DATA+AI SUMMIT 2022

CDC System

An efficient way to replicate Transactional data into Delta Lake

Dibyendu Karmakar
SDE 3, Swiggy

ORGANIZED BY databricks

Background

Demand for new efficient mechanisms to create a near real-time replica of a transactional database into an analytical database is growing.

Scalability

Traditional transactional database replicas are not suitable for analytical workloads (OLAP).

Cross database joins

Cross database joins are also not easy and typically span multiple transactional domain boundaries.

Side effects

OLAP queries can interfere with OLTP workloads which can have an impact on the transactional flow.

Motivation

Requirements for transactional data is growing organically to serve multiple use cases.

- Analytical uses cases required transactional data in near real-time.
- ML models require near real-time data for feature generation
- Business reports, dashboards, alerting use cases requires these data with low latency.

Challenges

Challenges of building this kind of systems

Freshness

To operate in real-time, there is a need for providing data consumers with the latest data. Typically there is a 12/24 hours latency, which is high for most use cases.

Performance

Storing large numbers of small files affects the read latency.

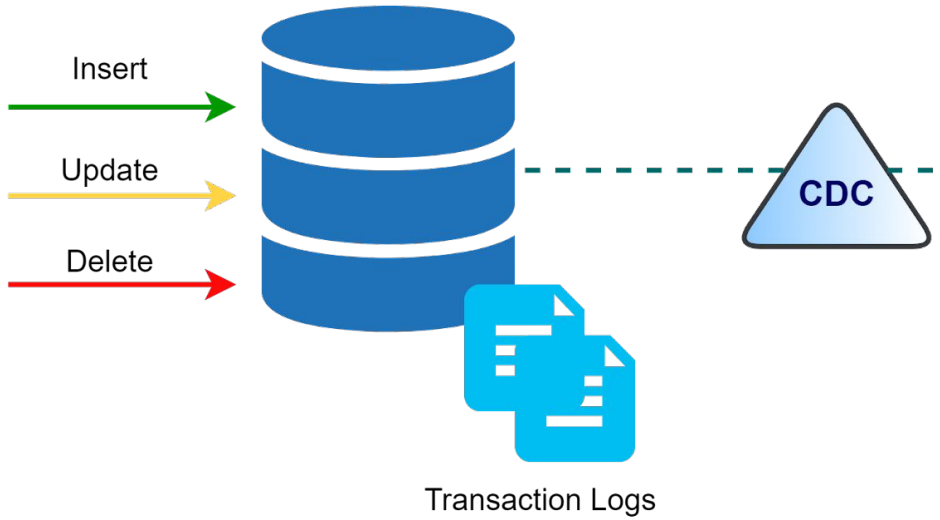
Consistency

For incremental snapshot based ingestion systems there is a need for a solution which supports update and delete operations for existing data.

CDC

Change Data Capture

CDC is a design pattern that captures all the changes happening on the data instead of dealing with the entire dataset.



CDC

Id	Name	Amount	Transaction
~~10002~~	~~Bobby~~	~~$500~~	Delete
10003	John	$1700	Update
10004	Ross	$1550	Insert

Motivating use-case

Late arriving updates and deletes

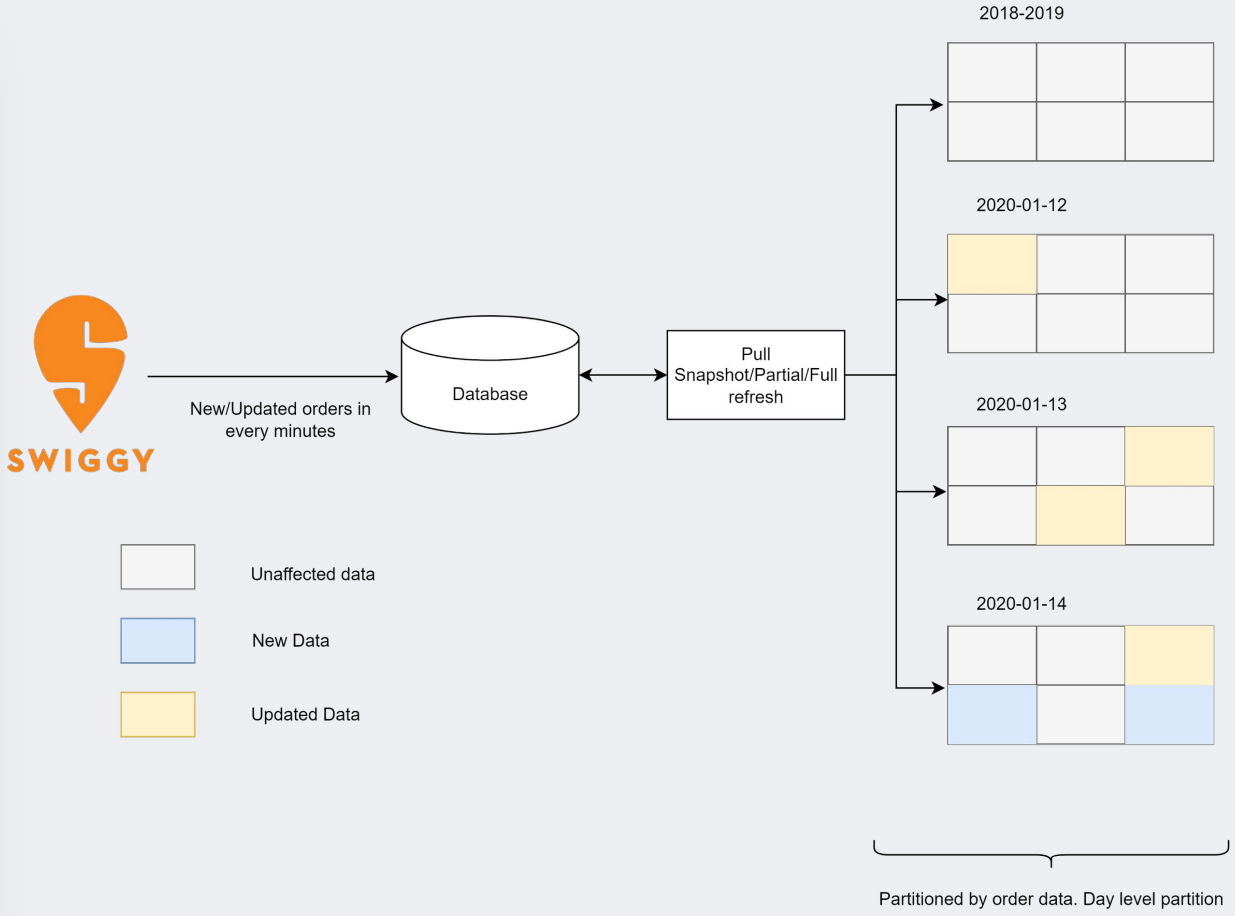
An order lifecycle in swiggy contains the following phases.

Pre-order system (Storefront) → Fulfillments → Delivery systems → Post-delivery systems

- ➔ Majority of order updates are done within a span of an hour.
- ➔ Some of these updates (e.g. customer service related updates, refund reconciliation etc) may come for the orders created in the last 15 days.

Challenges —

- Finding the correct partition and the correct file where data is present, requires a lot of data scan.
- Ensuring data consistency while updating the file is difficult.



Approach

Concept

Order matters

The principal idea of CDC based approach is to apply the changes on the analytical database in the ***same order*** as the transactional database does.

Type aware

Continuous and incrementally apply the changes based on the transaction type (insert / update / delete).

Approach

Two major steps

Initial load

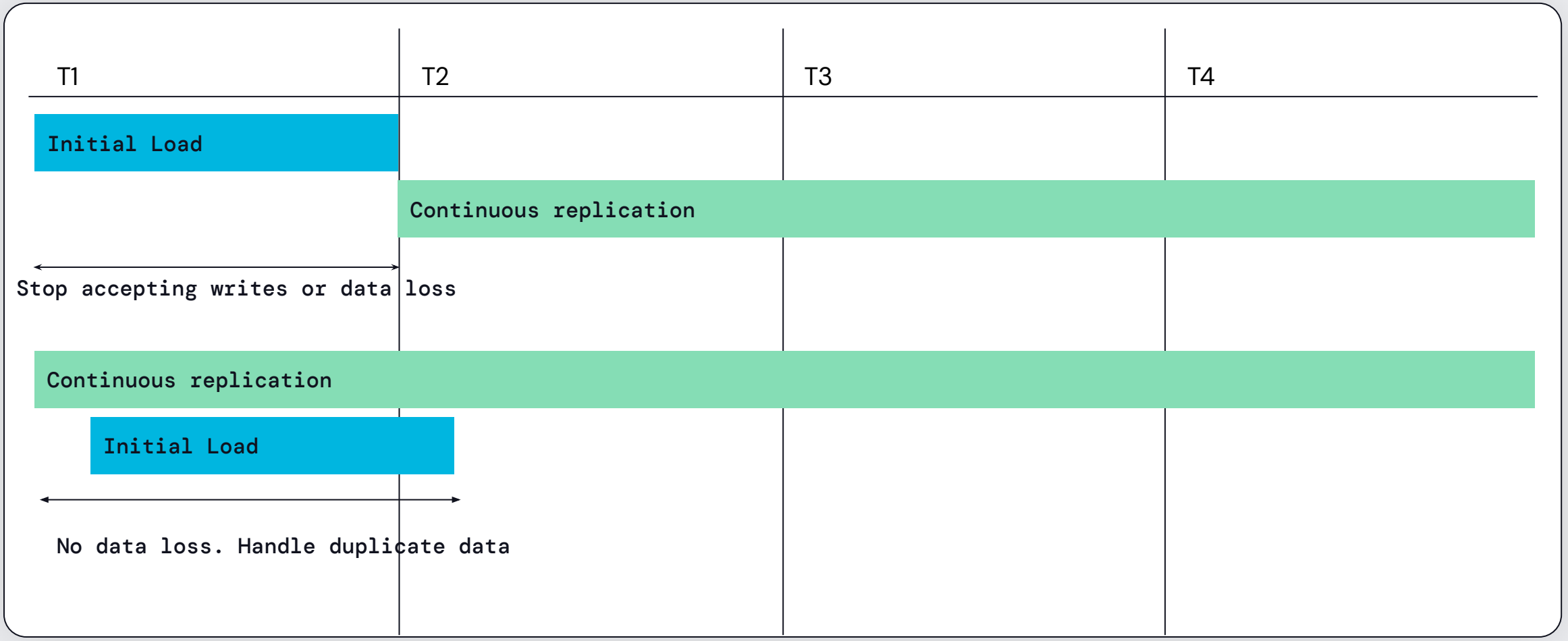
Create a copy of the current state of the transactional database.

Continuous replication

Apply current changes incrementally on initial copy in the same order as they appear in the transactional database.

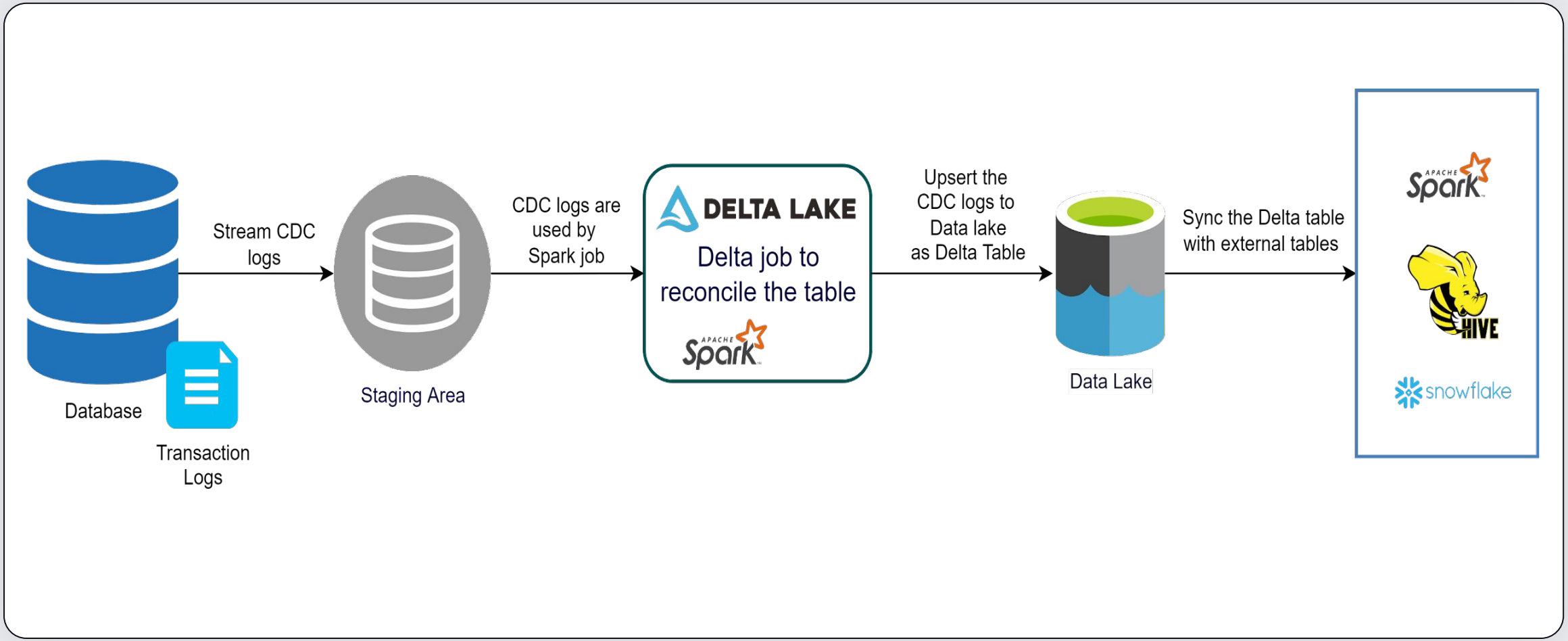
Approach

Key consideration is the timeline



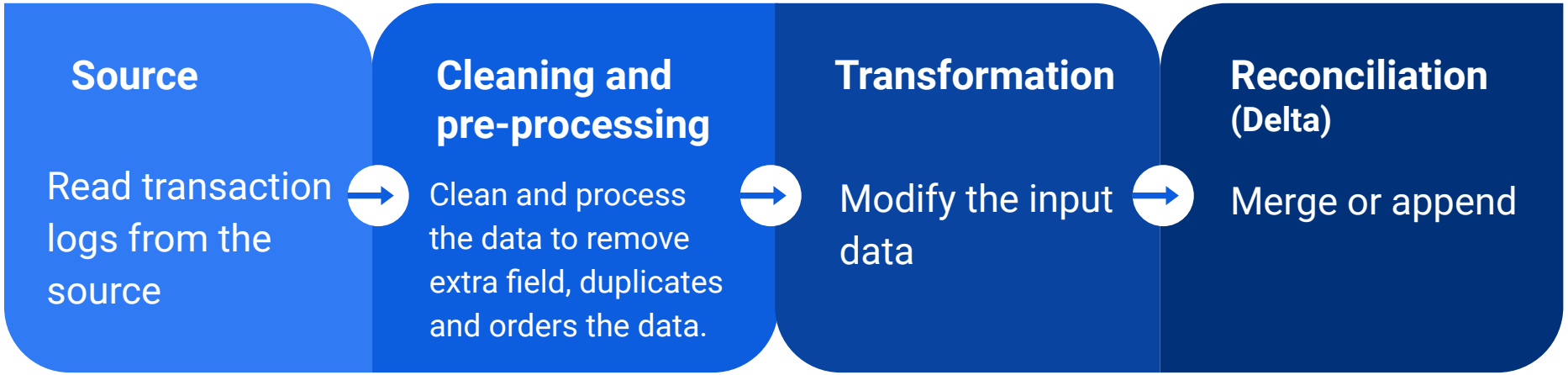
High Level Design

High level system components in CDC systems.



Data processing

Data flow from source to destination



Databricks and Delta lake

Delta lake is our primary storage format layer

Autoloader

Helped us to reduce the execution time from 8 hrs to 30 mins.

Delta merge

Powers the reconciliation process.

Schema evolution

Out-of-the-box schema evolution for backward compatible changes.

Optimize

Optimized the read jobs. Execution time reduced to 15 mins from 2 hrs.

Z-Order

Improved the read operations.

Vacuum

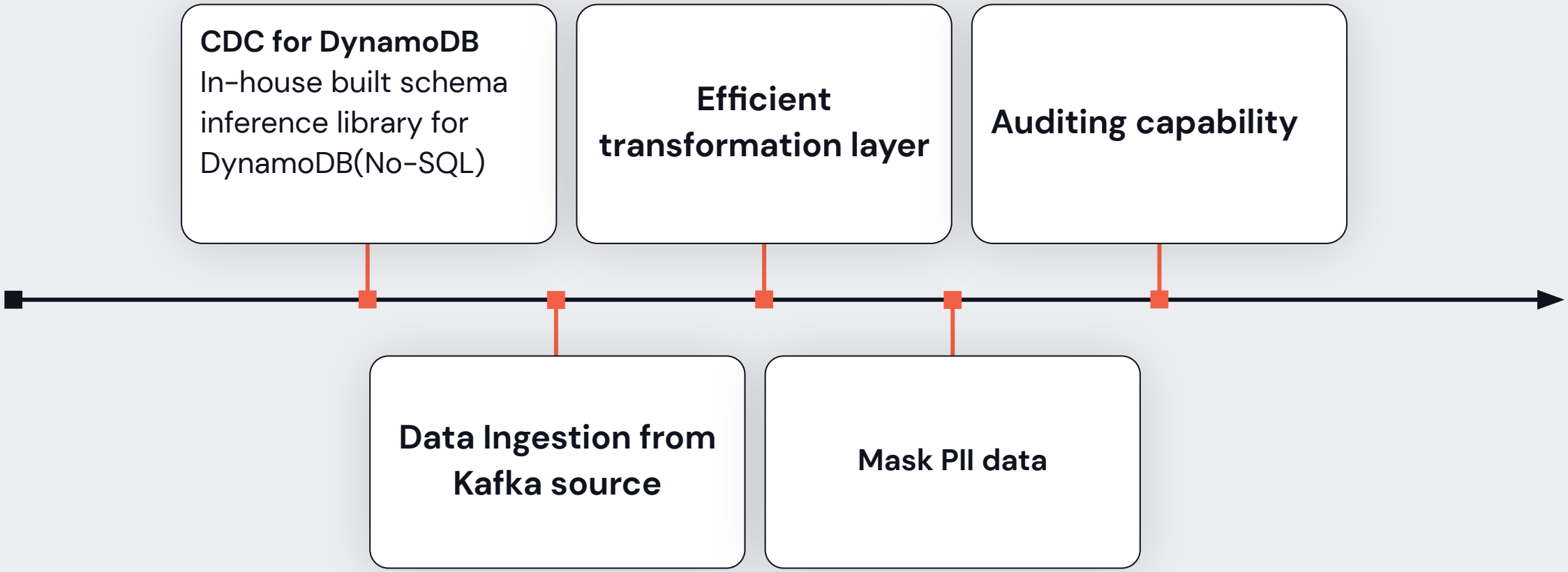
Clears the old commits and helps us to optimize storage cost.

Time travel

Reproduce the reports, Audit data changes.

Evolution of CDC system

Generalizing the ingestion system



Benefits of CDC system

Helps to obtain greater value from the data by allowing us to ingest and analyze data faster and use fewer system resources in the process.

Distributed load

It distributes the load round the clock. The overhead of bulk extract and load is broken down into small sets of incremental changes which makes the ingestion easy and efficient.

Scalable

The system is scalable to support a number of sources and it's very easy to scale at different points of the system like CDC replicator, Storage, Spark jobs etc.

Fast and efficient upsert

As it deals with incremental changes, time to upsert is very low as compared to bulk insert/overwrite.

Latency

It provides the freedom to the user to choose the latency requirements. Based on the requirement it decides the polling interval/ refresh frequency.

Benefits of CDC system

Helps to obtain greater value from the data by allowing us to ingest and analyze data faster and use fewer system resources in the process.

Consistency

One of the key tenets of this system is consistency. System is capable of handling data duplication, data loss etc.

Cost

As it deals with incremental data, overall load became very less and with that requirements of system resources also decreases.

Easy Recovery

The failure recovery process is very easy and simple. In case of failures, the system starts from the last checkpoint and duplicate events are handled at the spark layer.

DATA+AI
SUMMIT 2022

Thank you

Dibyendu Karmakar
SDE 3