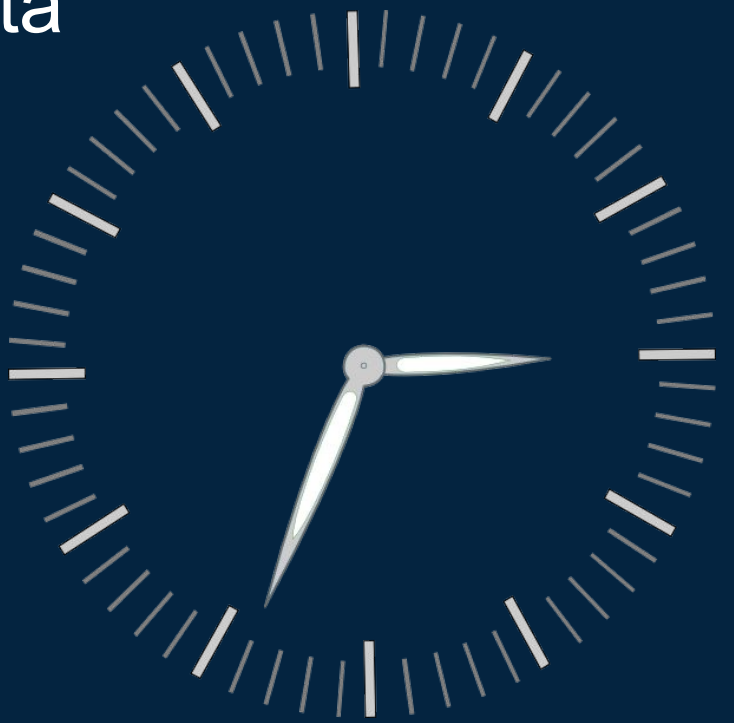
Working with real-time data

Learning Objectives

- Define real-time data
- Become familiar with streaming data processing
- Understand how to process files incrementally

Working with real-time data

- Data that grow over time indefinitely
- New records loaded to SQL database tables monitored with a CDC feed
- New CSV files being loaded to a data lake (cloud object storage)
- IoT messages and events stored continuously as raw JSON files

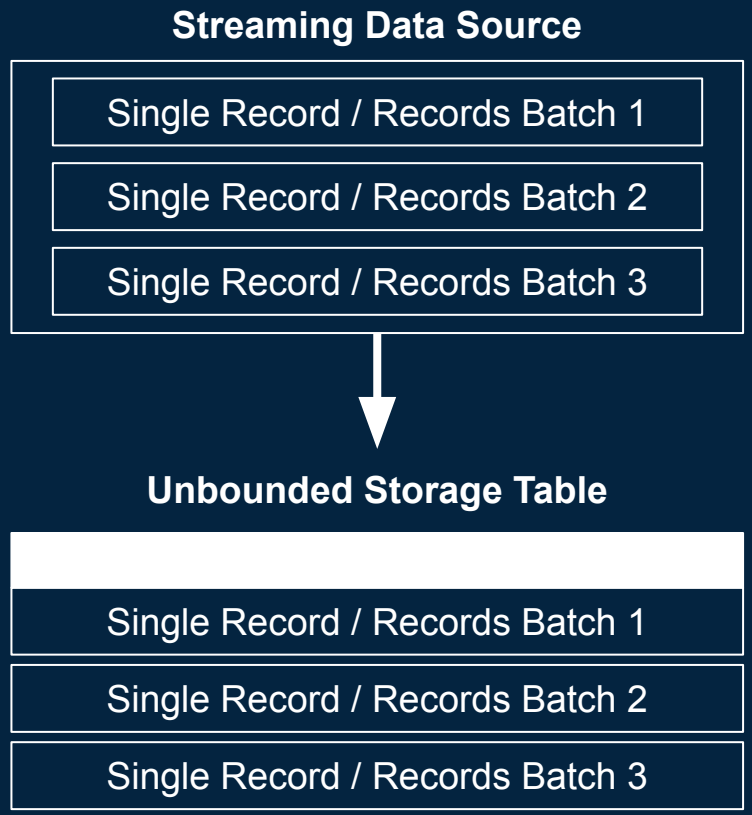


Structured Streaming

- Databricks Structured Streaming is built on Spark Streaming
- A streaming processing engine to query real-time data
- Persists query results in a storage object
- A storage object can be represented by files or tables

Structured Streaming

- Treats the storage object as a table
- The new data over time come in as new records
- That table can be queried as a normal static table
- That table is unbounded, new records might come in forever



Trigger Intervals

Trigger	Syntax	Output
Unspecified	default to processingTime = “500ms”	Processes data in micro batches at every specified interval
Fixed Interval	processingTime = “ 30 minutes”	Processes data in micro batches at every specified interval
Triggered batch	once = True	Processes all data available as a single batch and stops
Triggered micro-batches	availableNow = True	Processes all data available as multiple micro batches and stops

Output Modes

Mode	Syntax	Output
Append	“append”	Only new records are appended incrementally to the target table with every new batch
Complete	“complete”	The target table is completely overwritten every time with every new batch

Checkpointing and Guarantees

Checkpointing

- Saves the current stream status and stores it
- A checkpoint is unique to a streaming processing
- Records the range of data being processed
- The recording happens every time a streaming processing is triggered

Guarantees

- Streaming processing is fault tolerant
- If it fails, it can start from the last recorded checkpoint
- The target table will not append the same records
- We call this property idempotent

Unsupported Operations

- Distinct aggregations
- Sorting
- Deduplication

Auto Loader and Copy Into

In Databricks we have two main methods to ingest files incrementally:

- Auto Loader
- Copy Into

The idea is loading only new files, from a storage location, to a target storage location

Auto Loader

- Relies on Spark Structured Streaming to process files from a storage location
- Files are always loaded incrementally
- Especially effective with a very large volume of files
- Like Spark Structured Streaming, provides checkpointing and fault tolerance

Copy Into

- Copy Into is a SparkSQL statement
- With Copy Into we can load files from a storage location straight into a Delta Table
- Once again, the files are loaded incrementally
- This means that files already loaded in the Delta table will be skipped

Auto Loader vs Copy Into

Auto Loader

- Effective to process very large volumes of files
- Comes with all Spark Structured Streaming benefits
- Best practice to ingest data from cloud object storage

Copy Into

- Effective to process more limited volumes of files