DS-3002: Data Systems

Data Lakehouses & Streaming Data

Prof. Jon Tupitza – Spring 2022

UNIVERSITY of VIRGINIA

Modern Data Platform: Solution Scenarios

Big (Unstructured and/or Poly-Schematic) Data Integration and Advanced Analytics

"We want to integrate all our data into our data warehouse"

Modern Data Warehousing

"We're trying to predict which of our customers will churn"

Advanced Analytics

"We're trying to get insights from our devices in real-time"

Real-Time Analytics

Delta Lake: A New Standard for Data Lakes

DELTA LAKE

- Open Format Based on Parquet
- Optimized for Cloud Storage
- Built to Handle Scalable Metadata
- Transactional Support
- Apache Spark API's

Databricks: Data Lakehouse Architecture

Addresses the Shortcomings and Dysfunctions of Data Lakes and Data Warehouses

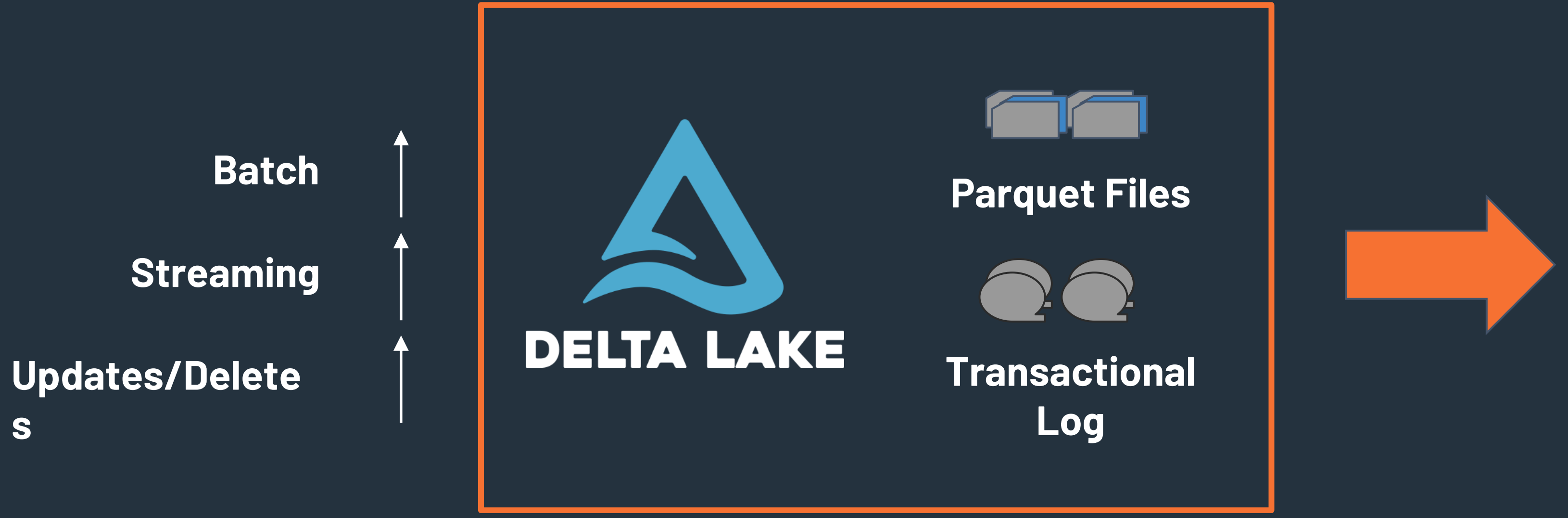
Data Lakes: Data Reliability Challenges

Failed production jobs leave data in corrupt state requiring tedious recovery

Lack of schema enforcement creates inconsistent and low quality data

Lack of consistency makes it almost impossible to mix appends and reads, batch and streaming

Delta Lake: Ensures Data Reliability

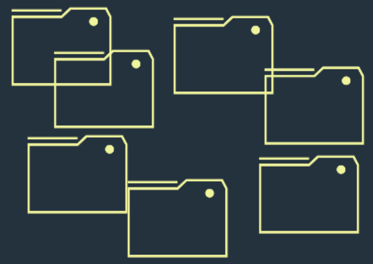


High Quality & Reliable Data
always ready for analytics

Key Features

- ACID Transactions
- Schema Enforcement
- Unified Batch & Streaming
- Time Travel/Data Snapshots

Data Lakes: Performance Challenges

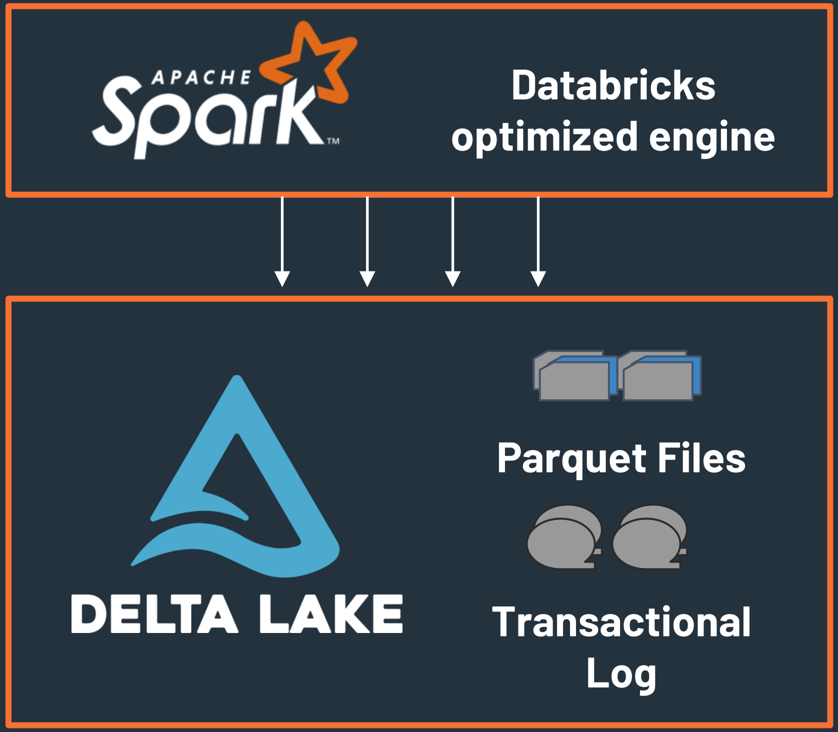


Too many small or very big files - more time opening & closing files rather than reading contents (worse with streaming).

Partitioning aka "poor man's indexing"- breaks down if you picked the wrong fields or when data has many dimensions, high cardinality columns.

No caching - cloud storage throughput is low (cloud object storage is 20-50MB/s/core vs 300MB/s/core for local SSDs).

Delta Lake: Optimizes Performance



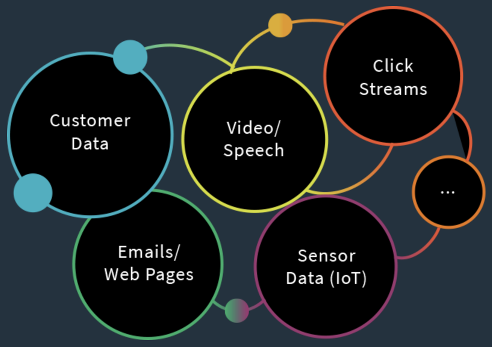
Highly Performant
queries at scale

Key Features

- **Indexing**
- **Compaction**
- **Data skipping**
- **Caching**

Delta Lakes: Makes Data Ready for Analytics

DELTA LAKE



Reliability

Performance

Data Science & ML

- Recommendation Engines
- Risk, Fraud Detection
- IoT & Predictive Maintenance
- Genomics & DNA Sequencing

Databricks: Why Use It?

Make Data Engineers More Effective

- Single, open, consistent way to do ETL and streaming
- BYO IDEs and tools
- Reliable APIs, CLIs, and other interfaces
- DevOps Integration
- Comprehensive library management

Make Data Scientists More Effective

- Collaborative Data Science Workspace
- ML Runtime with pre-installed, optimized libraries
- Open, Consistent tracking and deployment with MLFlow
- HyperOpt, AutoML

Make Compute More Effective

- 30%-500% faster than OSS Spark
- Always the latest, greatest Spark
- Query-load-based autoscaling
- Ephemeral jobs clusters
- Unparalleled Expertise

Data Integration

How to Approach Populating a Data Warehouse

UNIVERSITY of VIRGINIA

Data Processing: Batch vs. Stream Processing

Challenges and Components Encountered When Integrating Real-Time & Historic Data

- Data Motion:
  - At-Rest Data: Data that has settled
  - In-Motion Data: Data where new events arrive at some continuous interval
- Datasets:
  - Bounded Datasets: Data of a known & finite size; having a start point and endpoint
  - Unbounded Datasets: Data wherein events are continuously added to the dataset
- Data Processing Engines:
  - Batch Processing Engines: Only capable of processing data after it has settled
  - Streaming Processing Engines: Capable of processing data in-motion as it’s arriving

Data Processing Paradigms: Latency Requirements

Latency & Response:

The speed at which clients require new insights determines the frequency at which new data must be processed

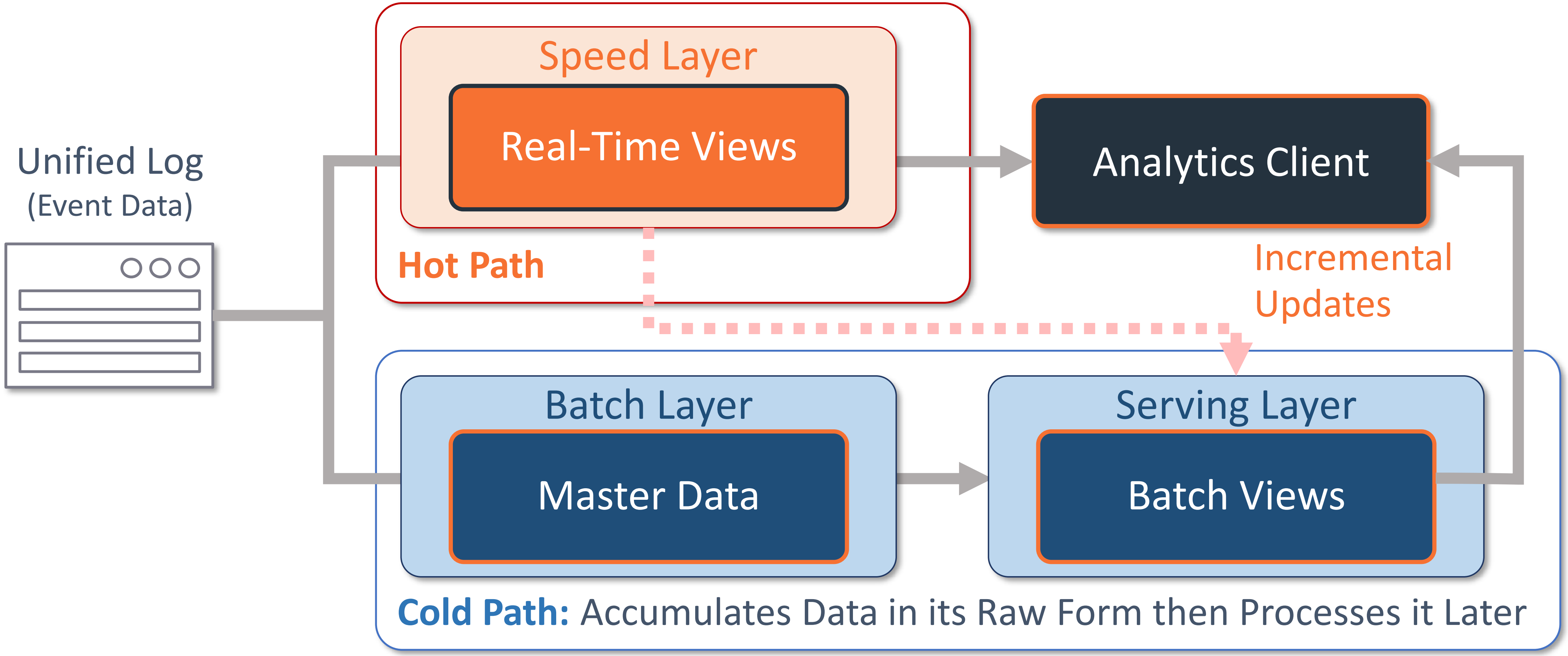
1. Batch
2. Continuous/Streaming
3. Real-time

10 ms → 100 ms → 1 sec → 1 min → 1 hour → 1 day

Low-Latency Real-Time	Real-Time	Micro-Batch	Batch
• Spark-less, highly-available prediction server	• Prediction server with Spark	• Structured Streaming	• Spark batch processing

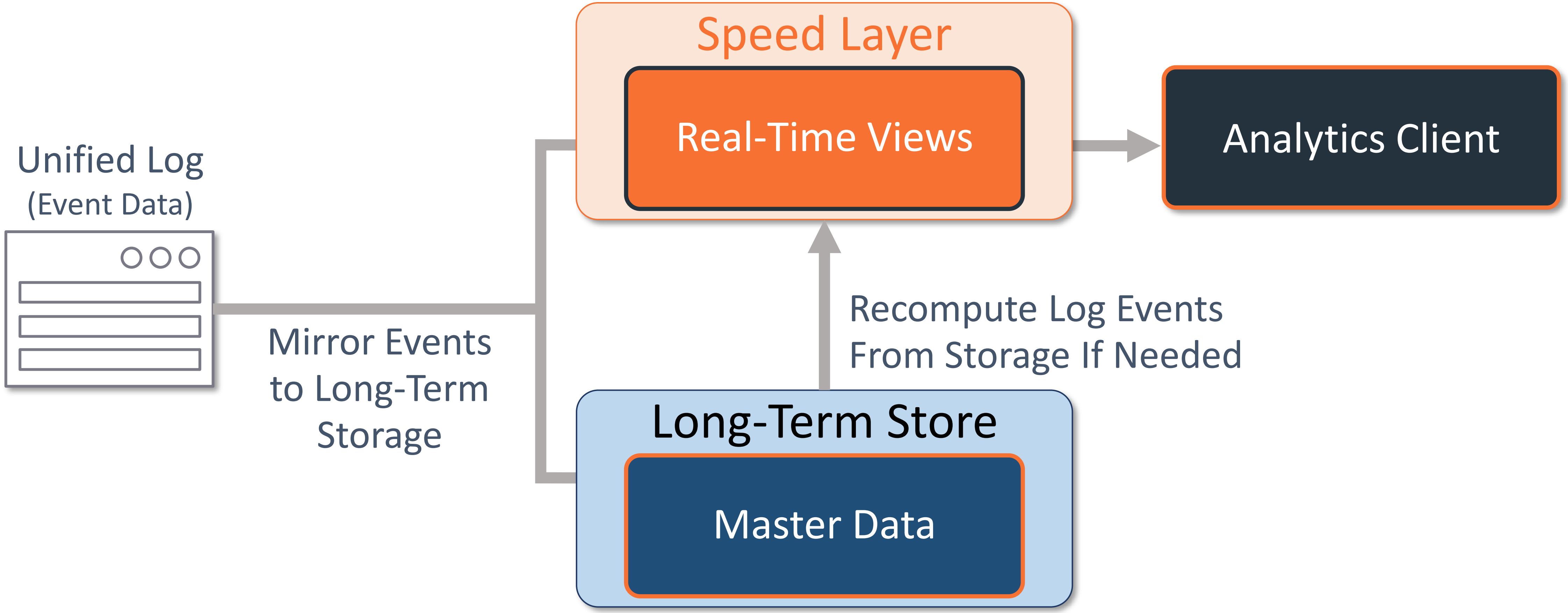
Data Processing Paradigms: Lambda Architecture

All Data Flows Through One of Two Paths: Hot or Cold



Data Processing Paradigms: Kappa Architecture

All Data Flows Through One of Two Paths: Hot or Cold



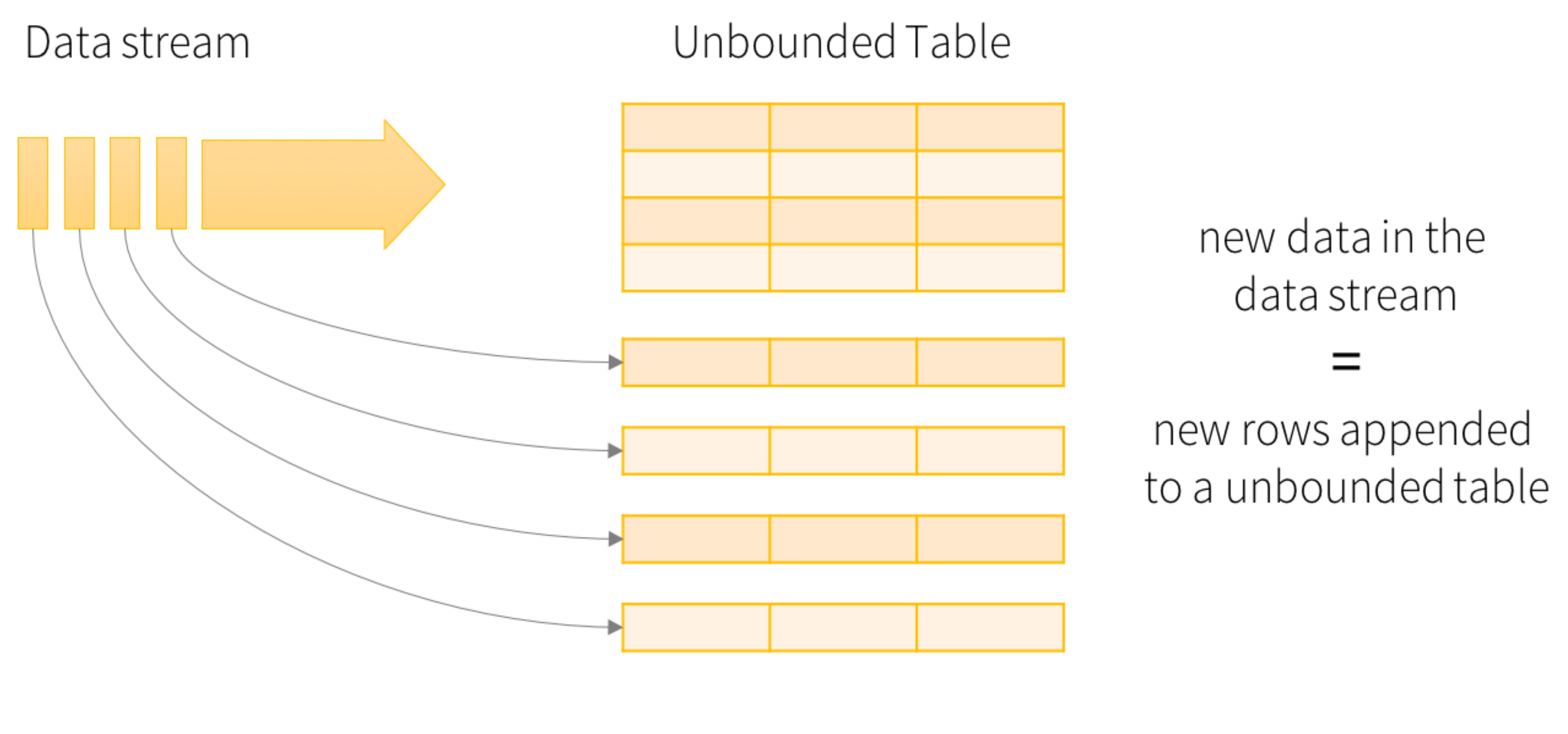
Databricks: Spark Structured Streaming

Implements Incremental Processing to Provide Near Real-Time Analytic Insights

- Designed for Distributed Computing:
  - Enables the incremental loading of new data files, each of which represent mini-batches or micro-batches, from streaming inputs.
  - Enables reasoning over potentially very large, continuously changing, sources.
- Prefix-integrity Guarantee:
  - Ensures Spark SQL can be used to query a table while a streaming query is continuously writing data transactionally such that concurrent interactive query processing always sees a consistent view of the latest data
- Joins with Static Data:
  - Structured Streaming data can be joined with static [reference] data

Databricks: Spark Structured Streaming

Enables Interaction with Unbounded Data Sources As If They Were SQL Database Tables



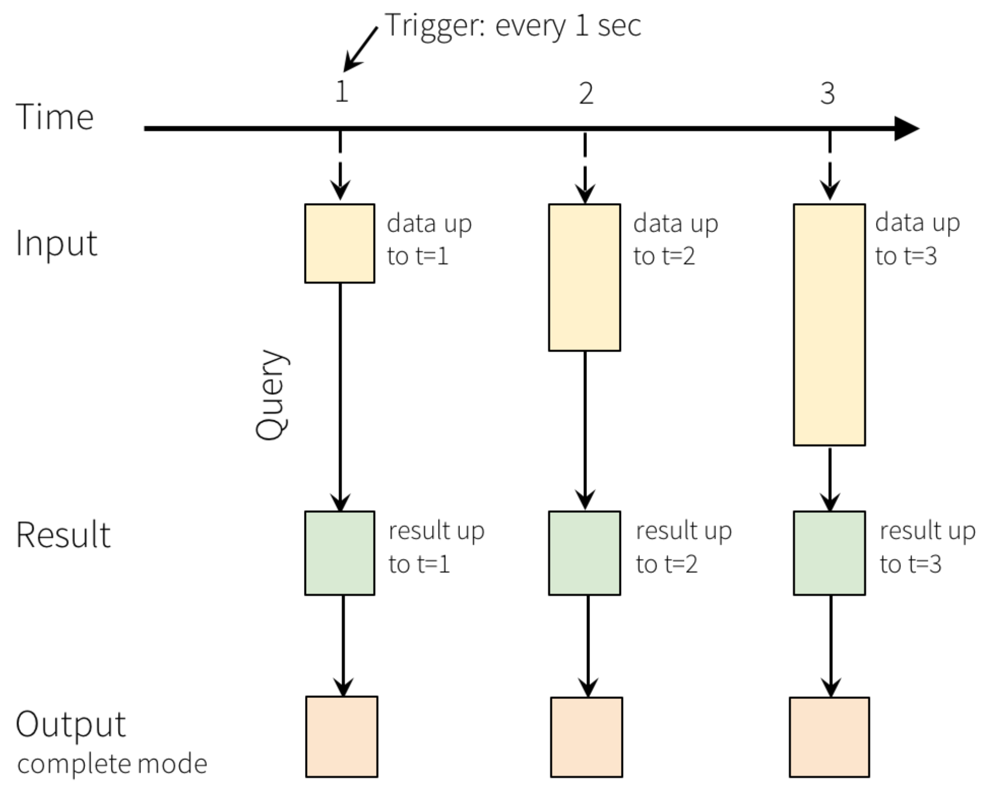
Data stream as an unbounded table

A **Data Stream** describes any data source that grows over time:

- JSON log files landing in cloud storage
- Database updates captured in a CDC feed
- Events queued in a messaging queue feed
- CSV files of sales closed the previous day

Databricks: Spark Structured Streaming

The Spark Structured Streaming Programming Model

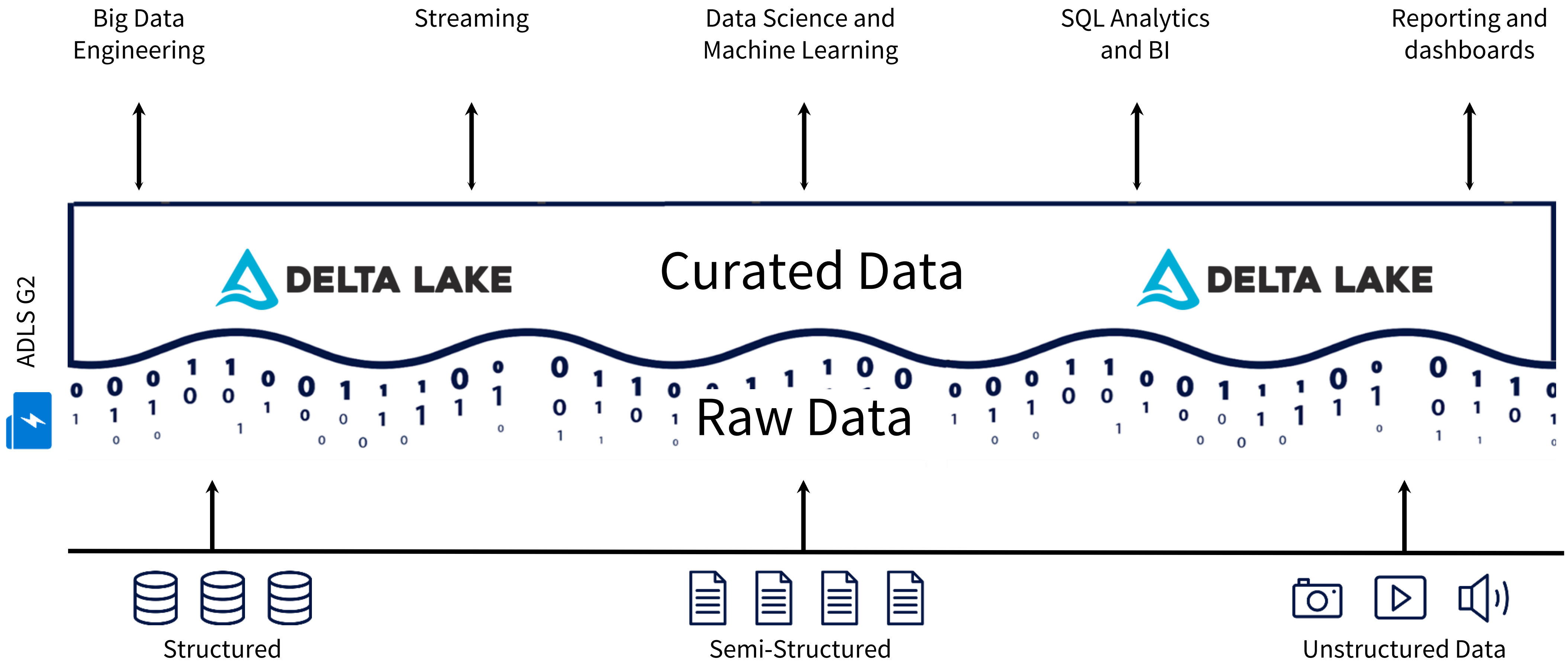


Programming Model for Structured Streaming

- The developer defines an **input table** by configuring a streaming read against a **source**. The syntax for doing this is similar to working with static data.
- A **query** is defined against the input table. Both the DataFrames API and Spark SQL can be used to easily define transformations and actions against the input table.
- This logical query on the input table generates the **results table**. The results table contains the incremental state information of the stream.
- The **output** of a streaming pipeline will persist updates to the results table by writing to an external **sink**. A sink will be a durable system such as files or a pub/sub messaging bus.
- New rows are appended to the input table for each **trigger interval**. These new rows are analogous to micro-batch transactions, and they will be automatically propagated through the results table to the sink.

Modern Data Lakehouse Architecture

Building a Modern Data Lakehouse using Best-in-Class Open-Standard Components



Paradigms: Data Storage and Retrieval

Schema on Write versus Schema on Read

Schema on Write: Stores data in an organized structure (schema)

Client or Process → Writes Data *to* → Schema

Client or Process ← Reads Data *from* ← Schema

RDBMS

Schema on Read: Applies schema only when read, data stored in its original format

Client or Process → Writes Data → *To* Storage

Client or Process ← Reads Data *into* Schema ← *From* Storage

Data Lake

Modern Data Platform: Data Services Pipeline

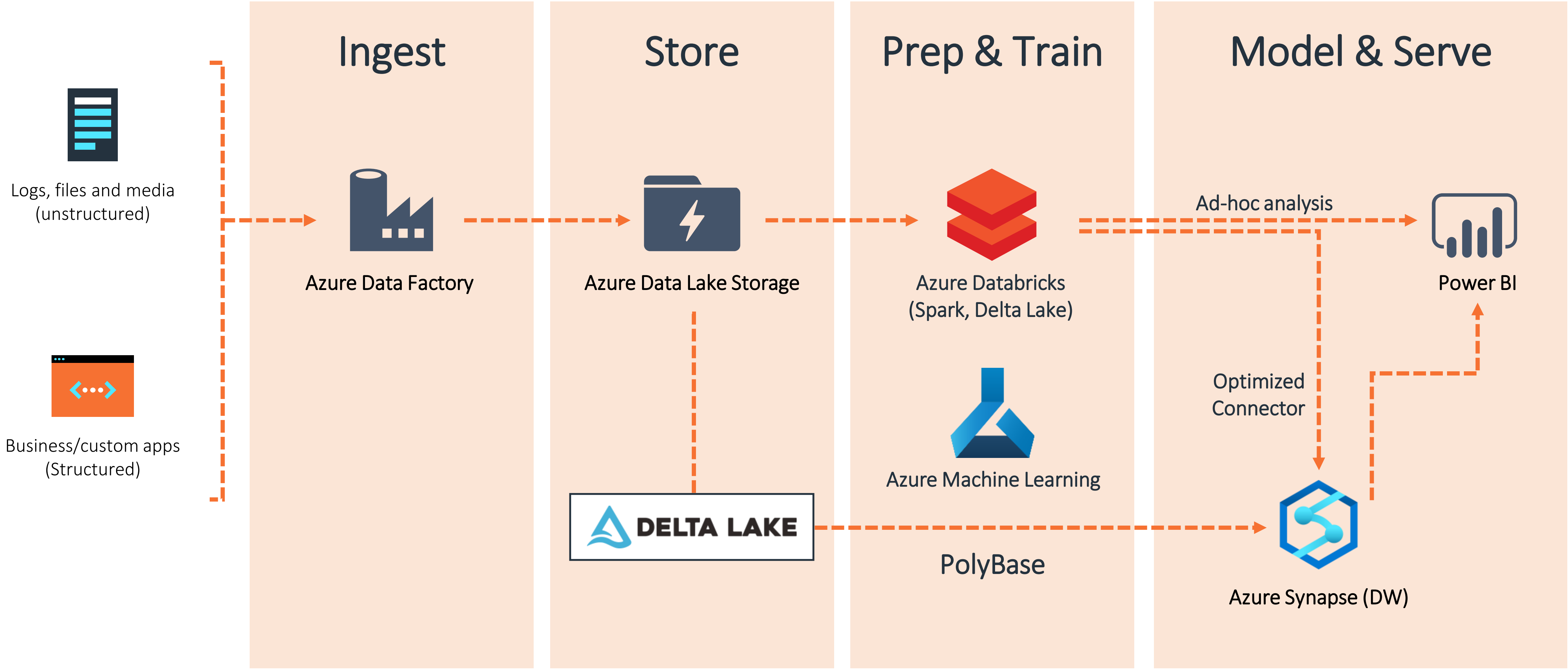
INGEST	STORE	PREPARE	SERVE	PREDICT
Data Orchestration & Monitoring	**Unstructured Data Storage**	**Data Cleansing, Transformation & Streaming**	**Data Modeling, Serving & Storing**	**Reporting, BI, Predictive Analytics & AI**
Data Integration	Data Lake Storage	Databricks	Data Warehouse	Machine Learning

DELIVER

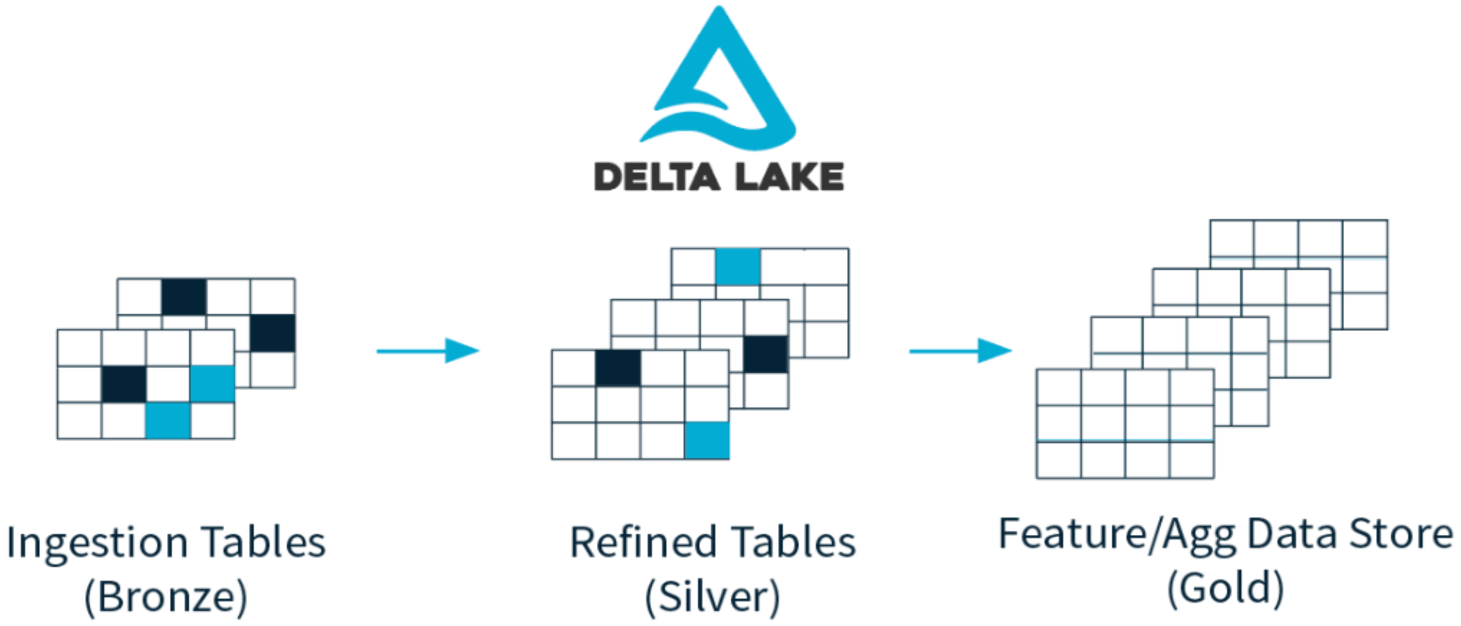
MLOps Integration: CI/CD Pipelines, Version Control, Monitoring, Test Automation, Infrastructure as Code, Containers, Microservices

DevOps

Data Engineering… for Data Science



Databricks: Delta Lake at Scale



ACID Transaction Guarantees

Atomic, Consistent, Isolated, Durable

Versioned Parquet Files

Delta transaction log keeps track of all operations

Efficient Upserts

MERGE, DELETE, UPDATE

Time Travel

Audit history, pipeline debugging, data reproducibility

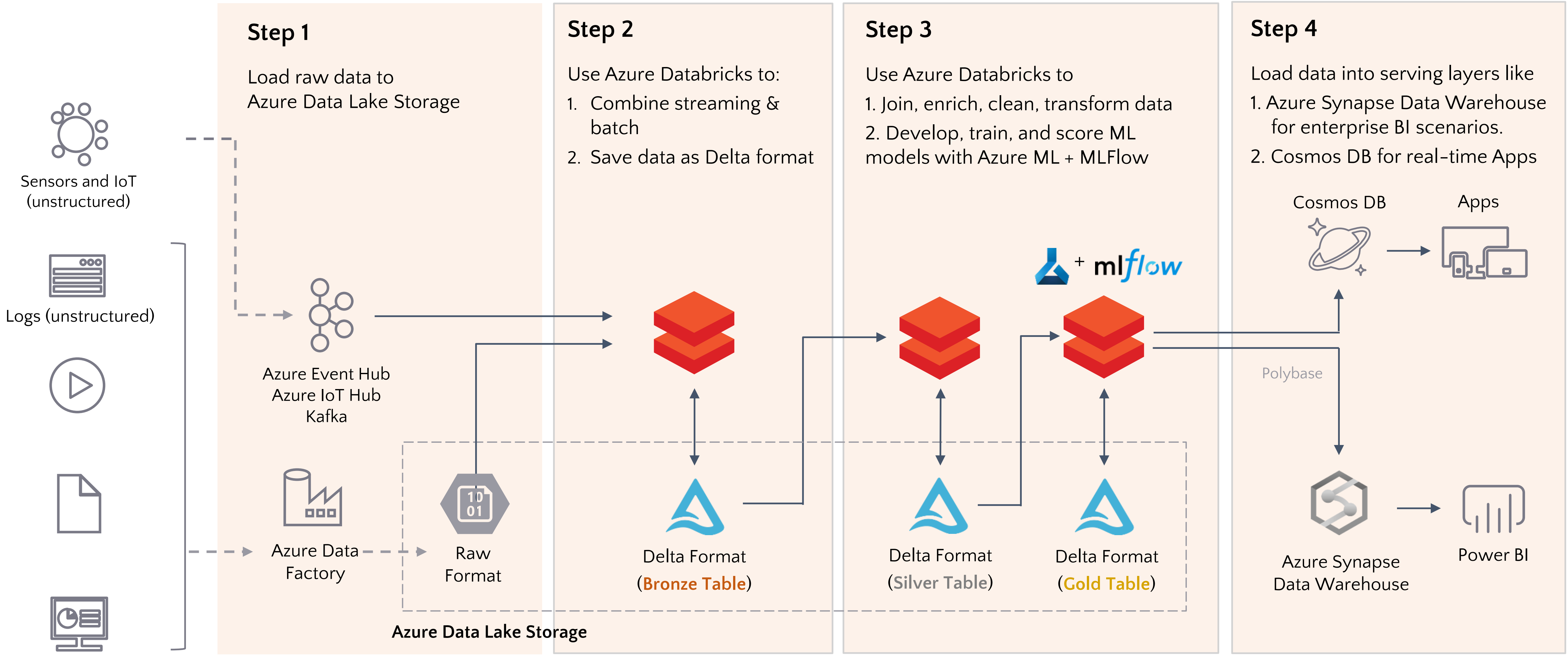
Small file compaction with no interrupt to availability

OPTIMIZE and VACUUM

Z-Order partitioning with up to 100x perf

New multidimensional partitioning enables data skipping

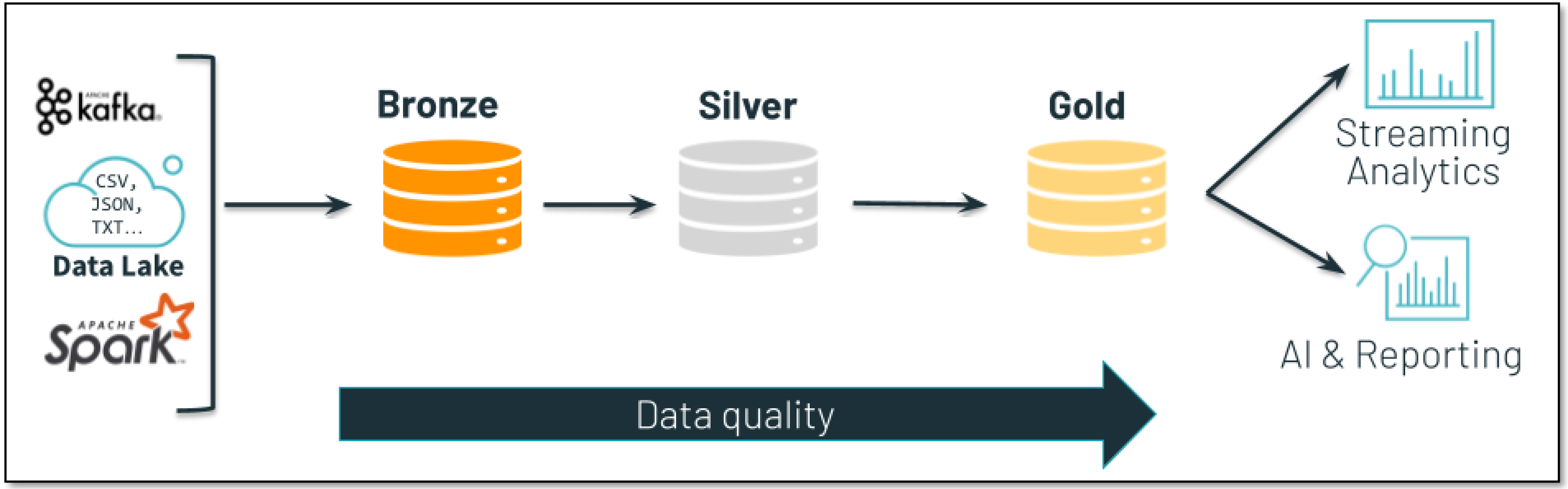
Design Pattern: Modern Data Warehousing



Data Lakehouse: Multi-Stage Architecture

Bronze Layer: Replaces the "Data Swamp" with Some Organization

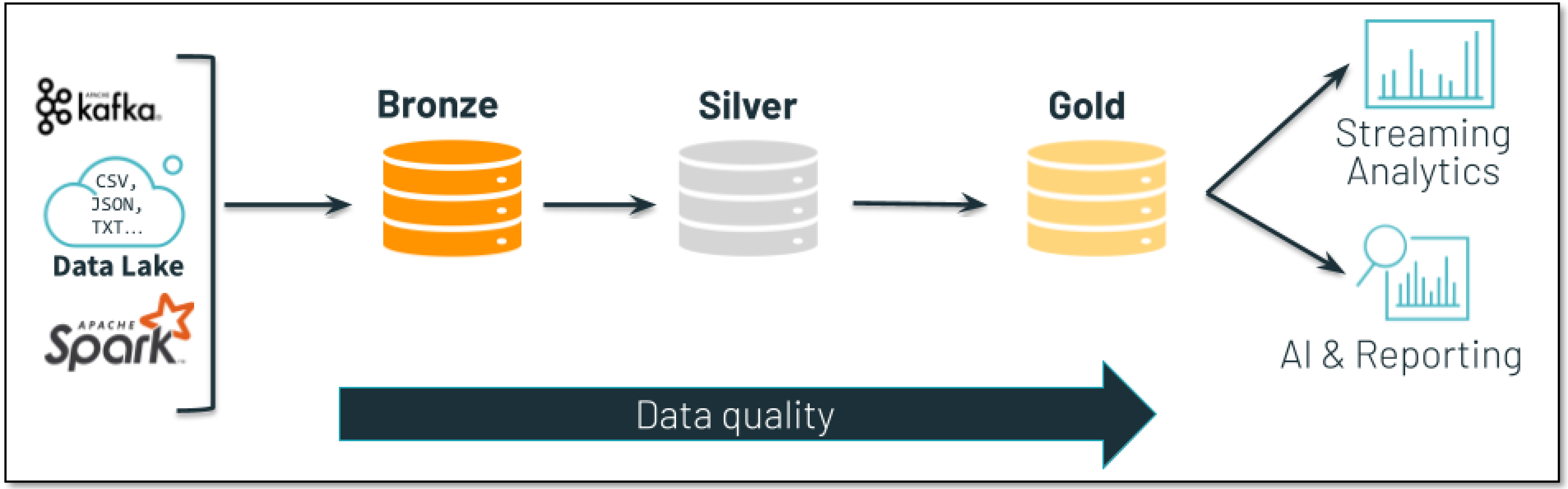
- Typically just a copy of the "raw" data that's being ingested
- Replaces the traditional Data Lake
- Provides efficient storage and querying of full, unprocessed history of data



Data Lakehouse: Multi-Stage Architecture

Silver Layer: Validated Single Source of Truth Data

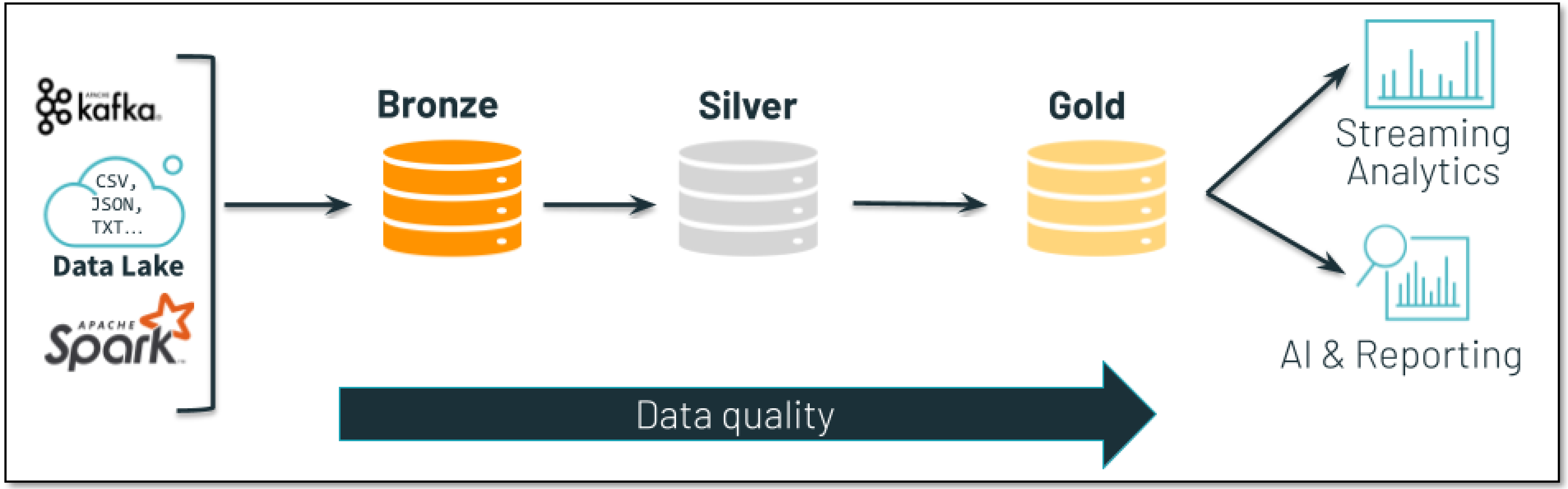
- Reduces data storage complexity, latency and redundancy
- Optimizes ETL throughput and analytic query performance
- Preserves the grain of original data while enforcing the Production schema
- Eliminates duplicate records, checks data quality & quarantines corrupt data



Data Lakehouse: Multi-Stage Architecture

Gold Layer: Customer-Ready Insights, Not Just Data

- Powers applications, reporting, dashboards, and ad hoc analytics
- Refined views of the data, typically including aggregations
- Reduces strain on production systems
- Optimizes query performance for business-critical data



Q & A

A Survey of Data Management Systems