#### Handling Late Data with Watermarks (sensorReadings .withWatermark("eventTime", "10 minutes") .groupBy("sensorId", window("eventTime", "10 minutes", "5 minutes")) .count()) Ø--O Data as (eventTime, sensorld) 12:20 12:21, id2 Intermediate Data late but within watermark state for 12:00 Data too late outside watermark 12:17, id3 **Event Time** 12:10 dropped 12:15 -- Max event Time seen till now as watermark 12:15, id1 🔵 Watermark = max eventTime - watermark delay 12:14, id2 12:13, id3 12:10 Data too 0---12:08, id2 12:07, id1 12:09, id3 late, ignored 12:08, id2 in counts Watermark 12:05 updated every trigger using delay = 10 min wm=12:14-10m=12:04 12:04, id1 12:00 12:05 12:25 12:10 12:15 12:20 **Processing Time** 12:00 - 12:10 id1 1 Table not with 5 min triggers updated with 12:00 - 12:10 id2 1 12:00 - 12:10 id2 2 2:00 - 12:10 id2 too late data 12:05 - 12:15 id1 1 12:00 - 12:10 id3 1 (12:04, id1) 12:05 - 12:15 id1 2 12:05 - 12:15 id2 1 12:05 - 12:15 id2 2 12:05 - 12:15 id2 3 Result Tables after each trigger 12:05 - 12:15 id3 2 12:05 -12:15 id3 1 12:10 -12:20 id2 1 Table updated with late data 12:10 - 12:20 id1 1 dark rows (12:17, id3) 12:10 - 12:20 id3 1 12:10 - 12:20 id3 2 are updated Watermarking in counts Windowed Grouped Counts 45

#### **Data lakes**

- decouples the distributed storage system from the distributed compute system
  - Allows each system to scale out as needed by the workloads
- Organizations build their data lakes by independently choosing
  - Storage system: HDFS, S3, Cloud and etc.
  - File format:
    - Structured: Parquet, ORC
    - semi-structured: JSON
    - unstructured formats: text, images, audio, video
  - Computing / Processing engine(s):
    - batch processing engine: Spark, Presto, Apache Hive
    - stream processing engine: Spark, Apache Flink
    - machine learning library: Spark MLlib, scikit-learn, R

#### **Data Lakes**

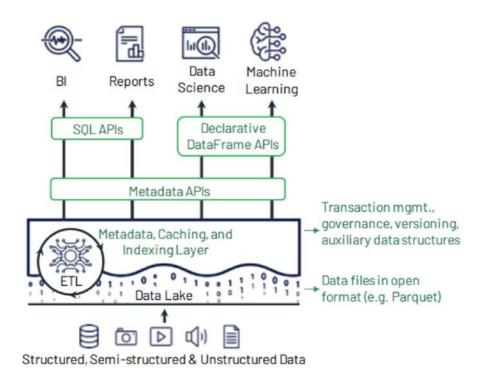
#### Pros

- Flexibility on choosing storage, data format and processing engines
- A much cheaper solution than databases → explosive growth of the big data ecosystem

#### Cons:

- Fail to provide ACID guarantees
- Building and maintaining an effective data lake requires expert skills
- Easy to ingest data but very expensive to transform data to deliver business values
- Data quality issues due to the lack of schema enforcement

## **Data Lakehouse implementation**



#### **Delta Lake**

- the metadata, caching and indexing layer on top of a data lake storage that provides an abstraction level to serve ACID transactions and other management features
  - Transactional ACID guarantees
  - Full DML (Data Manipulation Language) support
  - Audit History
  - Unification of batch and streaming into one processing model
  - Schema enforcement and evolution
  - Rich metadata support and scaling

### **Delta Lake Format**

- a standard Parquet file with additional metadata
- Parquet Files
  - Column oriented: perform compression on a column-by-column basis
  - Open source
  - Self-describing: actual data + metadata (schema & file structure)



14

# The Delta Lake Transaction Log (DeltaLog)

- The transaction log is an ordered record of every transaction made against a Delta table since it was created.
- It acts as a single source of truth and tracks all changes made to the table.
- The main goal is to enable multiple readers and writers to operate on a given version of a dataset simultaneously.
- It is at the core of many important features
  - ACID transactions
    - Spark looks at the transaction log to get the latest version of the table
    - If an operation is not recorded in the transaction log, it never happened.
  - Scalable metadata handling
  - Time travel

## Five Steps to Define a Streaming Query

- Step I: Define input sources
- Step 2:Transform data
- Step 3: Define output sink and output mode
  - Output writing details (where and how to write the output)
  - Processing details (how to process data and how to recover from failures)
- Step 4: Specify processing details
  - Triggering details: when to trigger the discovery and processing of newly available streaming data.
  - Checkpoint Location: store the streaming query process info for failure recovery
- Step 5: Start the query