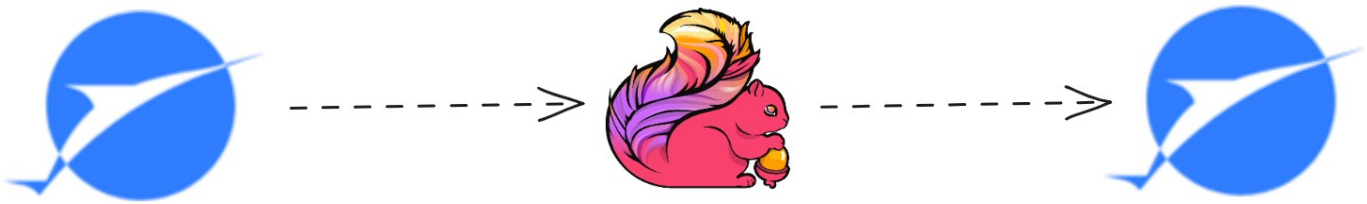
	Same job compaction	Dedicated compaction Job
Apache hudi	✓	✓
	✓	✓
ICEBERG	✗	✓

Streaming Read from Apache Paimon

Read from Apache Paimon



Downstream ETL



1. CREATE TABLE source_table… AS (…)
2. CREATE TABLE target_table…AS (…)
3. INSERT INTO target_table SELECT * from

   source_table;

- Streaming read
- Read from Consumer Offset
- Configure Consumer-ID

Read from Consumer ID

```
SELECT * FROM word_count /*+ OPTIONS(
    'consumer-id' = 'myconsumer-1’,
    'consumer.expiration-time' = '60000000'
) */;
```

Read from Consumer Offset

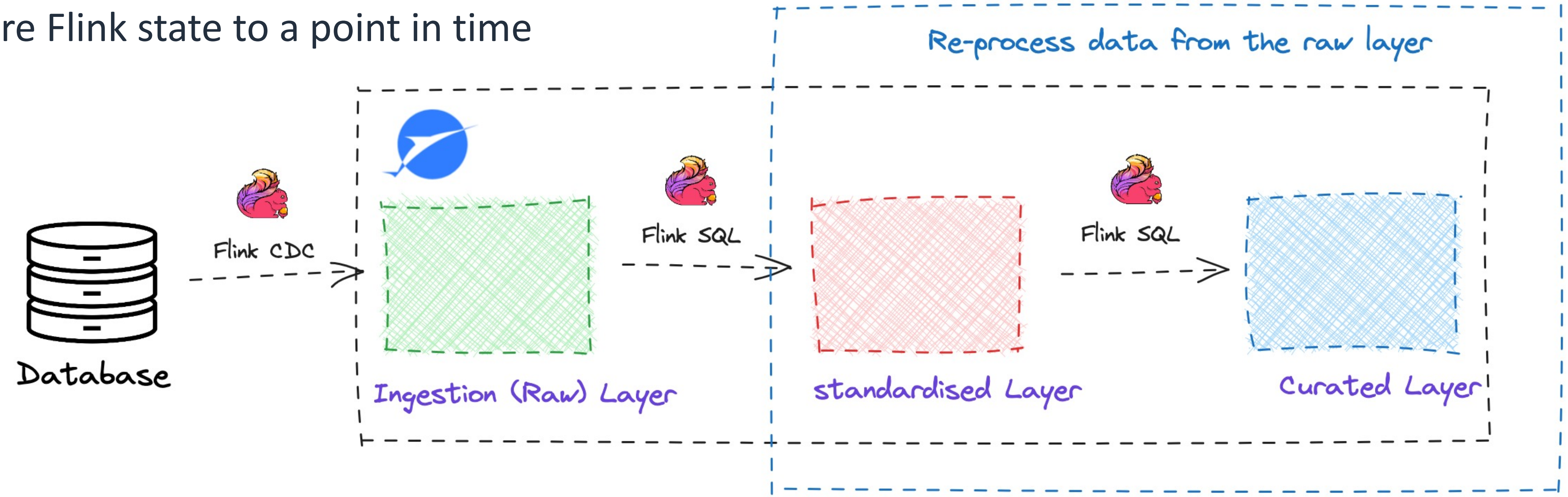
```
SELECT * FROM word_count /*+ OPTIONS(
   'scan.mode' = 'latest') */;
```

```
SELECT * FROM word_count /*+ OPTIONS(
   'scan.timestamp-millis' = '1678883047356'
) */;
```

Data backfill - when and how?

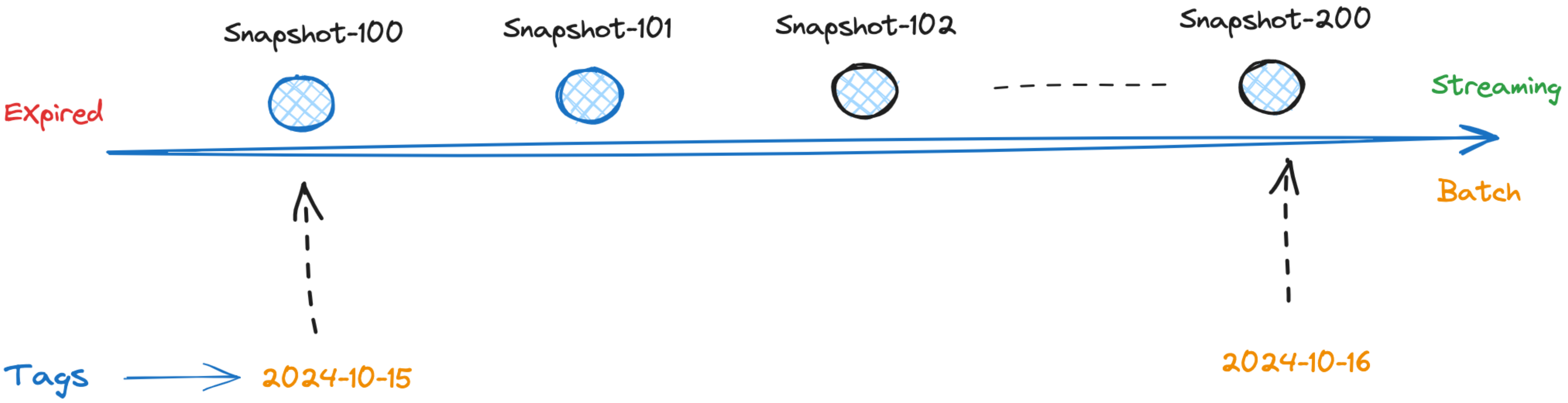
Backfill

- Historical Data correction
- Business logic changes
- Restore downstream table to a point in time
- Restore Flink state to a point in time

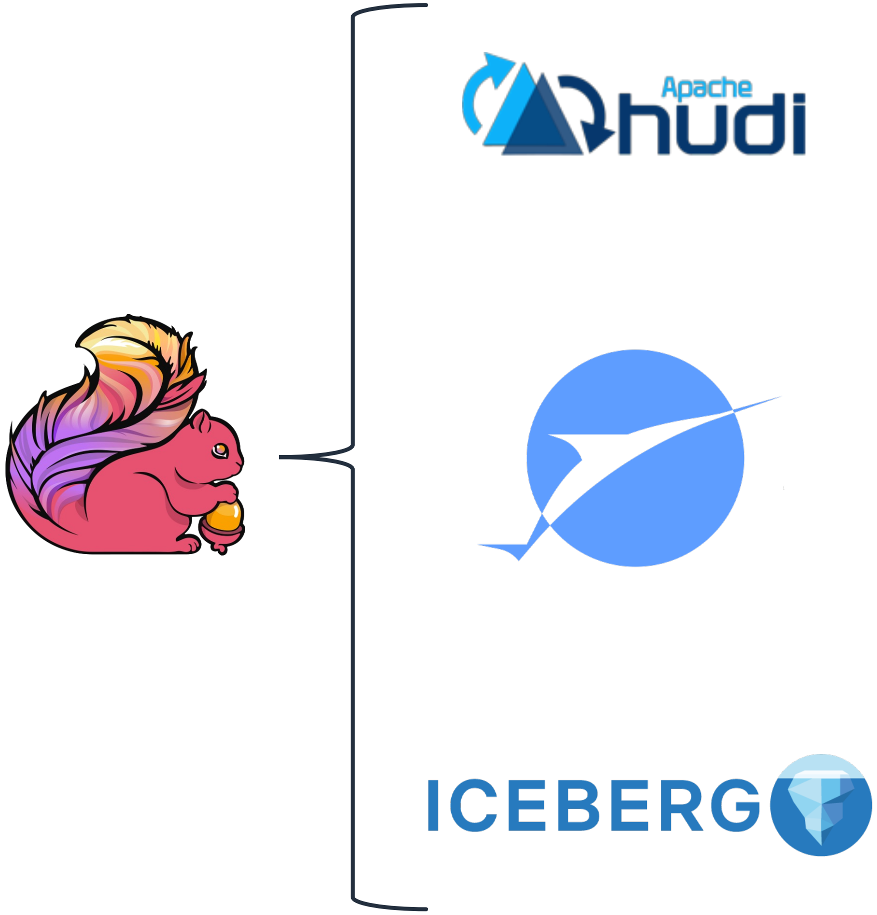


Apache Paimon Tags

- Paimon allows creation of tags to query data from previous snapshots.
- Tag contain the manifest and data files of a snapshot
- Tags can be automatically created and expired.
- You can also rollback a table to a specific tag



Read patterns



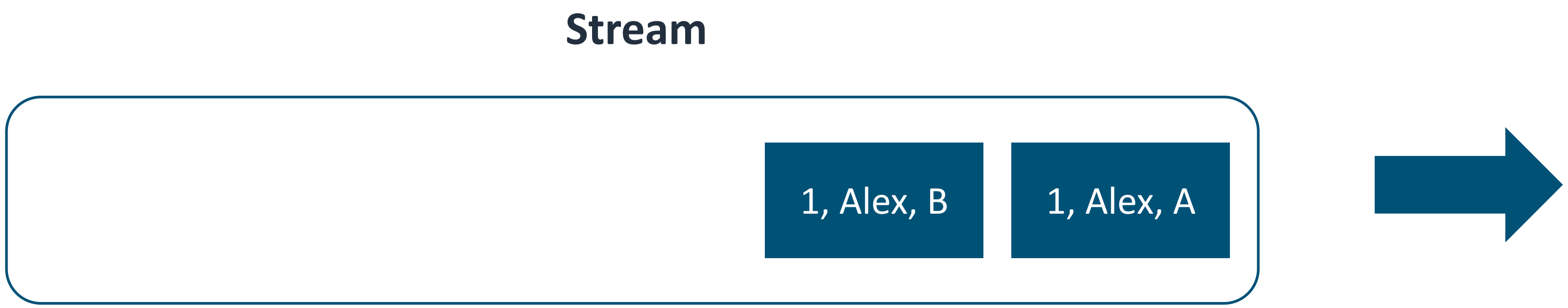
Start / End commit time or EARLIEST

Timestamp, Snapshot_id, tag, branch, consumer-id

Snapshot_id, tag, branch

What can a robust merge engine do?

Merge engine - Deduplicate



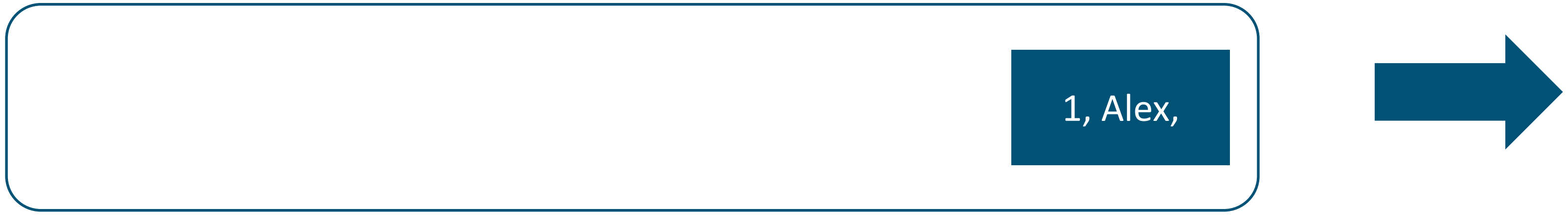
Sink Operator

Key	name	email
1	Alex	B
2	Sam	Z
3	David	X

Merge engine - Partial Update

```
'merge-engine' = 'partial-update’
```

Stream



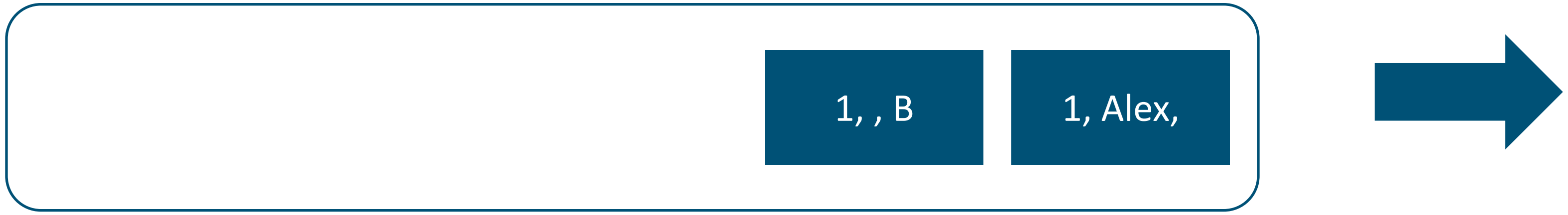
Sink Operator

Key	name	email
1	Alex	
2	Sam	Z
3	David	X

Merge engine – Partial Update

```
'merge-engine' = 'partial-update'
```

Stream



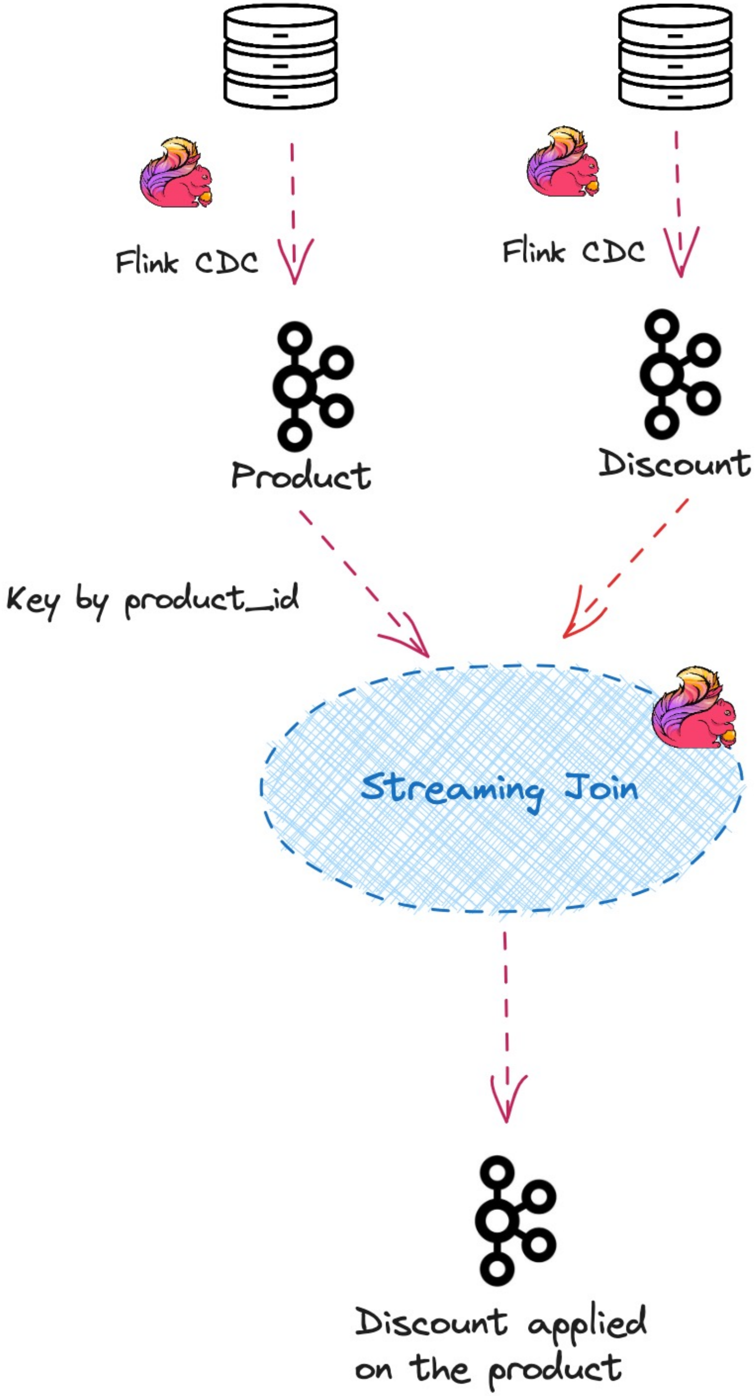
Sink Operator

Key	name	email
1	Alex	B
2	Sam	Z
3	David	X

Before Apache Paimon

Challenges with

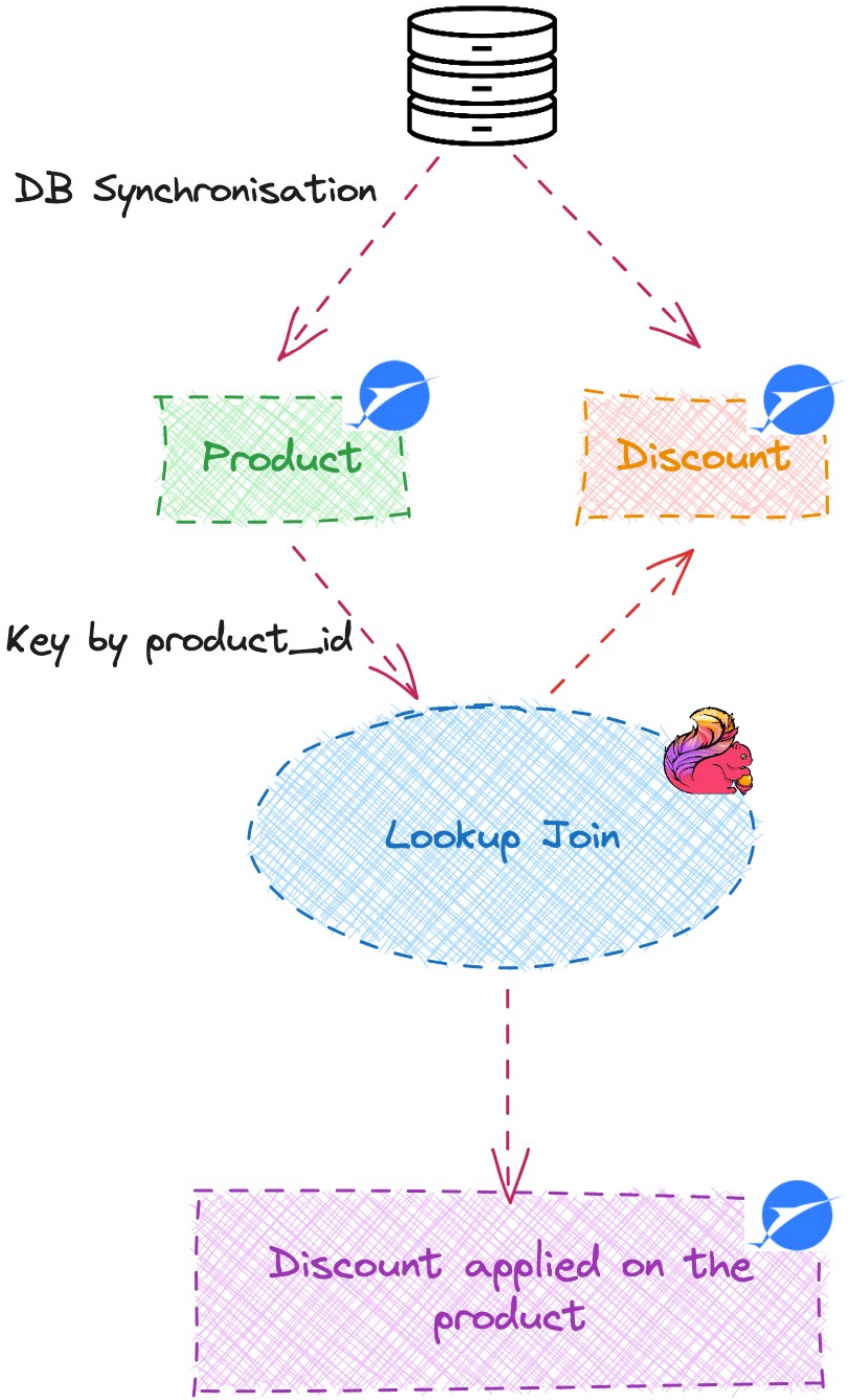
- Schema Evolution
- Kafka Storage scaling
- Kafka capacity during backfill
- Longer retention
- Large Flink state
- Higher Cost



KAFKA is a registered trademark of The Apache Software Foundation and has been licensed for use by AWS. AWS has no affiliation with and is not endorsed by The Apache Software Foundation.

After Apache Paimon

- Full support for Schema Evolution
- Scalable storage with Amazon S3
- Infinite retention
- Backfill is much easier
- Reduced Flink state by leveraging lookup join with Paimon table
- Low cost



Conclusion

- Data lake solves storage challenges with latency trade-off
- Apache Kafka for low-latency, data lake for under a minute
- Paimon Actions for database to lake synchronization
- Use the power of data lake, instead of Flink State
- Iceberg compatibility eliminates tooling challenges
- Additionally Paimon supports
  - Additional rich Merge Engines – Partial Update with Aggregation, Aggregation, First Row
  - Kafka like behavior for append only table

Q&A

alialem@amazon.com

rakssubh@amazon.co.uk