O'REILLY®

Data Lakehouse Fundamentals

Prerequisites

- Some familiarity with working on the cloud
- Comfortable writing SQL queries to analyze data
- Comfortable working with Python to analyze data

Set up for demos

- Please sign up for a an Azure account
  https://portal.azure.com

- Google Drive link for resources
  https://drive.google.com/drive/folders/1gE10t1c-6ZOvK5ZeH8UxhsZcYokcUlI7?usp=sharing

General Poll

How comfortable are you with SQL?

- Never written queries in SQL before
- Somewhat comfortable writing SQL queries
- Very comfortable writing SQL queries

General Poll

How comfortable are you with Python?

- Never written Python code before
- Somewhat comfortable writing coding in Python
- Very comfortable coding in Python

General Poll

Have you worked with data lakes or data warehouses?

- Worked with neither data lakes nor data warehouses
- Worked with data warehouses not data lakes
- Worked with data lakes not data warehouses
- Worked with both data lakes and data warehouses

O'REILLY®

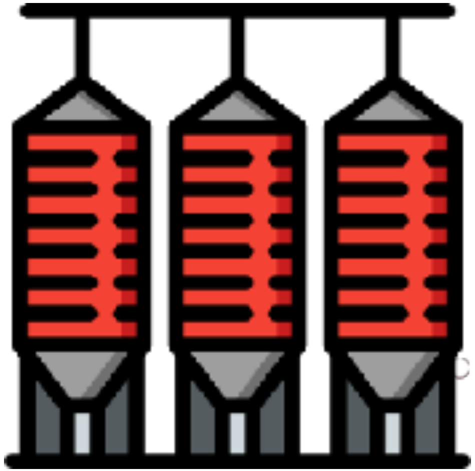
Data Silos

Data Silo

An isolated store of enterprise data, unconnected to other data repositories, and unavailable to most users in the organization

Data silos pose many serious problems

- No single source of truth
- Data is usually replicated
- Consistency across silos is difficult
- Who owns which data?
- Storage costs can be significant
- May not fulfill audit requirements



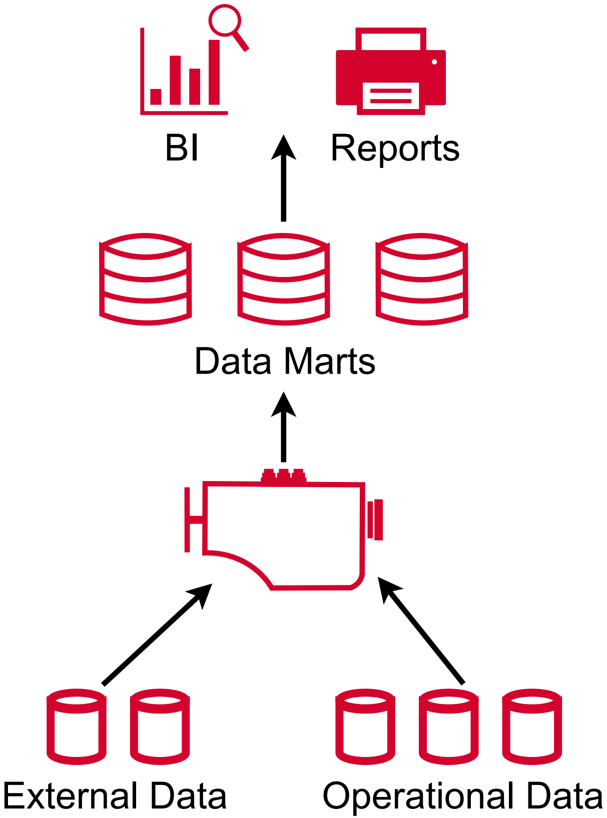
Over time, the issues with data silos outweigh the benefits - a better solution is needed

Data Warehouses and Data Lakes

Data Warehouse

A repository of structured data meant for data analytics. Data is gathered from several disparate sources using ETL pipelines.

Data Warehouse



A Traditional Data Warehouse

- A system to store and manage large volumes of data
- An organization’s single source of truth
- Data typically collected from multiple, disparate sources
- Data is structured (adheres to a schema)
- Meant to support business intelligence tasks
  - Data analysis
  - BI reports
  - ETL tasks

Limitations of a Data Warehouse

- Expensive to store data
- Vendor lock-in - data often in a proprietary format
- Cannot work with unstructured data
  - Text
  - Media Images/Audio/Video
- Not applicable in several domains
  - Data science and machine learning
  - Real-time streaming

Most limitations of a warehouse are addressed by a Data Lake

Data Lake

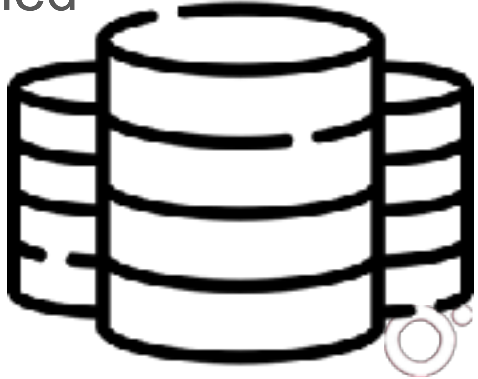
A single, low-cost repository for all enterprise data which is stored in a raw form until it is needed.

Data Lake

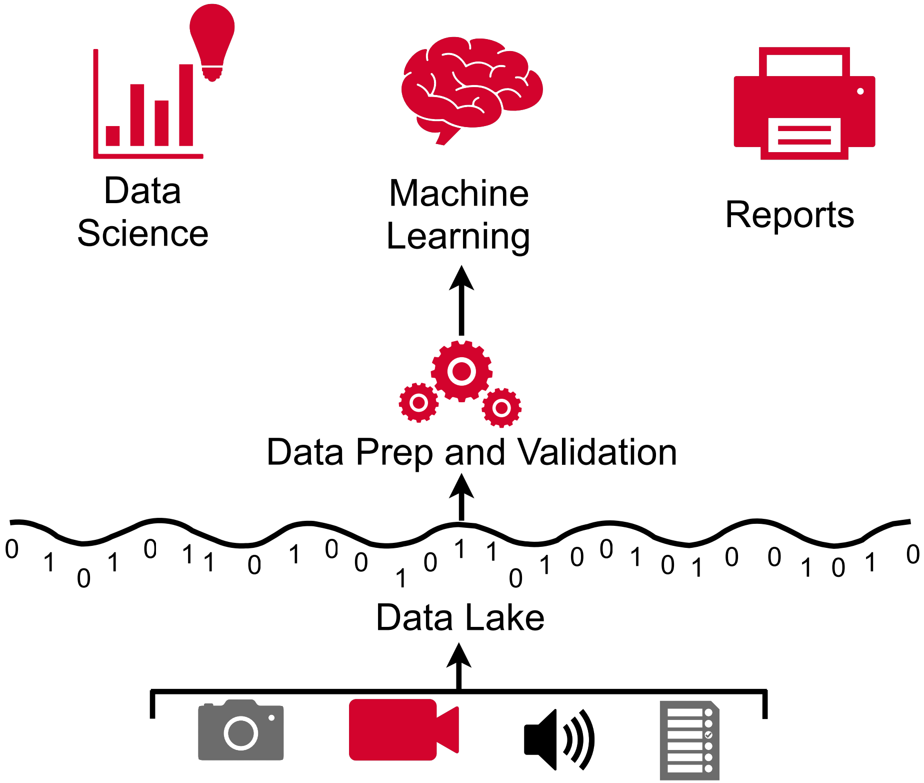
A single, low-cost repository for all enterprise data which is stored in a raw form until it is needed.

Data Lakes

- Can store all your data no matter what form
- Support structured, semi-structured, and unstructured data
- Repositories for data in several forms
  - Batch and streaming
  - Cloud-hosted and on-premises
- Can store data whose use case is yet to be defined
- E.g. Azure Data Lake Storage, AWS S3



Data Lake



Data Science

Machine Learning

Reports

Data Prep and Validation

Data Lake

Structured, Semi-structured and Unstructured Data

Benefits of Data Lakes

- Inexpensive
- Flexible
- Fulfill multiple use cases
    - Machine learning
    - Data analysis
    - Archival storage
- May serve as a staging area for data
    - Until use cases are defined
    - As a prelude to warehousing

Limitations of Data Lakes

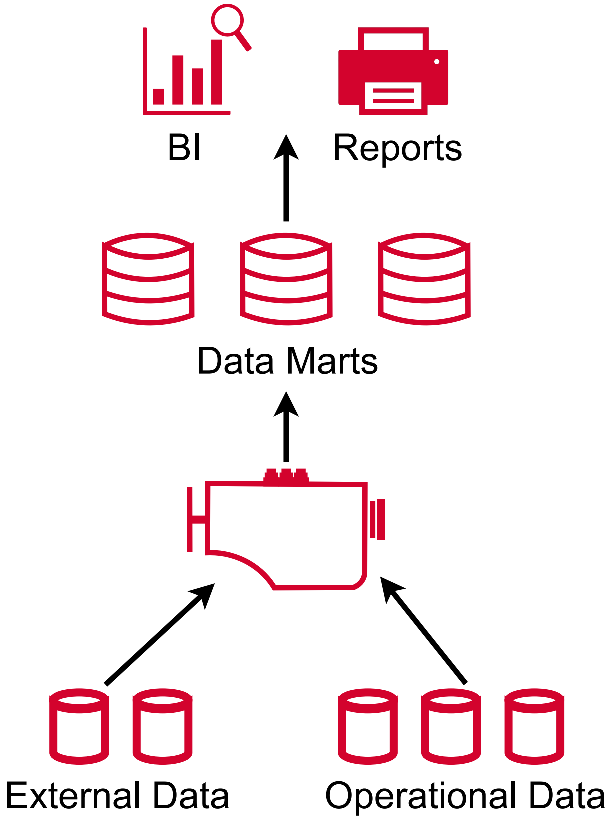
- Data is stored in its raw form
  - Needs to be processed on-the-fly for analysis
  - Slows down BI and analytics tasks
- Lack of structure can make data hard to find
- Do not support ACID transactions
- Unreliable data swamps

To satisfy all use cases, organizations often use a warehouse and a data lake - this brings us back to silos

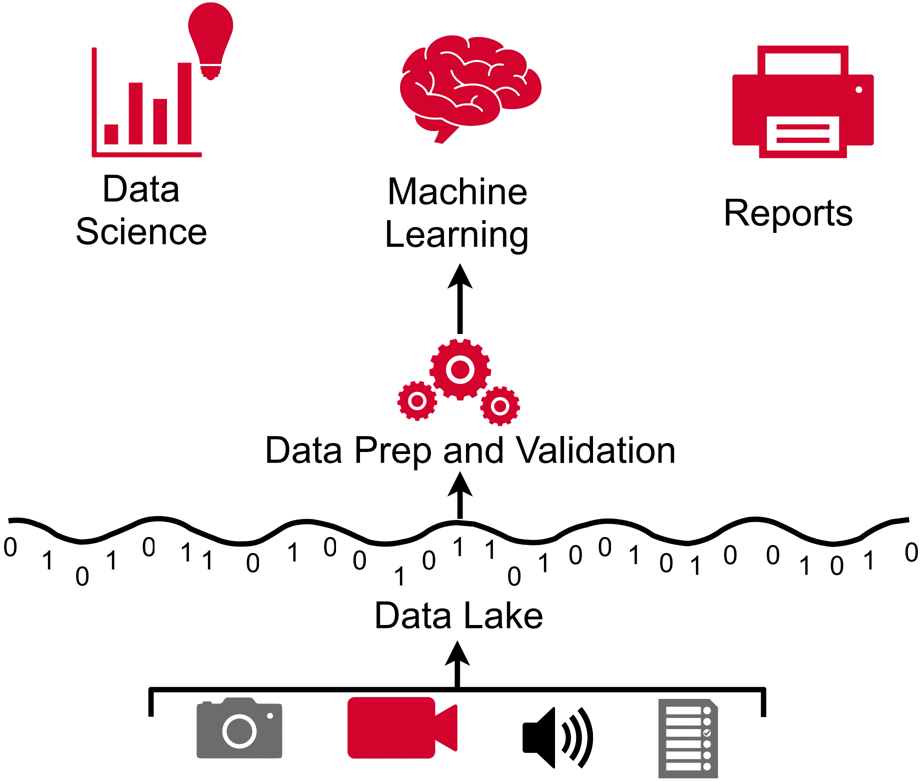
O'REILLY®

Data Lakehouses

Data Warehouse



Data Lake



Data Science

Machine Learning

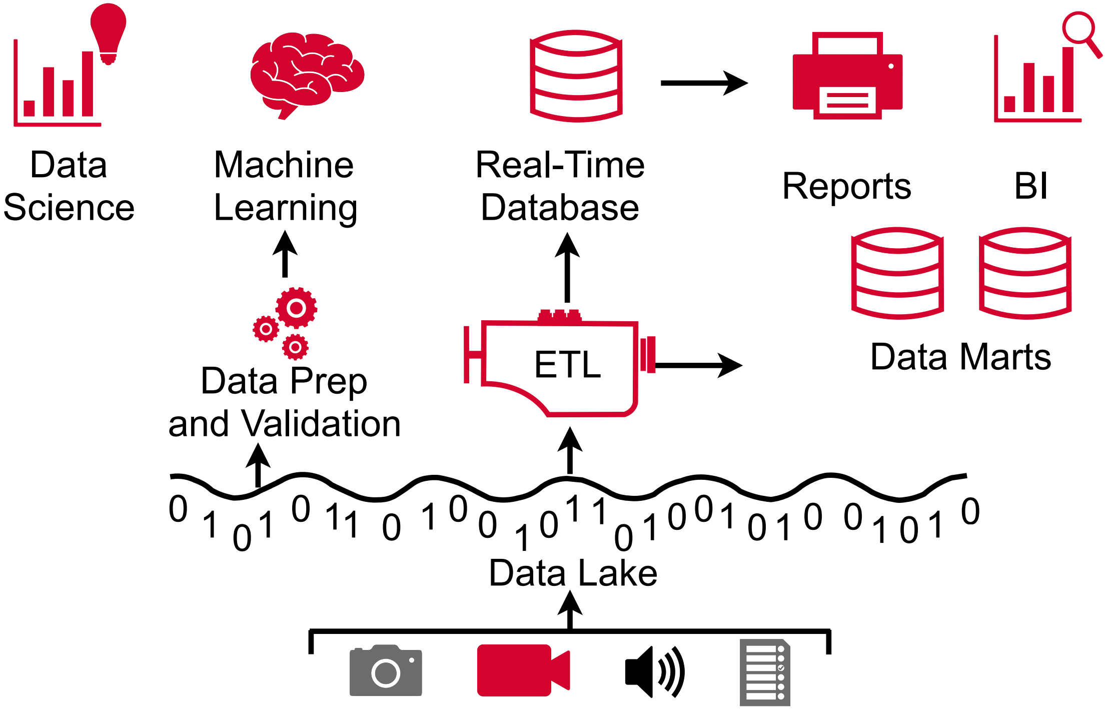
Reports

Data Prep and Validation

Data Lake

Structured, Semi-structured and Unstructured Data

What Would be Best for Users?



Data Science

Machine Learning

Real-Time Database

Reports

BI

Data Marts

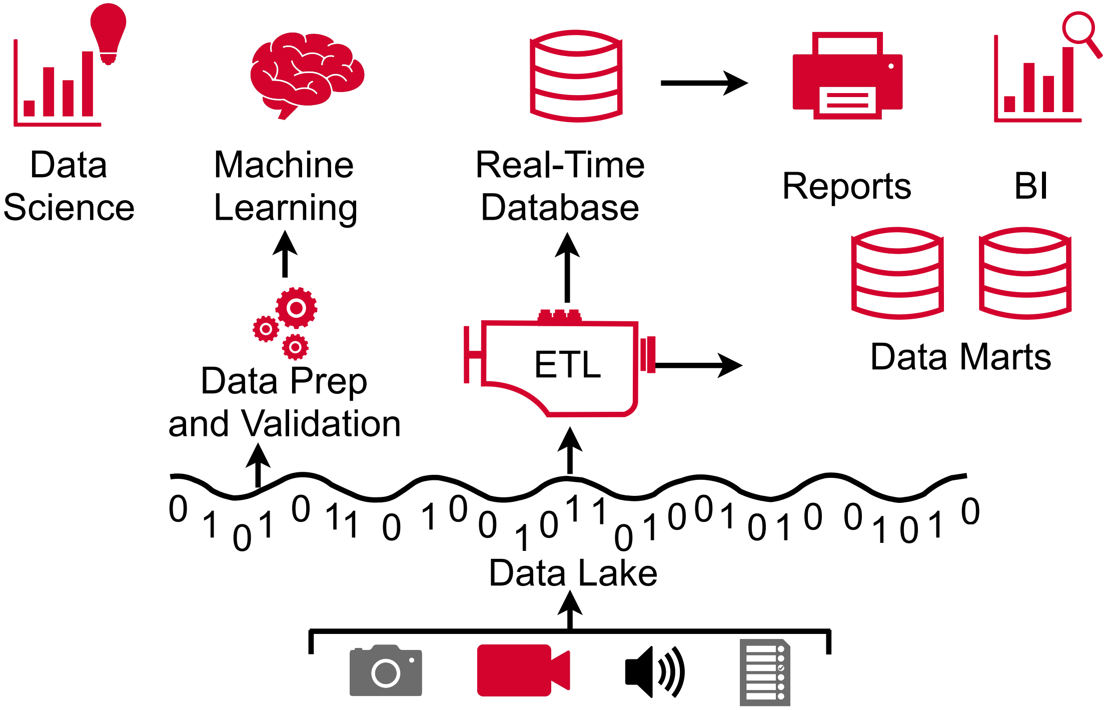
Data Prep and Validation

ETL

Data Lake

Structured, Semi-structured and Unstructured Data

Single System for All Use Cases



Data Science

Machine Learning

Real-Time Database

Reports

BI

Data Marts

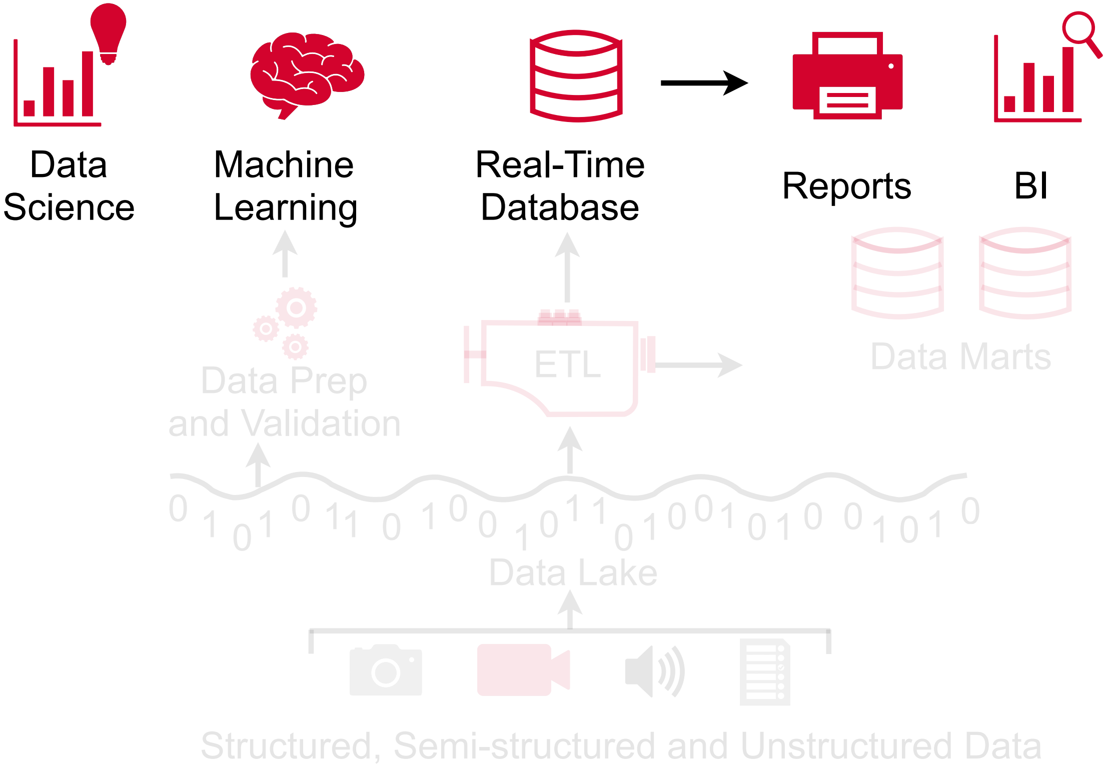
Data Prep and Validation

ETL

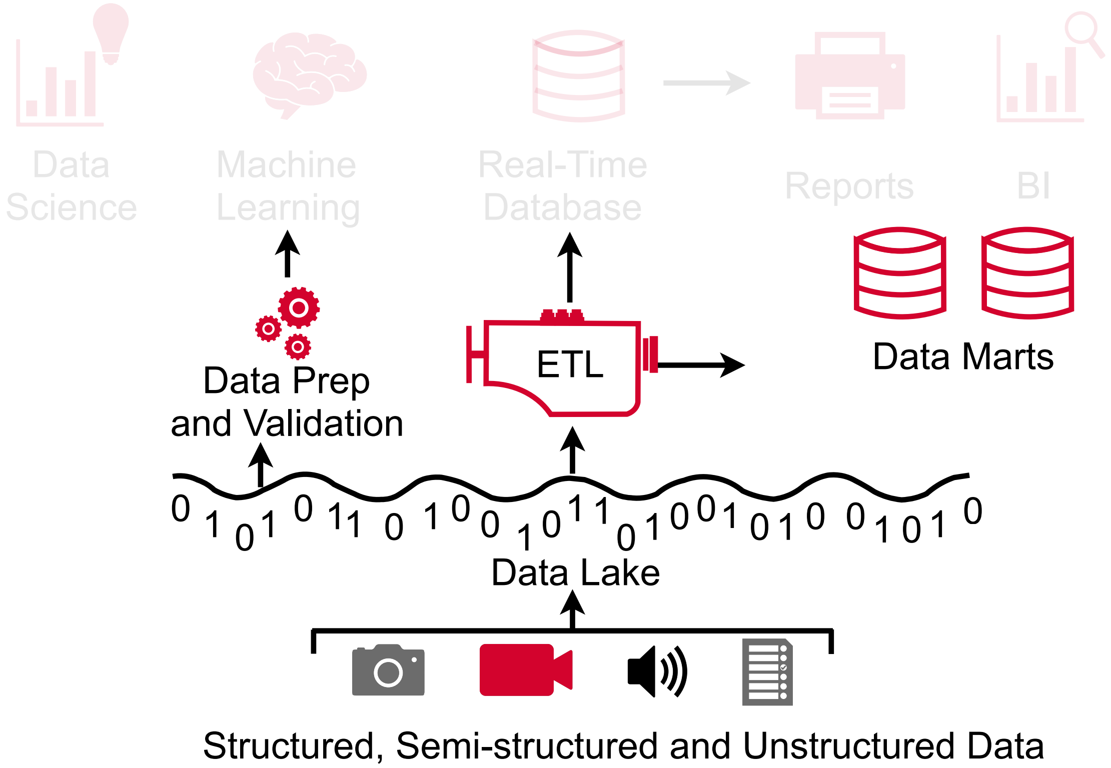
Data Lake

Structured, Semi-structured and Unstructured Data

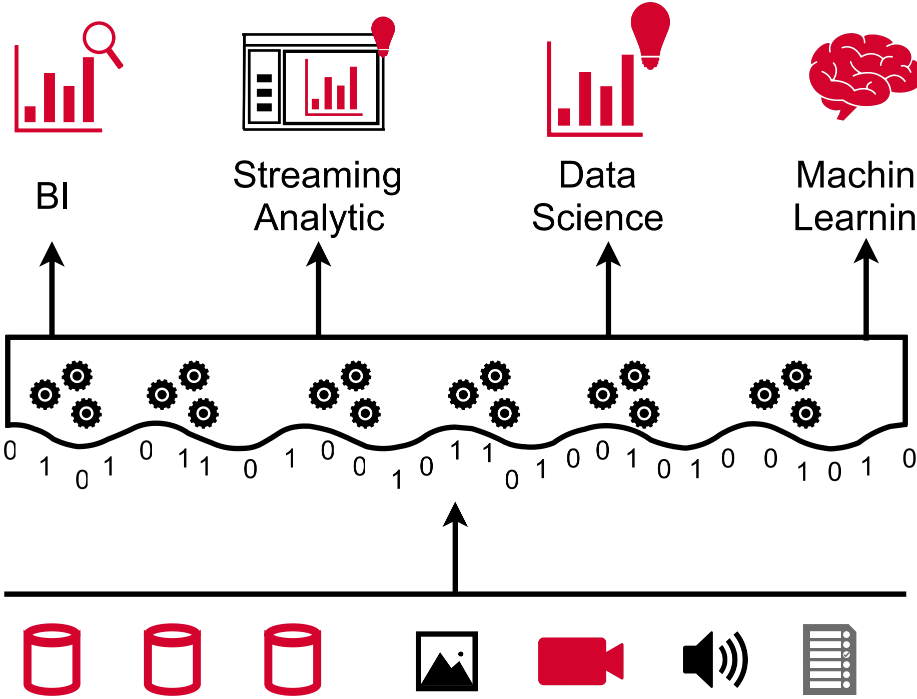
Business Analytics and Machine Learning



Hard in the Old Paradigm



Data Lakehouse



BI

Streaming Analytic

Data Science

Machine Learning

Structured, Semi-structured and Unstructured Data

What is a Data Lakehouse?

A data lakehouse is a new, open data management architecture that combines the flexibility, cost-efficiency, and scale of data lakes with the data management and ACID transactions of data warehouses, enabling business intelligence (BI) and machine learning (ML) on all data.

Source: https://databricks.com/glossary/data-lakehouse

What is a Data Lakehouse?

Data Lake

Data Warehouse

= Data Lakehouse

Data Lakehouse

The flexibility and cost of a data lake

Performance and reliability of a warehouse

Data Lakehouse

- One platform for all data
    - Unstructured data for ML
    - Structured data for BI
    - Batch and streaming
- Data reliability and consistency
- Open data management architecture - no proprietary formats
- As inexpensive as a data lake
- Support for diverse workloads (BI, Data Science, ML, Analytics)

How do Lakehouses Work?

- Metadata layers on top of data lakes
  - Data management and governance
  - ACID transactions
  - Access control and auditing
  - Schema enforcement
  - Data validation
  - Data versioning

What is a Data Lakehouse?

A data lakehouse is a new, open data management architecture that combines the flexibility, cost-efficiency, and scale of data lakes with the data management and ACID transactions of data warehouses, enabling business intelligence (BI) and machine learning (ML) on all data.

Source: https://databricks.com/glossary/data-lakehouse

Data Lakehouse is an architecture - to bring it to life, we can use a Delta Lake

Poll 1

Which of the following storage technologies is used to store unstructured data at a low cost?

- Database
- Schema
- Data warehouse
- Data lake

Poll 1

Which of the following storage technologies is used to store unstructured data at a low cost?

- Database
- Schema
- Data warehouse
- Data lake

Poll 2

Which of the following storage technologies is used to store processed data meant for analysis using business intelligence tools?

- Database
- Schema
- Data warehouse
- Data lake

Poll 2

Which of the following storage technologies is used to store unstructured data at a low cost?

- Database
- Schema
- Data warehouse
- Data lake

Delta Lake

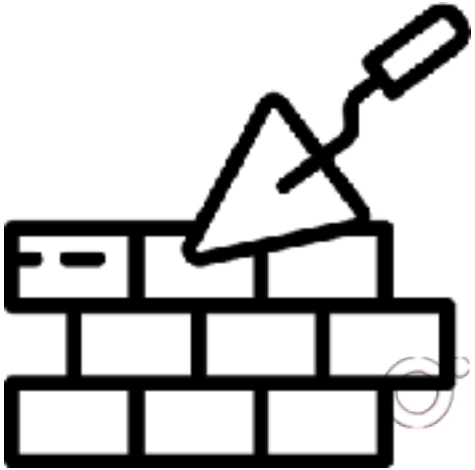
Data Lakehouse is an architecture - to bring it to life, we can use a Delta Lake

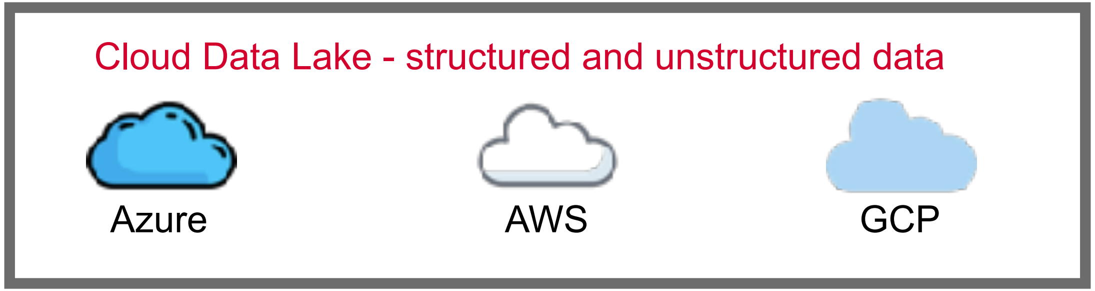
Delta Lake

Delta Lake is an open source storage layer which is which brings reliability to data lakes. This is how Data Lakehouses are built in Databricks.

Building a Lakehouse

- Raw data will be stored on a data lake such as AWS S3 or Azure Data Lake Storage
- Layers on top of the lake implement lakehouse features
- Data is stored in the open Apache Parquet format



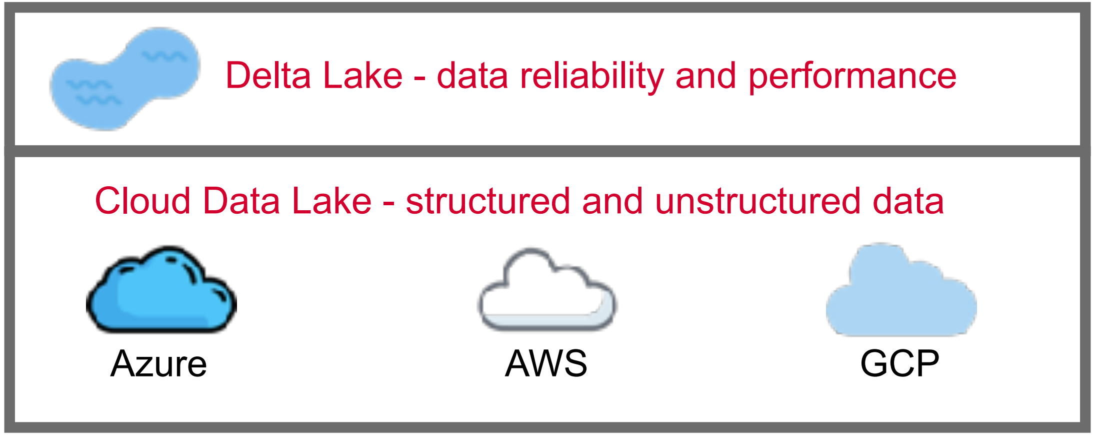


Cloud Data Lake - structured and unstructured data

Azure

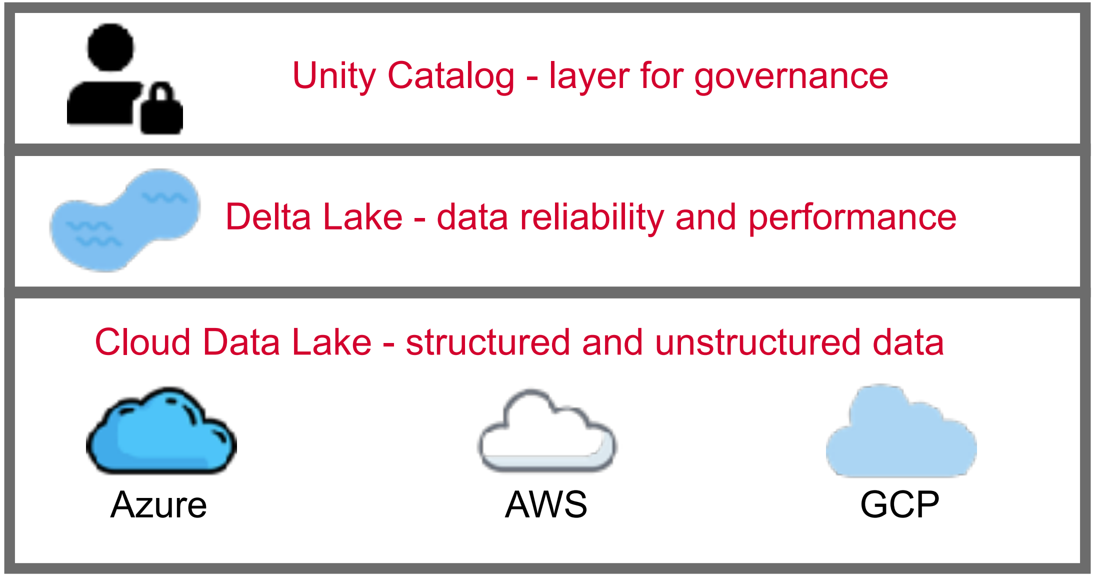
AWS

GCP



Delta Lake - data reliability and performance

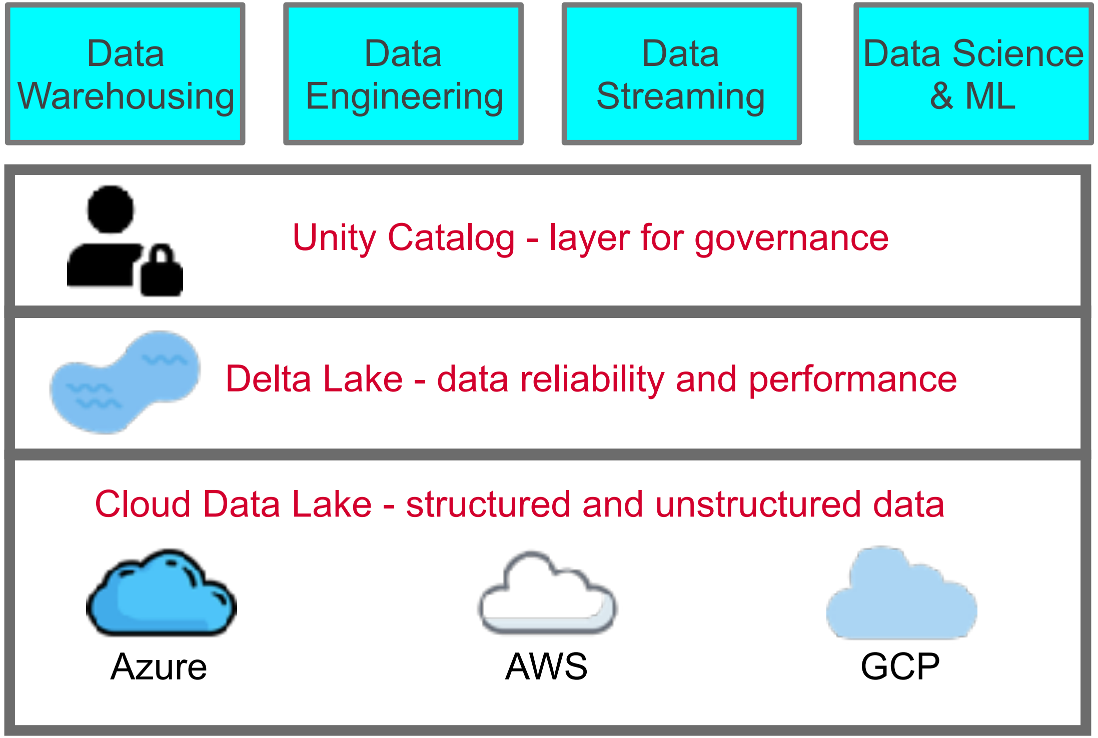
Cloud Data Lake - structured and unstructured data



Unity Catalog - layer for governance

Delta Lake - data reliability and performance

Cloud Data Lake - structured and unstructured data



Data Warehousing

Data Engineering

Data Streaming

Data Science & ML

Unity Catalog - layer for governance

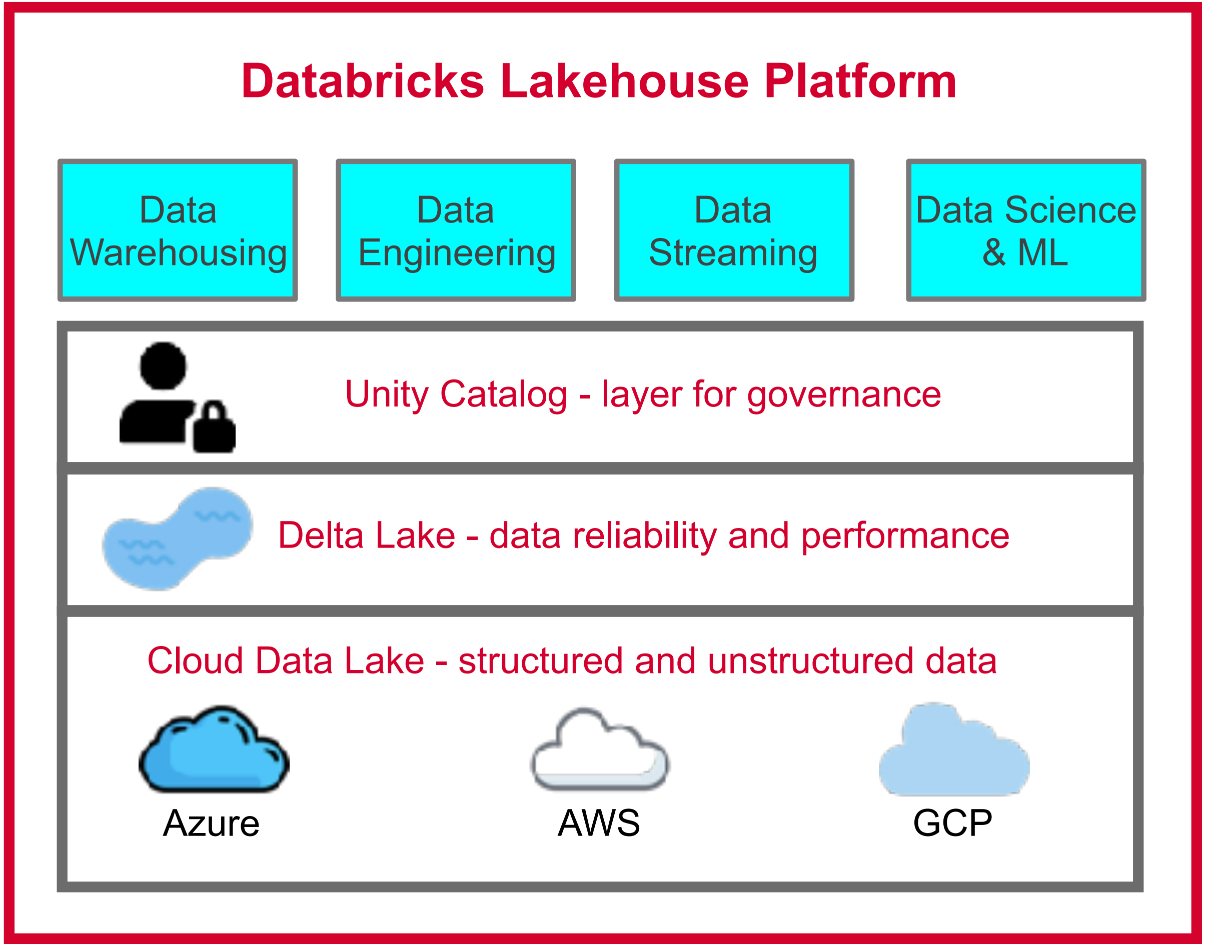
Delta Lake - data reliability and performance

Cloud Data Lake - structured and unstructured data

Azure

AWS

GCP



Databricks Lakehouse Platform

Data Warehousing

Data Engineering

Data Streaming

Data Science & ML

Unity Catalog - layer for governance

Delta Lake - data reliability and performance

Cloud Data Lake - structured and unstructured data

Azure

AWS

GCP

Databricks

A cloud-based platform that unifies your data warehousing, data science, and AI use cases

Databricks

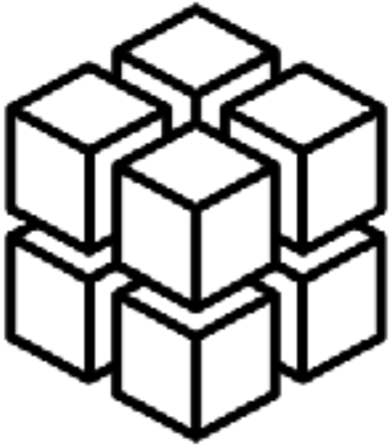
An enterprise software company founded by the creators of Apache Spark. Company employees have also created Apache Spark, Delta Lake, and MLflow, open source projects that span data engineering, data science, and machine learning.

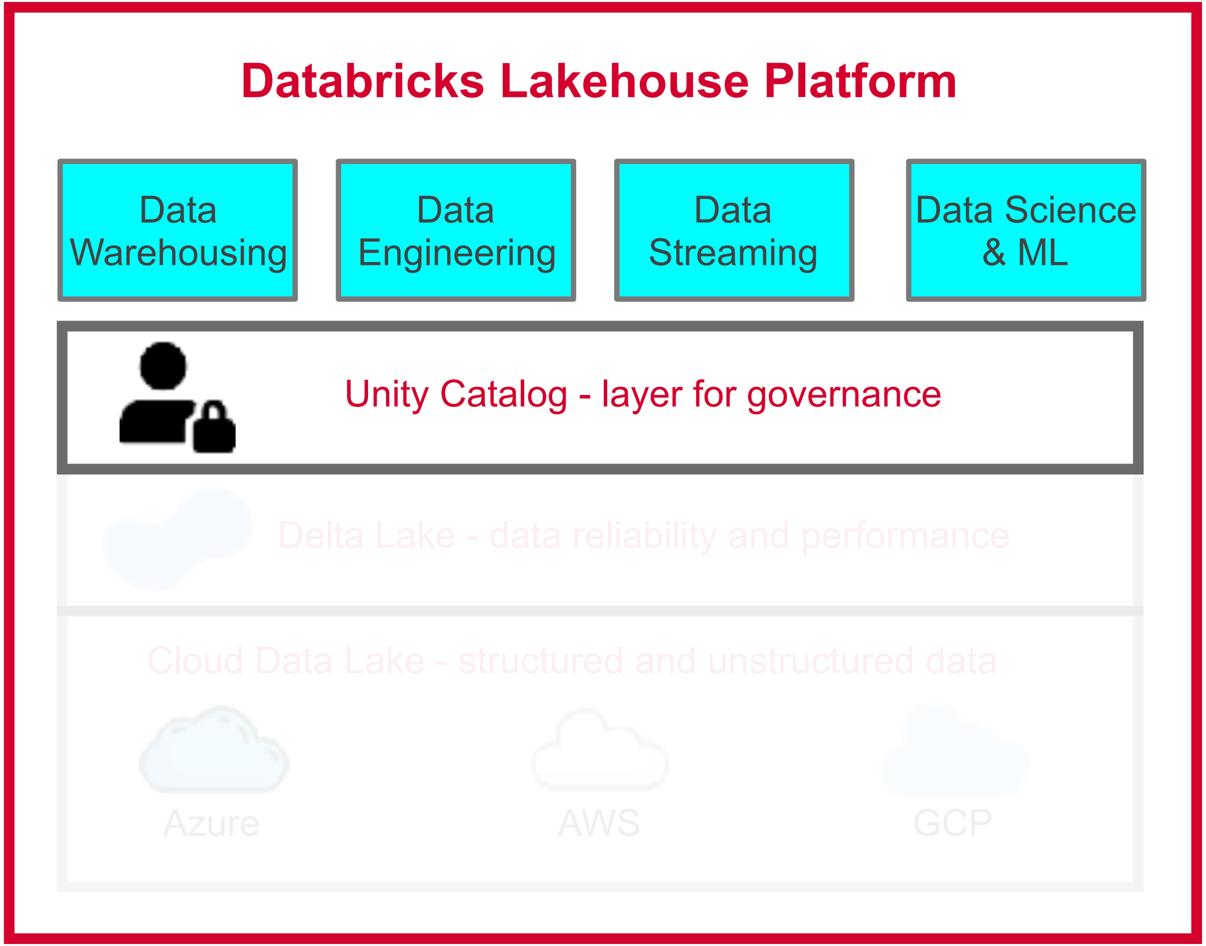
Databricks

- Web platform built using the Data Lakehouse architecture
- Supports Apache Spark as a big data processing engine
- Supports SQL for queries and analytics
- Supports ML frameworks and experimentation

Delta Lakes and Databricks

- Databricks natively supports Delta Lake
- Delta Lakes provide transaction layer in Data Lakehouse
- Create Delta tables for structured data
- Work with Delta tables using
  - SQL
  - Spark
  - Databricks UI





Databricks Lakehouse Platform

Data Warehousing

Data Engineering

Data Streaming

Data Science & ML

Unity Catalog - layer for governance

Delta Lake - data reliability and performance

Cloud Data Lake - structured and unstructured data

Azure

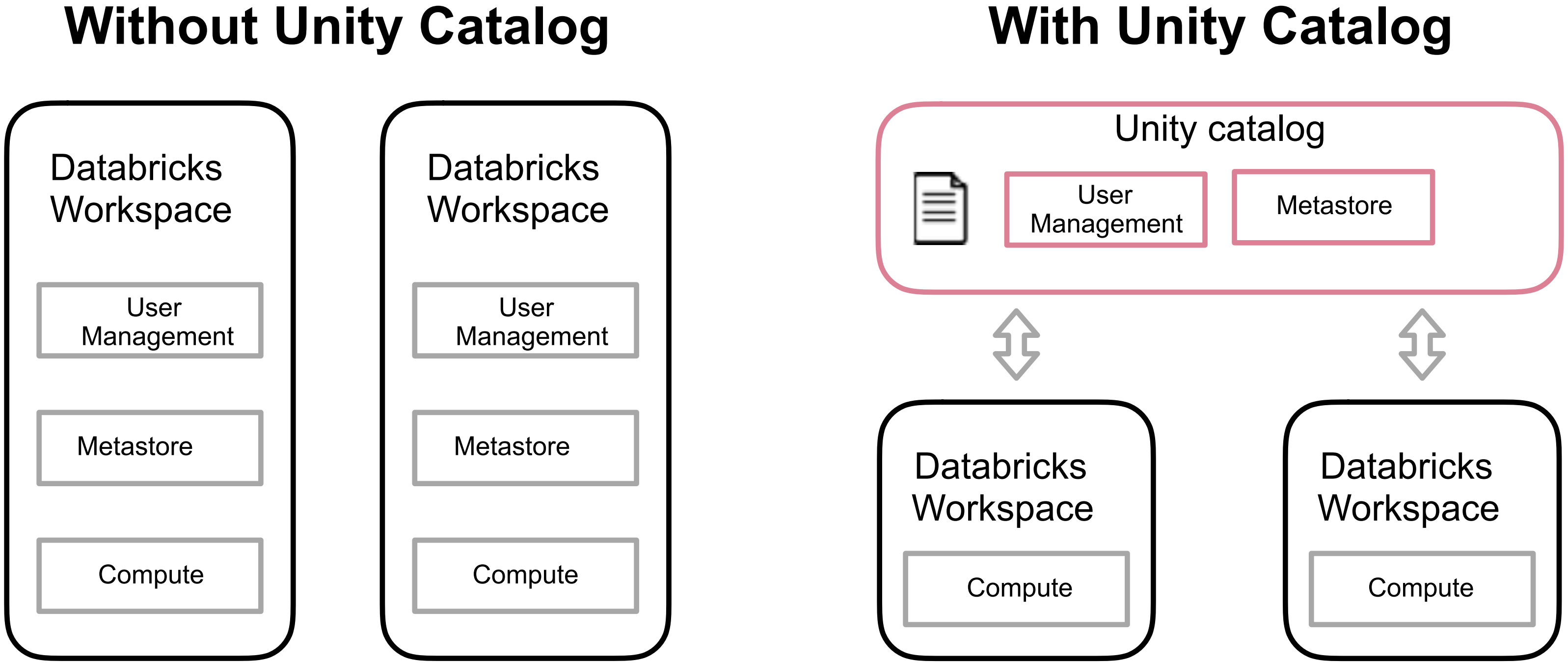
AWS

GCP

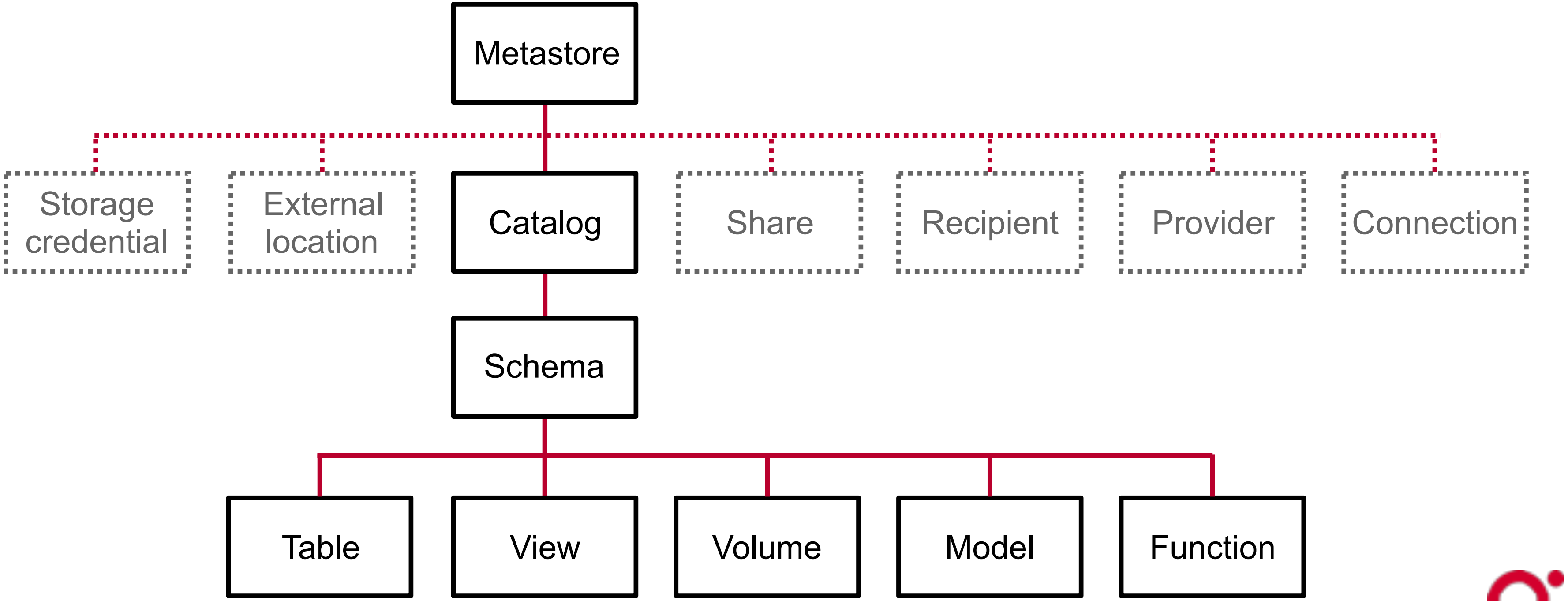
The Unity Catalog

Unity Catalog provides centralized access control, auditing, lineage, and data discovery capabilities across Databricks workspaces.

The Unity Catalog



The Unity Catalog Object Model



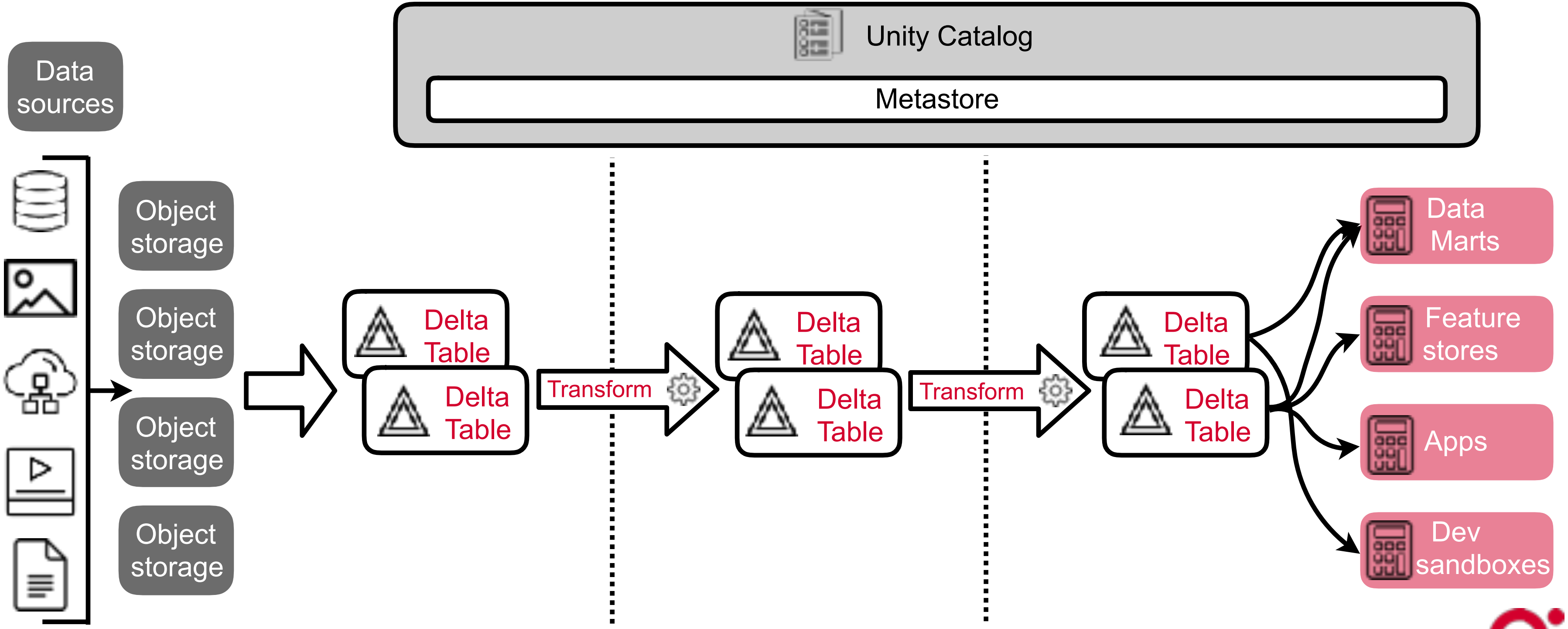
The Unity Catalog Object Model

- **Metastore**: The top-level container for metadata. Each metastore exposes a three-level namespace that organizes your data.
- **Catalog**: Used to organize your data assets.
- **Schema**: Also known as databases, schemas contain tables and views.
- **Tables, views, and volumes**: At the lowest level in the data object hierarchy are tables, views, and volumes. Volumes provide governance for non-tabular data.

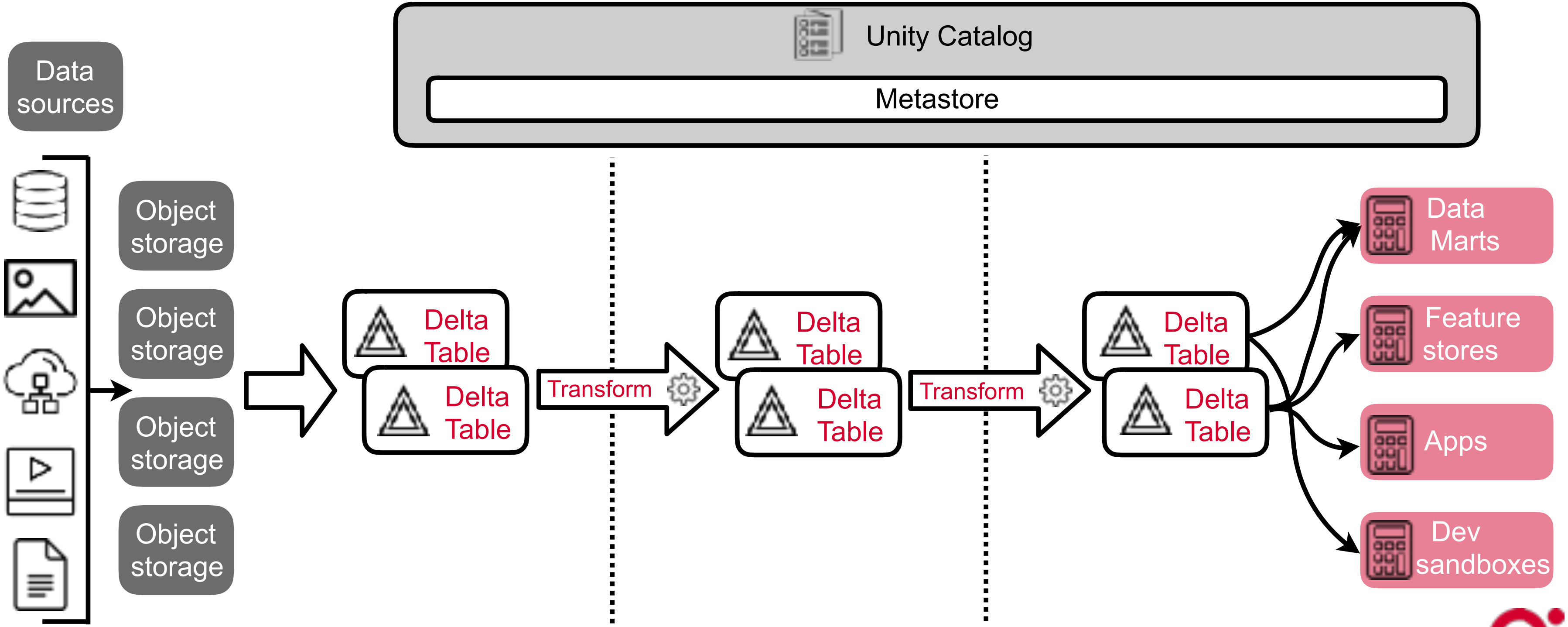
Medallion Architecture

The medallion architecture describes a series of data layers that denote the quality of data stored in the lakehouse. A multi-layered approach to building a single source of truth for enterprise data products.

Medallion Architecture



Bronze, Silver, and Gold Layers



Bronze, Silver, and Gold Layers

- **Bronze layer**:
  - Contains unvalidated raw data
  - Grows over time
- **Silver layer**:
  - Validated and deduplicated data
- **Gold layer**:
  - Highly refined and aggregated
  - Relied on by analysts

Hands-on Demos: Delta Lake on the Databricks Lakehouse Platform

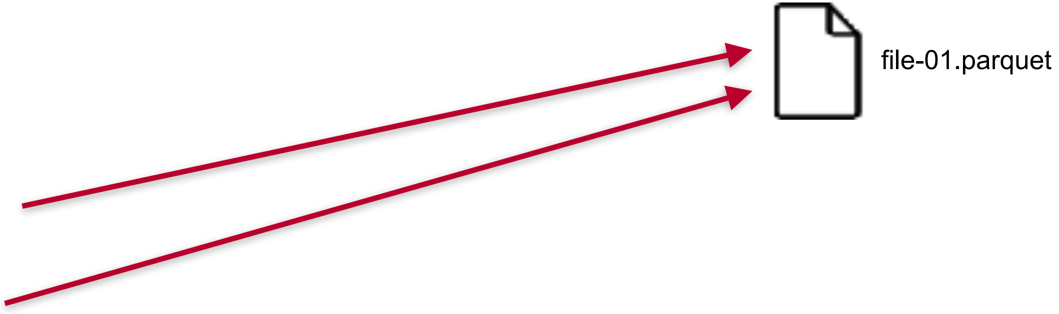
The DeltaLog

Delta Lake Transaction Log

Also known as the DeltaLog, it is an ordered record of every transaction performed on a Delta Lake table since its inception

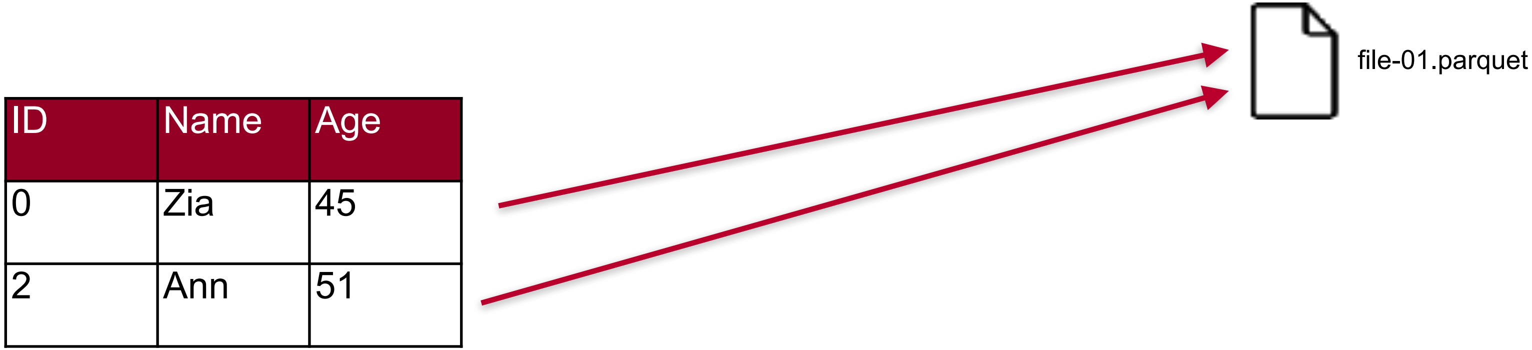
Data in a Delta Table

ID	Name	Age
0	Zia	45
2	Ann	51



Initial records are stored in a .parquet file

Data in a Delta Table



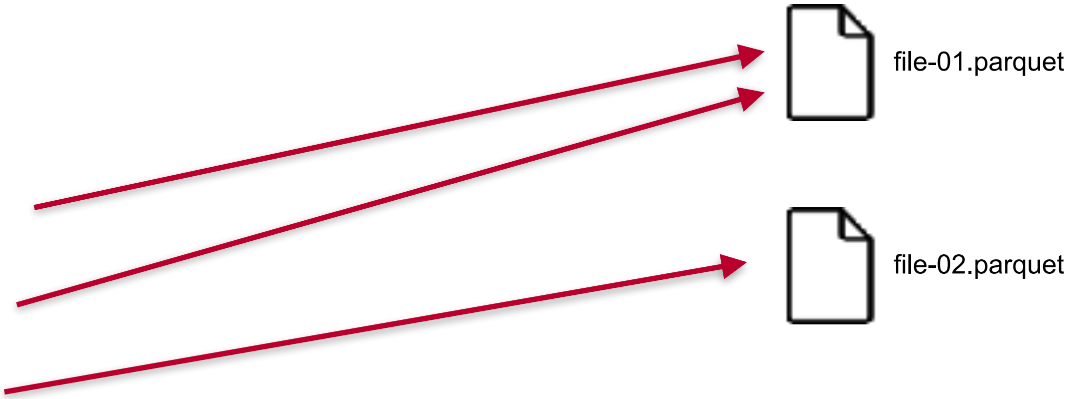
ID	Name	Age
0	Zia	45
2	Ann	51

file-01.parquet

The data will be stored in a columnar format

Data in a Delta Table

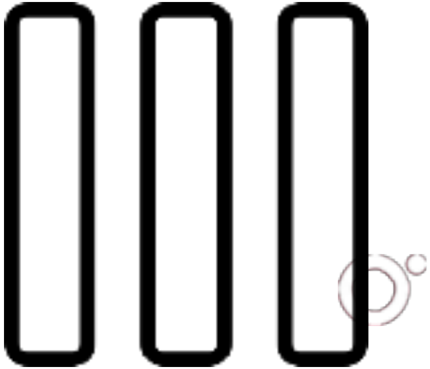
ID	Name	Age
0	Zia	45
2	Ann	51
3	Paul	26



Inserts may cause new .parquet files to be created

The Apache Parquet Format

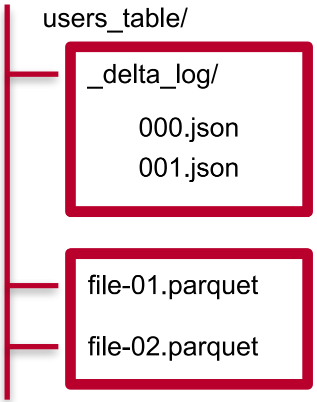
- A column-oriented storage format
- Free and open-source
- Values in a column are stored in contiguous memory locations
    - Quick data retrieval for column references
    - Compressing values of the same type is more efficient



The Delta Table

- Parquet + transaction log
    - Data stored in Parquet format
    - A transaction log to track changes to the data
    - Transactions are recorded as commits

Data in a Delta Table



users_table/

_delta_log/

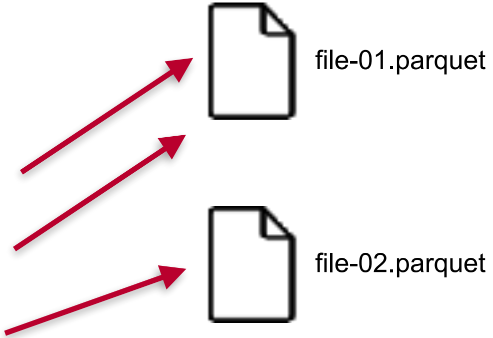
000.json

001.json

file-01.parquet

file-02.parquet

ID	Name	Age
0	Zia	45
2	Ann	51
3	Paul	26

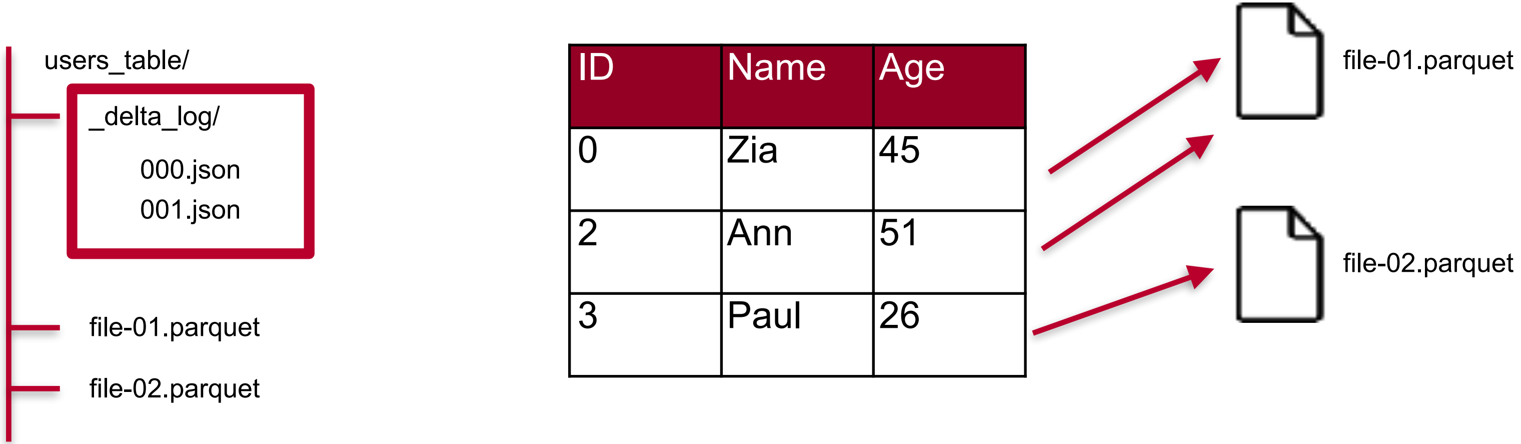


file-01.parquet

file-02.parquet

Parquet + Transaction Log

Data in a Delta Table

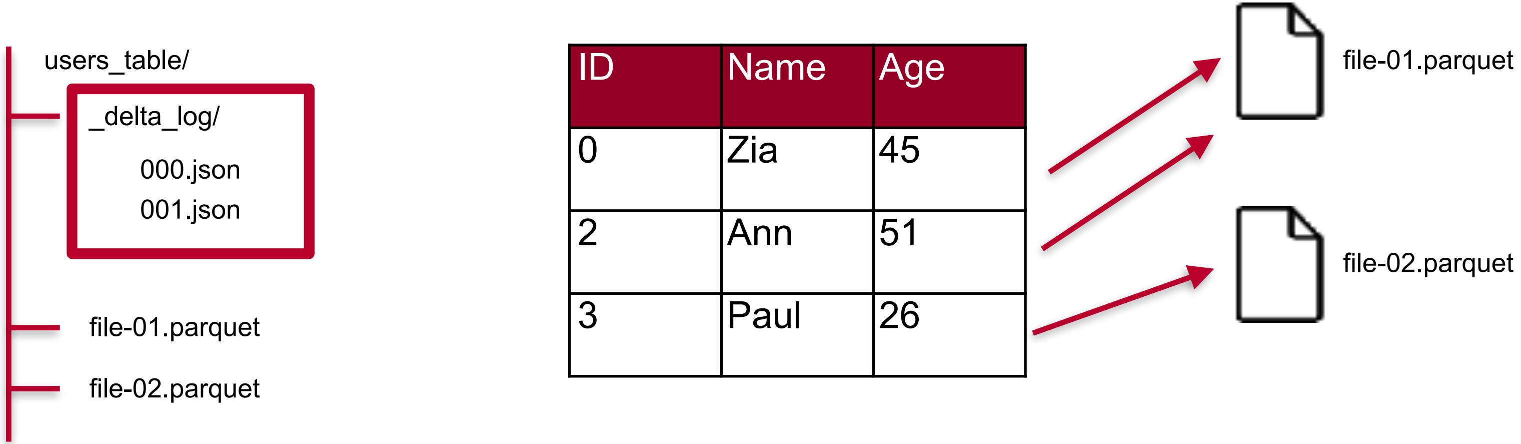


```
users_table/
    _delta_log/
        000.json
        001.json
    file-01.parquet
    file-02.parquet
```

ID	Name	Age
0	Zia	45
2	Ann	51
3	Paul	26

The _delta_log directory maintains commit data

Data in a Delta Table



users_table/

_delta_log/

000.json

001.json

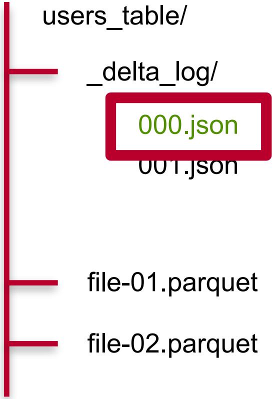
file-01.parquet

file-02.parquet

ID	Name	Age
0	Zia	45
2	Ann	51
3	Paul	26

Commits are recorded in JSON files

The DeltaLog



users_table/

_delta_log/

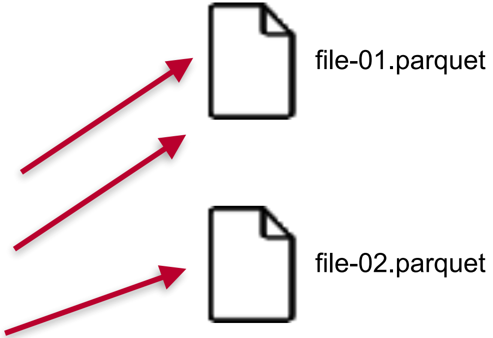
000.json

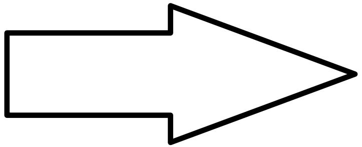
001.json

file-01.parquet

file-02.parquet

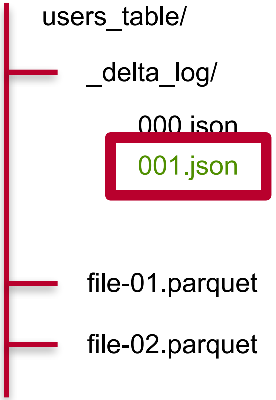
ID	Name	Age
0	Zia	45
2	Ann	51
3	Paul	26





000.json → Add file-01.parquet

The DeltaLog



users_table/

_delta_log/

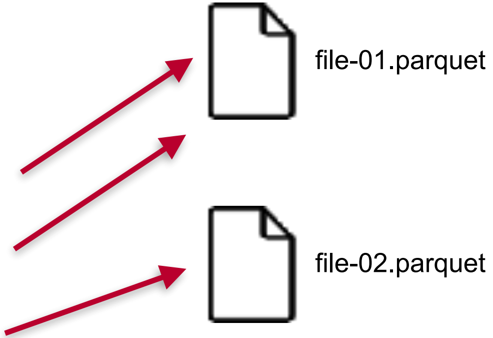
000.json

001.json

file-01.parquet

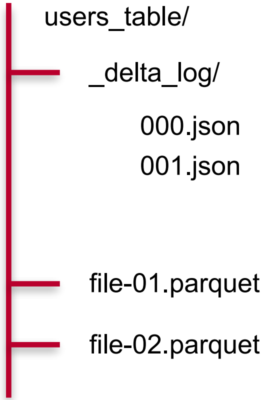
file-02.parquet

ID	Name	Age
0	Zia	45
2	Ann	51
3	Paul	26



001.json → Add file-02.parquet

The DeltaLog



users_table/

_delta_log/

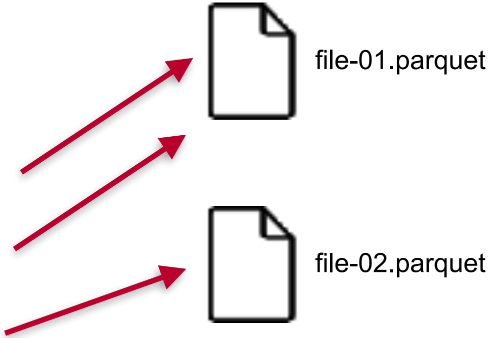
000.json

001.json

file-01.parquet

file-02.parquet

ID	Name	Age
0	Zia	45
2	Ann	51
3	Paul	26

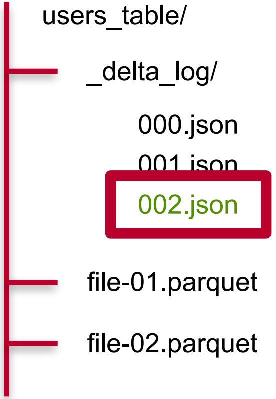


file-01.parquet

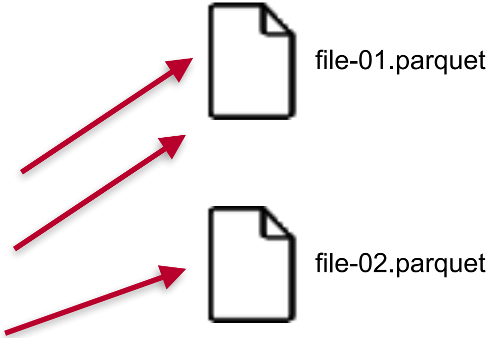
file-02.parquet

How is a delete of Paul’s row recorded?

The DeltaLog

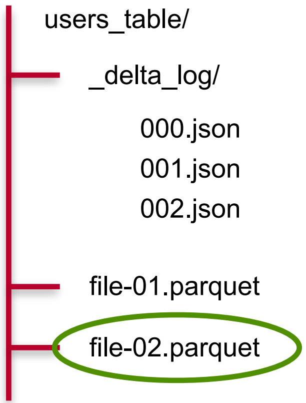


ID	Name	Age
0	Zia	45
2	Ann	51
3	Paul	26



002.json → Remove file-02.parquet

The DeltaLog



users_table/

_delta_log/

000.json

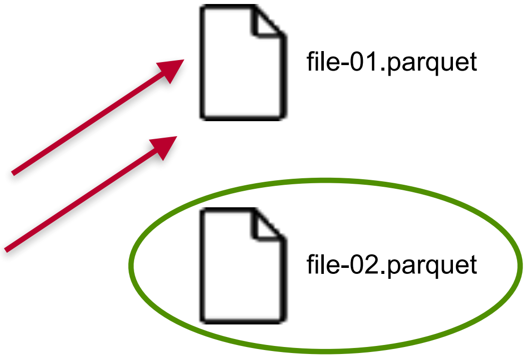
001.json

002.json

file-01.parquet

file-02.parquet

ID	Name	Age
0	Zia	45
2	Ann	51

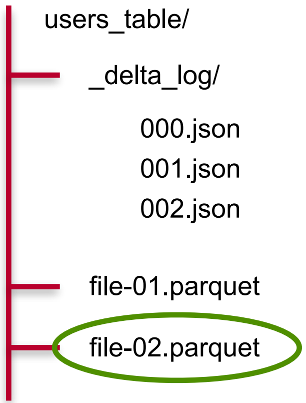


file-01.parquet

file-02.parquet

file-02.parquet is retained even though it is not referenced by the table

The DeltaLog



users_table/

_delta_log/

000.json

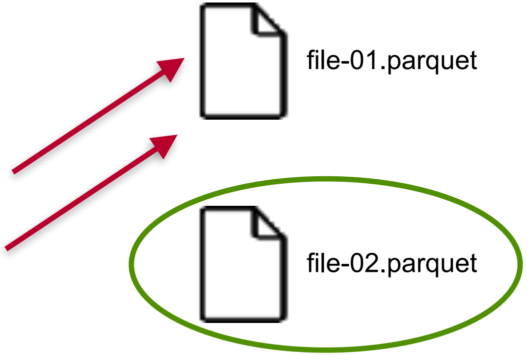
001.json

002.json

file-01.parquet

file-02.parquet

ID	Name	Age
0	Zia	45
2	Ann	51

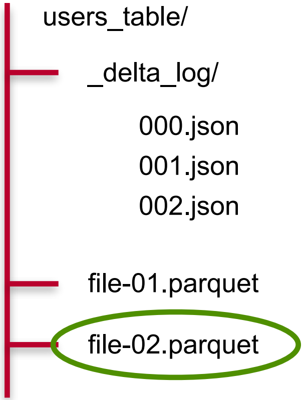


file-01.parquet

file-02.parquet

Retaining old files enables time-travel

The DeltaLog



users_table/

_delta_log/

000.json

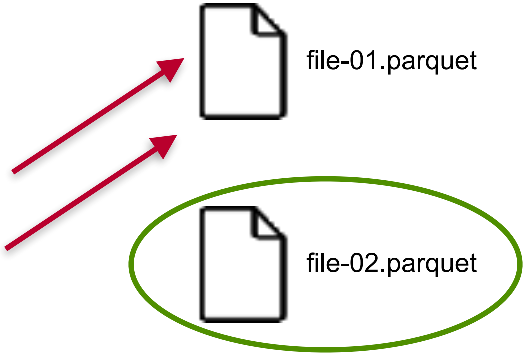
001.json

002.json

file-01.parquet

file-02.parquet

ID	Name	Age
0	Zia	45
2	Ann	51



Older files can be purged with a VACUUM operation

Benefits of the DeltaLog

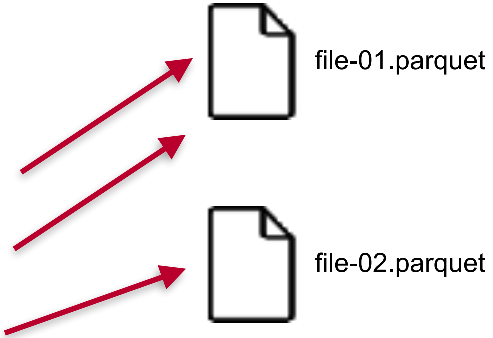
- Represents the single source of truth for a Delta table
- Is the guarantee of atomicity
    - Writes occur completely or not at all
    - A completed transaction will result in a log entry

Data in a Delta Table

```
users_table/
    _delta_log/
        000.json
        001.json

    file-01.parquet
    file-02.parquet
```

ID	Name	Age
0	Zia	45
2	Ann	51
3	Paul	26



Commits are recorded in JSON files

Time Travel

- View the state of a Delta table at a point in time
- Restore the table to a point in the past
- “Time” can be referenced with
    - Version number
        - e.g. “view table as of version 15”
    - Timestamp
        - e.g. “restore to 2022-01-01 05:00:00”

O'REILLY®

Hands-on Demos: Working with Delta Lake, Time Travel

Poll 3

Delta tables only support two kinds of constraints. Identify which of the constraints below will be enforced by Delta

- NOT NULL
- PRIMARY KEY
- FOREIGN KEY
- NOT EMPTY

Poll 3

Delta tables only support two kinds of constraints. Identify which of the constraints below will be enforced by Delta

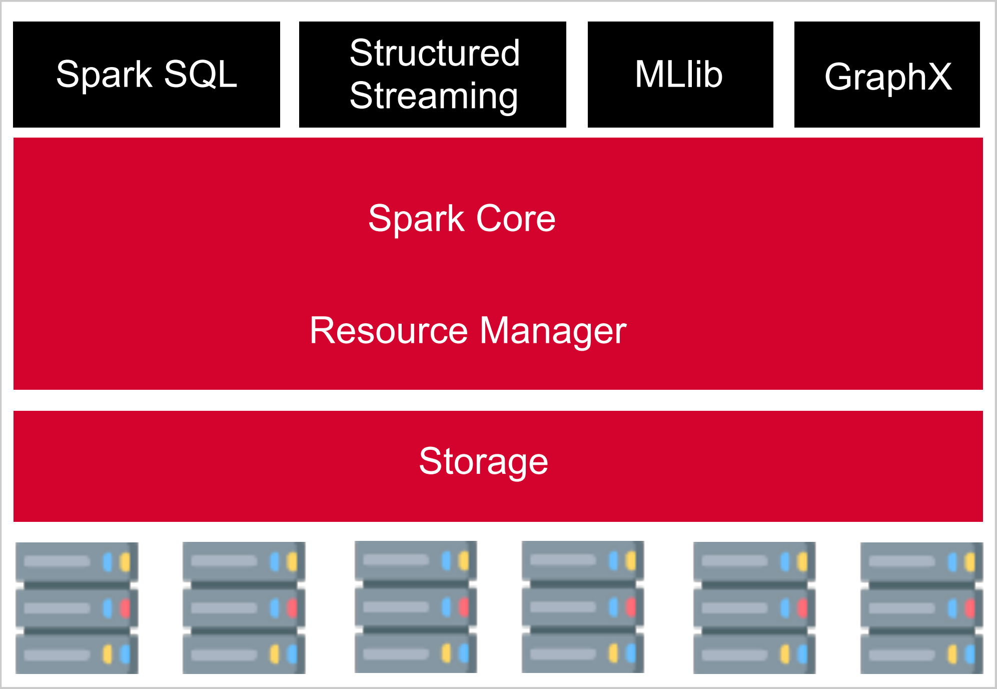
- NOT NULL
- PRIMARY KEY
- FOREIGN KEY
- NOT EMPTY

Apache Spark and SQL

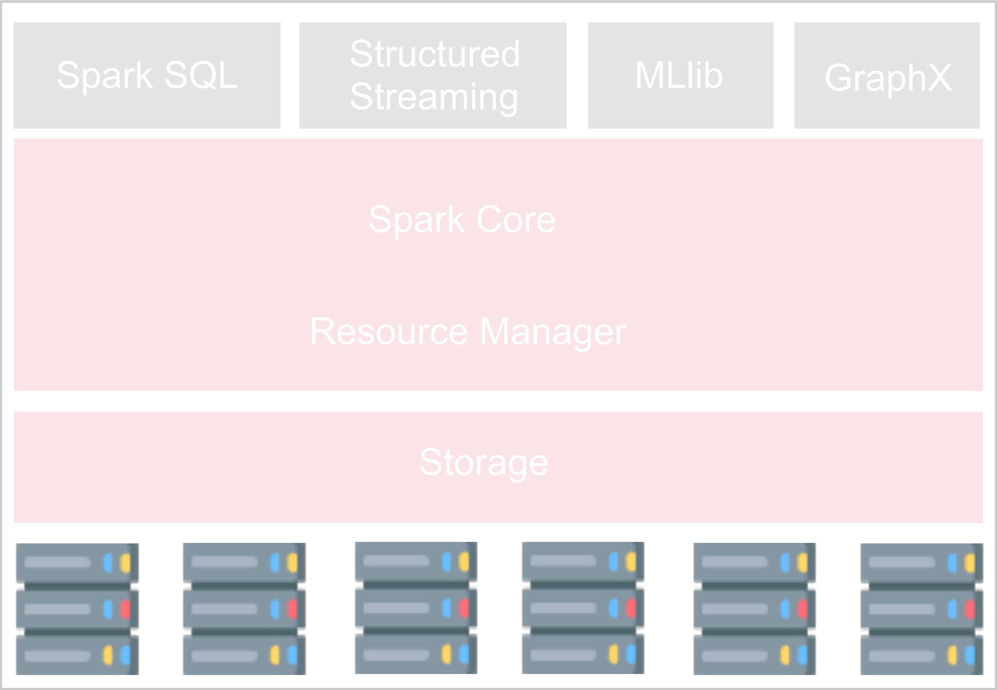
Apache Spark

A unified processing engine for big data analytics on batch as well as streaming data

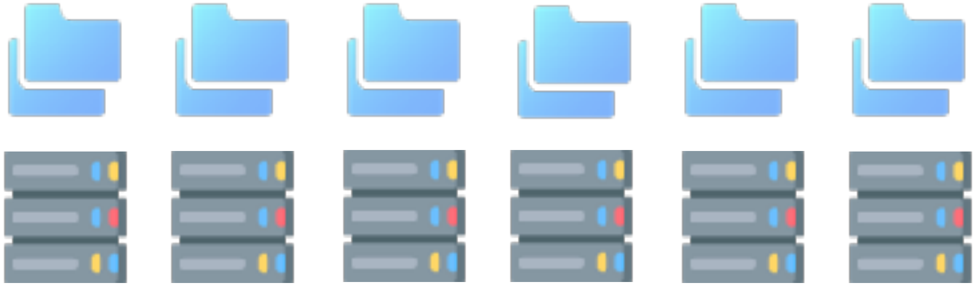
Apache Spark Components



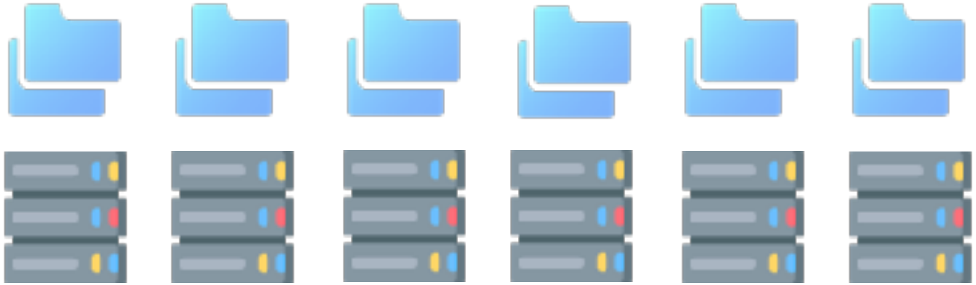
Distributed Processing Engine



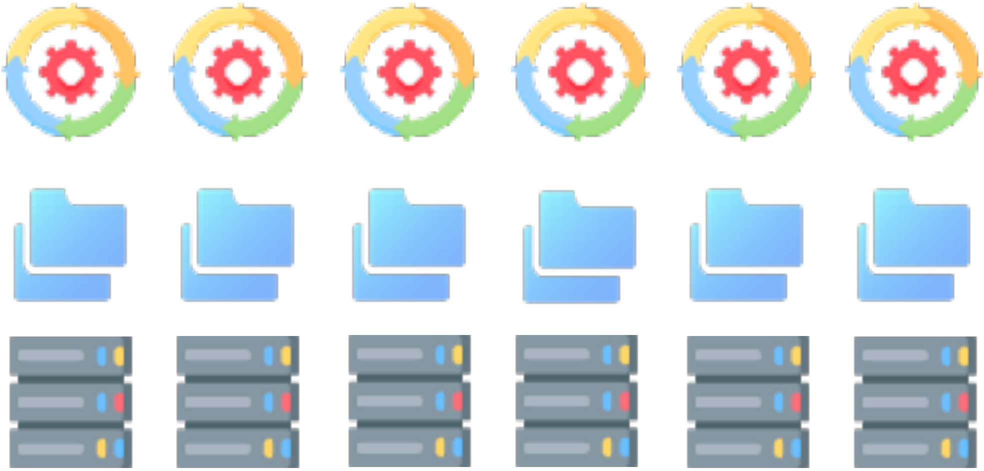
Data Split Across Cluster Nodes



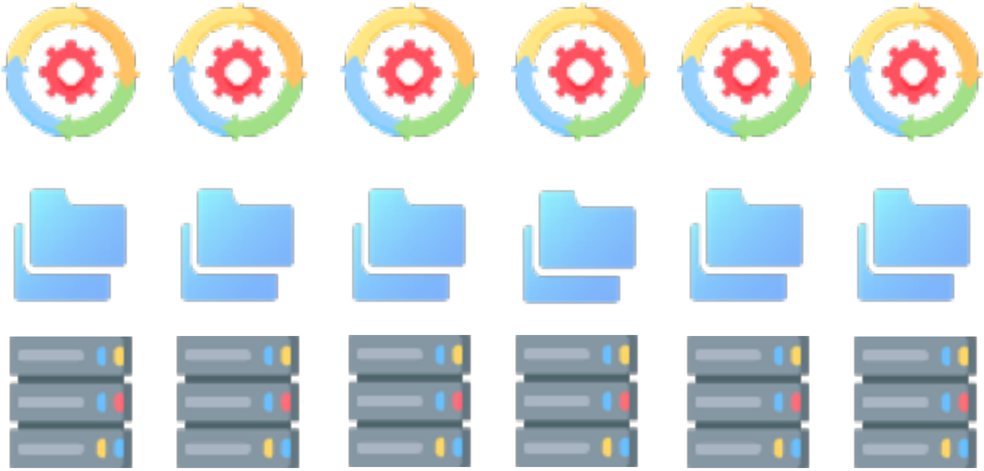
Each Split == Partition



Spark Process Operates on Split



Processes Run in Parallel



Apache Spark on Databricks

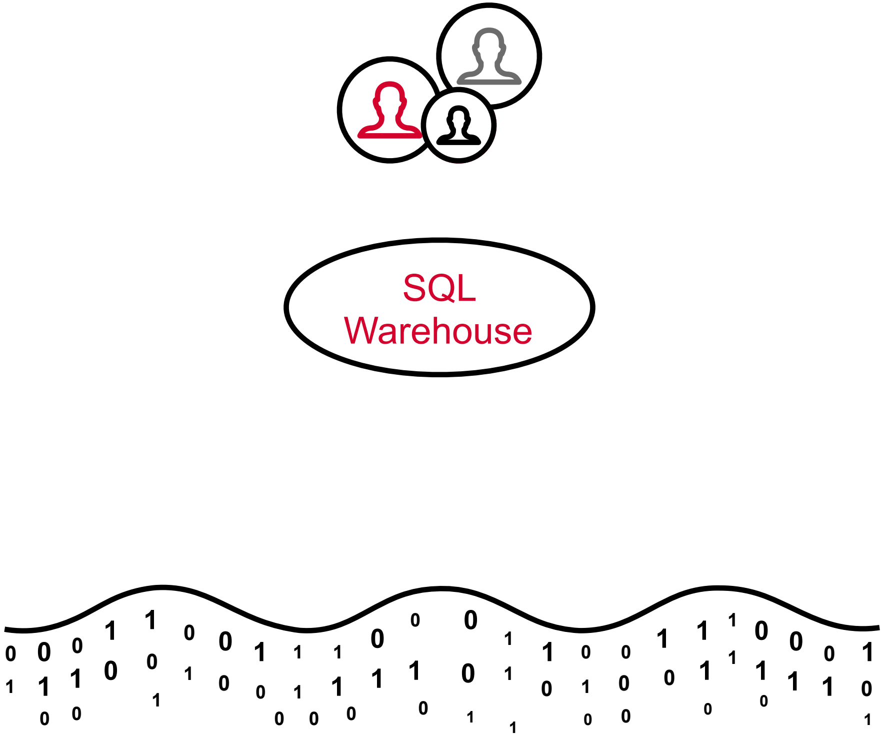
The data processing core for Databricks is Apache Spark. Integral part of the Data Lakehouse Architecture.

Databricks creates a managed, autoscaled Apache Spark environment, allowing you to prototype and run your code using Jupyter notebooks.

Databricks SQL

SQL environment for analysts to run SQL queries on data, create visualizations, share dashboards

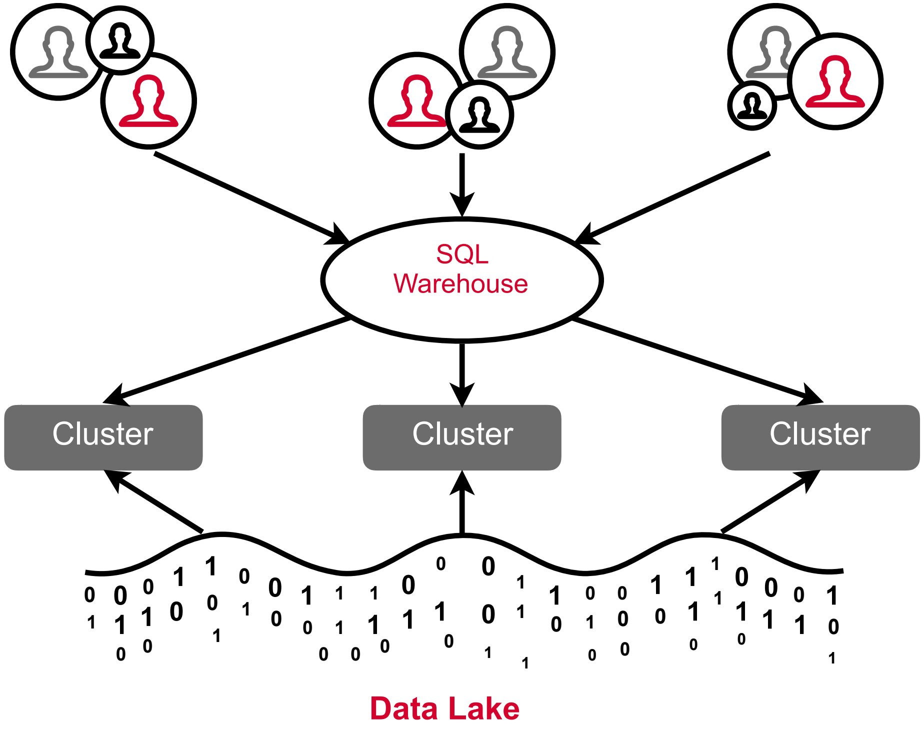
SQL Warehouses



SQL Warehouse

Data Lake

SQL Warehouses: Autoscaling

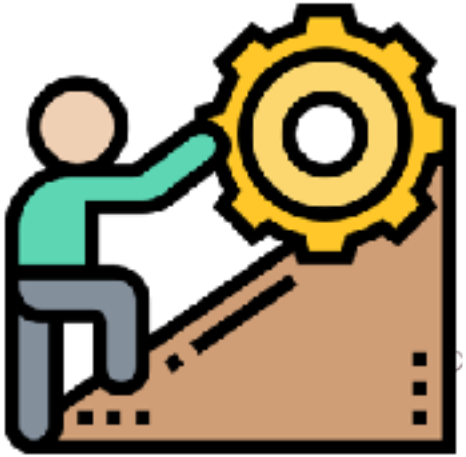


Hands-on Demos: Databricks Spark, Databricks SQL

Loading Data into a Lakehouse

Challenges of Data Ingestion

- Diversity - treat each source format differently
- Speed - minimize use of expensive cloud resources
- Complexity - data sources change over time
- Semantics - interpretation of data may change



Data Ingestion Best Practices

- Single view of data - data from multiple sources accessible in one place
- Support for unreliable networks - data buffering for reliable ingestion

Data Ingestion Best Practices

- Low latency - update data regularly and in small batches
- Auditable - maintain intermediate state for audits

Auto Loader and COPY INTO

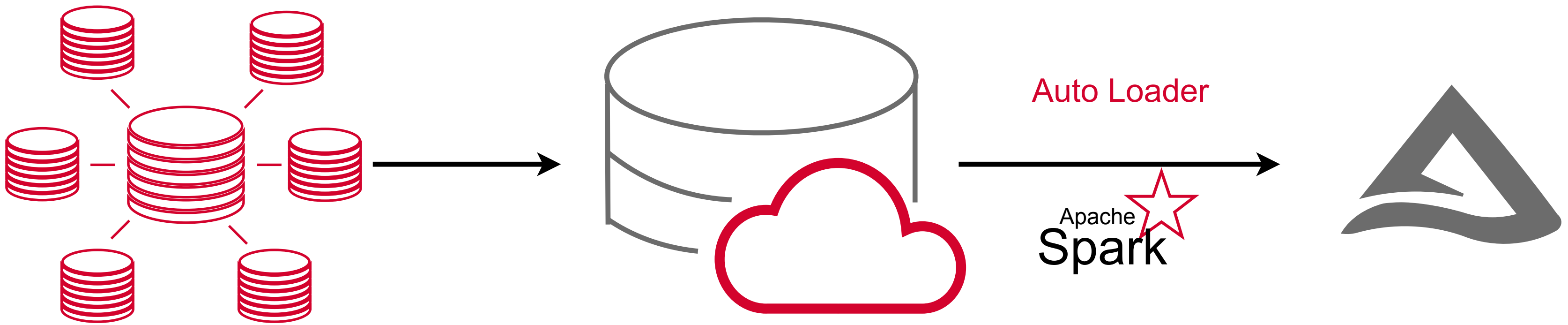
Auto Loader

Auto Loader is an optimized file source for Apache Spark that processes data incrementally and efficiently as they arrive on cloud storage.

Auto Loader

- Mechanism to add data to a Delta table incrementally
- Available as Python or Scala methods
- Offers a structured streaming source called cloudFiles
  - Does not need to run continuously
- Can process data at scale
- Ensures that data is processed exactly once
- Includes optimizations to reduce cloud costs

Loading Data with Auto Loader

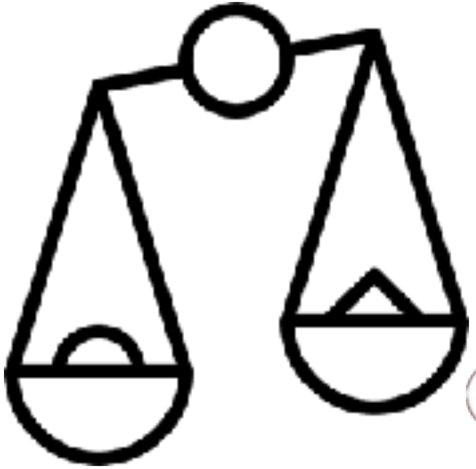


Ingesting Data with COPY INTO

- Command executed as SQL statement
- One more mechanism to ingest data into a Delta table
- Data copied with exactly-once guarantees

Auto Loader vs. COPY INTO

- Size of files to be processed
    - COPY INTO for smaller datasets
    - Auto Loader for millions of files over time
- Schema evolution
    - Auto Loader better with inferencing schema
- Re-upload of files
    - COPY INTO offers more flexibility



Hands-on Demos: Loading Data - Scheduled Queries and Auto Loaders

Poll 4

Which of the following statements about Auto Loader is false?

- Maintains state as to which files were already read
- Can track schema as it evolves
- Can be used to process millions of files
- Meant only for batch data

Poll 4

Which of the following statements about Auto Loader is false?

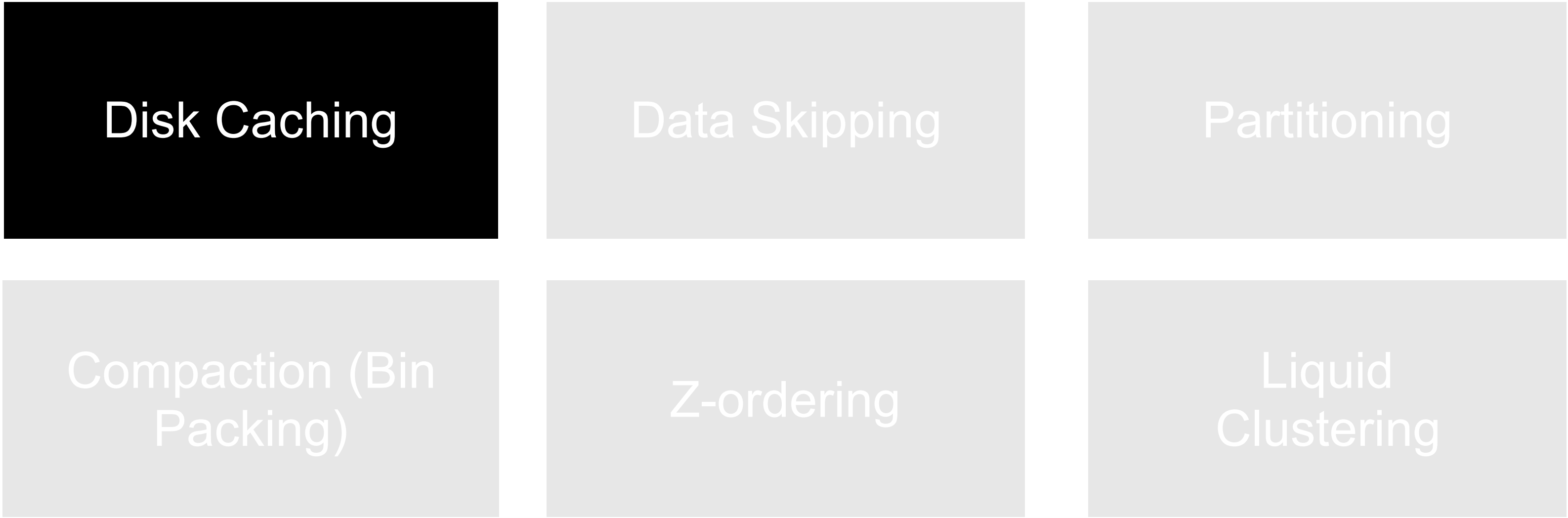
- Maintains state as to which files were already read
- Can track schema as it evolves
- Can be used to process millions of files
- Meant only for batch data

Delta Lake Optimizations

Optimizing Delta Lake Operations

- Disk Caching
- Data Skipping
- Partitioning
- Compaction (Bin Packing)
- Z-ordering
- Liquid Clustering

Disk Caching



Disk Caching

Data Skipping

Partitioning

Compaction (Bin Packing)

Z-ordering

Liquid Clustering

Delta Caching

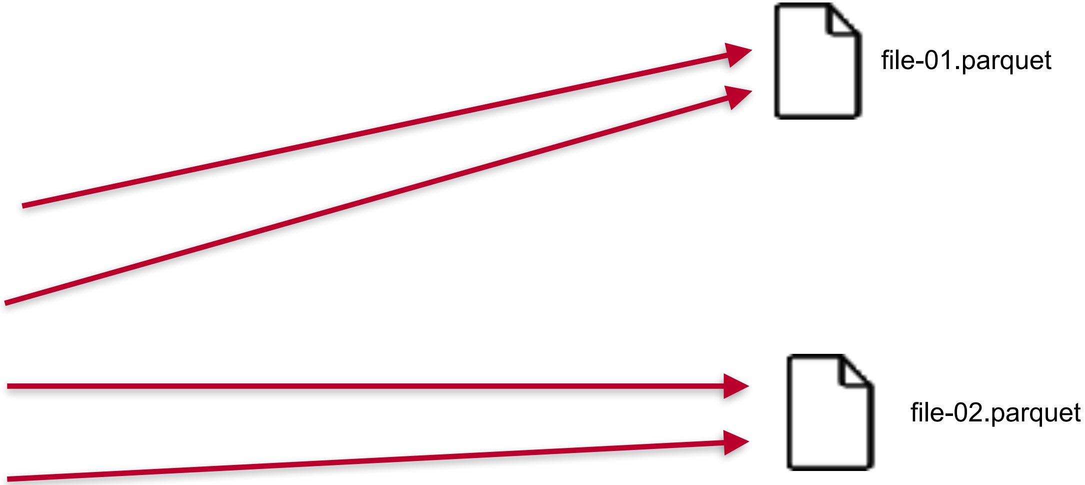
- Stores copies of remote data as local files on worker nodes
- Caches in a faster intermediate data format
- Available for any parquet-based storage

Data Skipping

- Disk Caching
- **Data Skipping**
- Partitioning
- Compaction (Bin Packing)
- Z-ordering
- Liquid Clustering

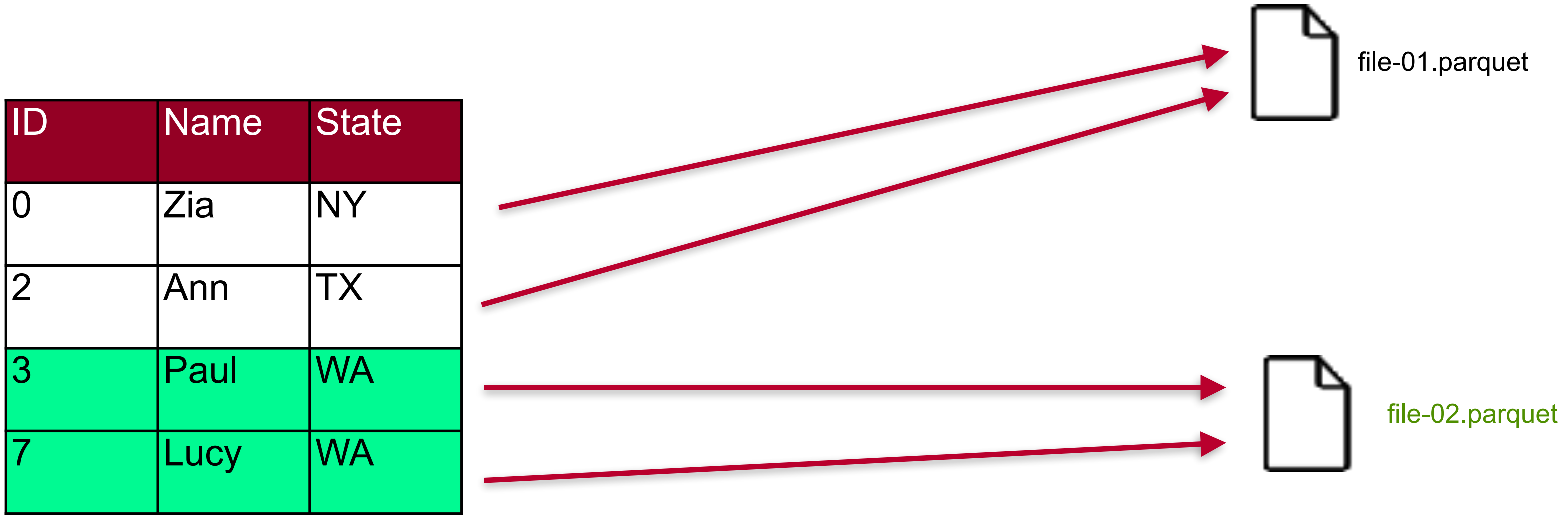
Data Skipping

ID	Name	State
0	Zia	NY
2	Ann	TX
3	Paul	WA
7	Lucy	WA



Consider a query where we search for users in WA

Data Skipping

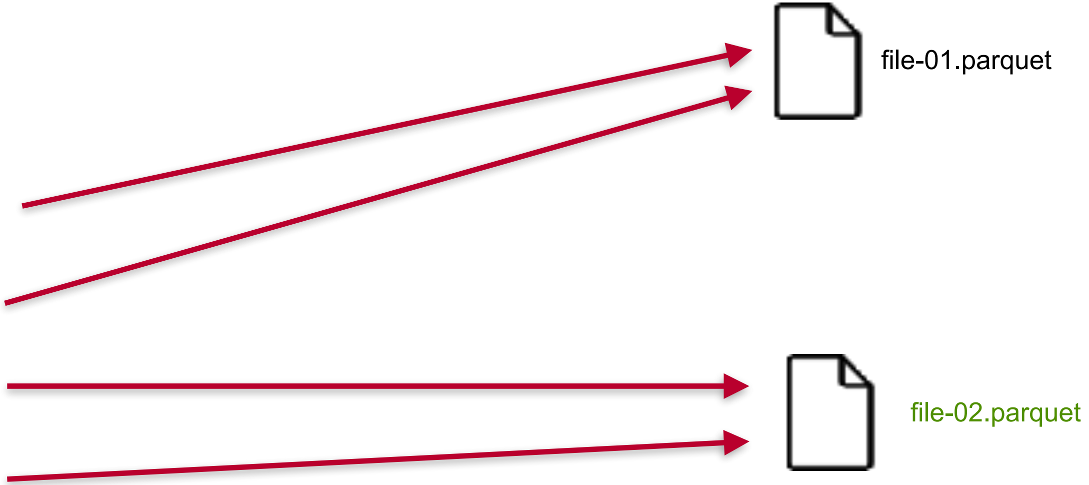


ID	Name	State
0	Zia	NY
2	Ann	TX
3	Paul	WA
7	Lucy	WA

All the data is present in one file

Data Skipping

ID	Name	State
0	Zia	NY
2	Ann	TX
3	Paul	WA
7	Lucy	WA



file-01 should be skipped for an optimal search

Data Skipping

- Uses file-level data to decide whether to scan a file
    - e.g. min and max values for columns
- Avoids irrelevant file scans
- Optimizes search operations
- Implemented in Delta tables by default
    - May include bloom filters

File Layout Optimizations

Disk Caching

Data Skipping

Partitioning

Compaction (Bin Packing)

Z-ordering

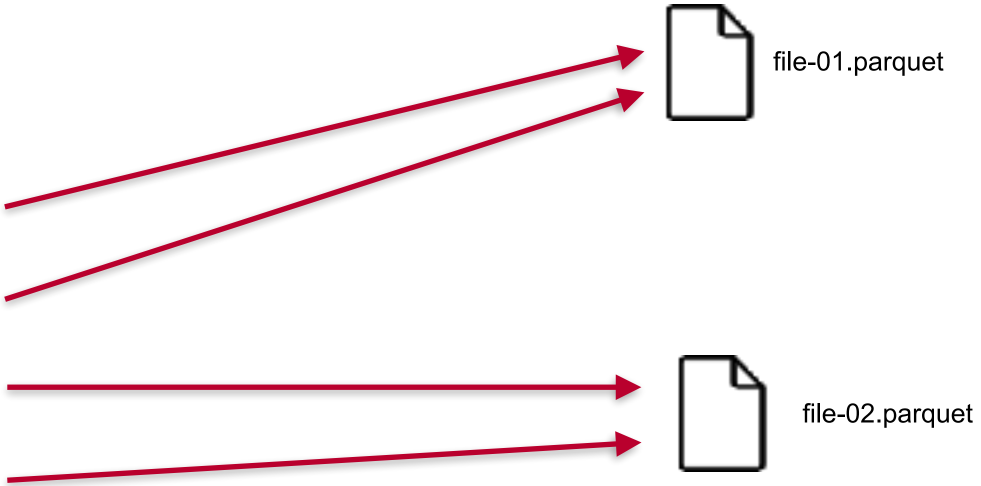
Liquid Clustering

File Layout Optimizations

Disk Caching	Data Skipping	Partitioning
Compaction (Bin Packing)	Z-ordering	Liquid Clustering

Partitioning

ID	Name	State
1	Faye	WA
2	Ann	TX
3	Paul	WA
7	Lucy	WA



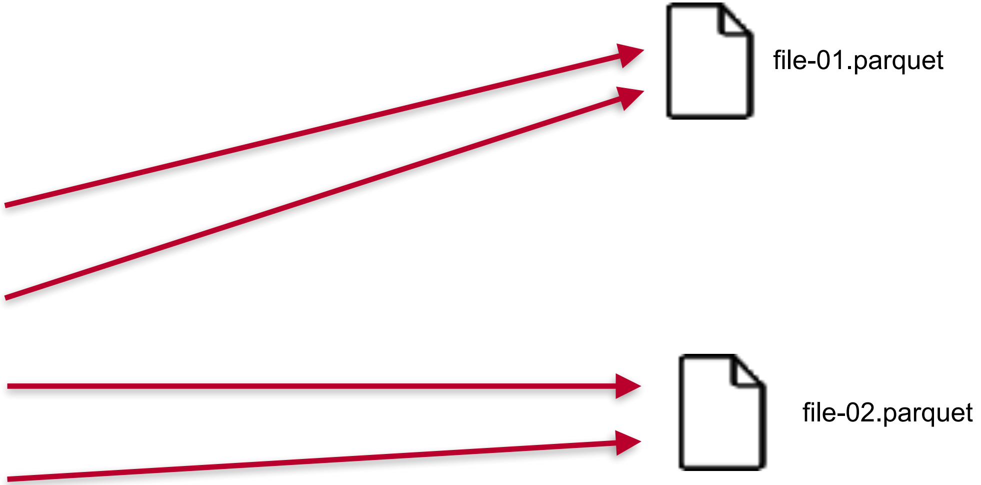
file-01.parquet

file-02.parquet

Consider a query where we search for users in WA

Partitioning

ID	Name	State
1	Faye	WA
2	Ann	TX
3	Paul	WA
7	Lucy	WA



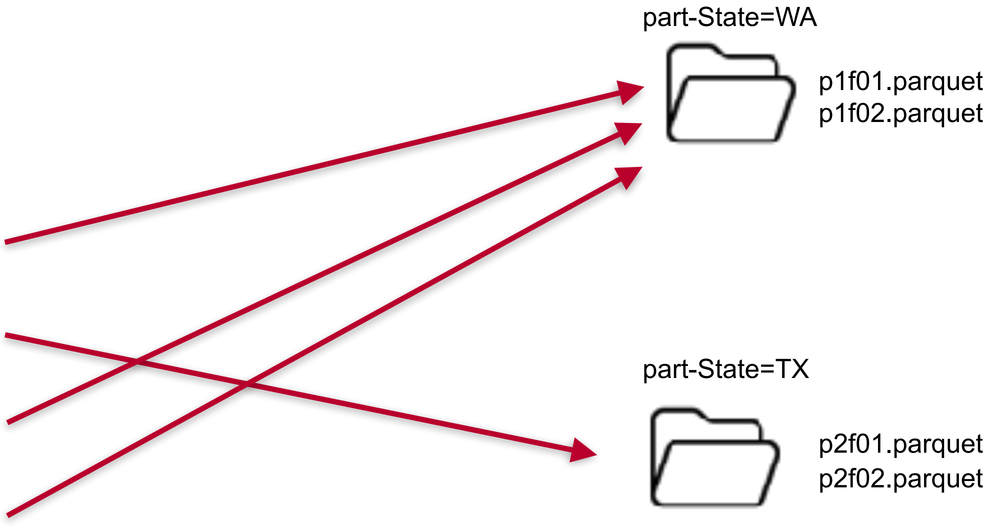
file-01.parquet

file-02.parquet

Both files will need to be scanned

Partitioning

ID	Name	State
1	Faye	WA
2	Ann	TX
3	Paul	WA
7	Lucy	WA



Partitioning by State groups all WA users together

Partitioning Data

- Break up data into partitions based on one or more columns
- Each partition is a folder in underlying storage
- Each partition contains rows with the same values for the partitioning columns
- Searches based on partitioned columns will only scan relevant partitions

Partitioning Data

- The challenge is to find the right partition column
  - Splits the data evenly
  - Referenced often in queries

File Layout Optimizations

Disk Caching

Data Skipping

Partitioning

Compaction (Bin Packing)

Z-ordering

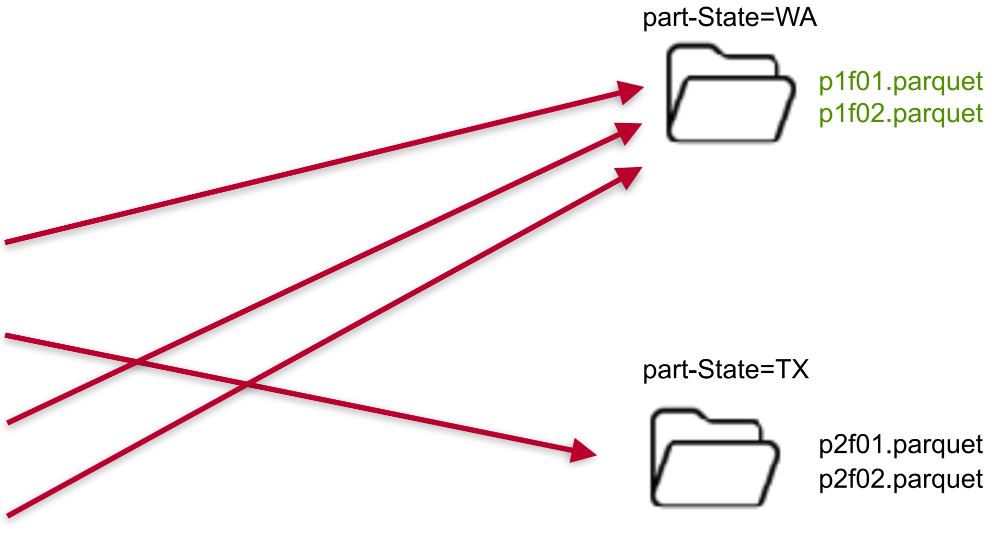
Liquid Clustering

Compaction (bin-packing)

- Over time, delta tables may contain several small files
- Especially likely when inserts are in small batches
- Compaction coalesces multiple small files
- Applied using the OPTIMIZE command

Compaction

ID	Name	State
1	Faye	WA
2	Ann	TX
3	Paul	WA
7	Lucy	WA



part-State=WA

p1f01.parquet
p1f02.parquet

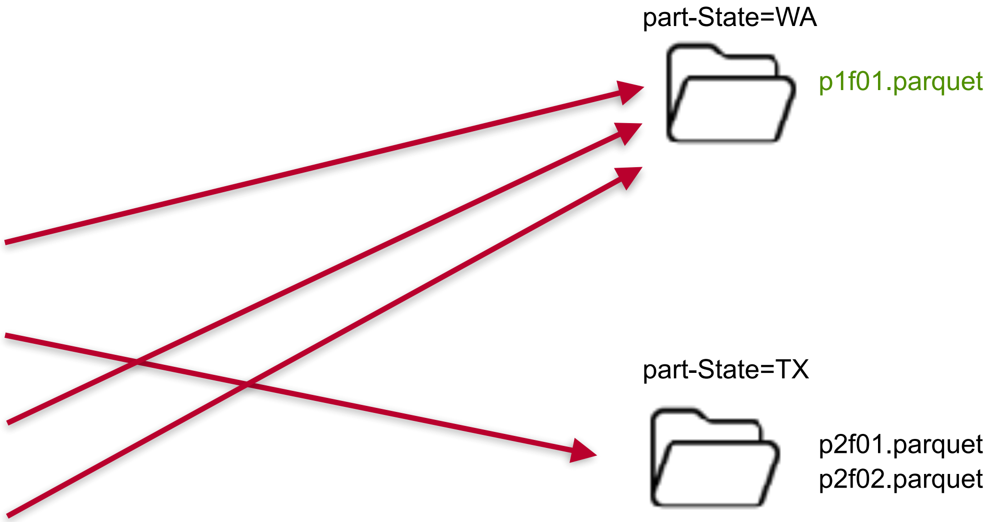
part-State=TX

p2f01.parquet
p2f02.parquet

All WA users may be split across multiple files in a partition

Compaction

ID	Name	State
1	Faye	WA
2	Ann	TX
3	Paul	WA
7	Lucy	WA



part-State=WA

p1f01.parquet

part-State=TX

p2f01.parquet
p2f02.parquet

Compaction can combine many small files into fewer large ones

Compaction (bin-packing)

- Bin-packing is idempotent (if run twice, second run has no effect)
- Tries to produce evenly balanced files with respect to the file size

File Layout Optimizations

Disk Caching

Data Skipping

Partitioning

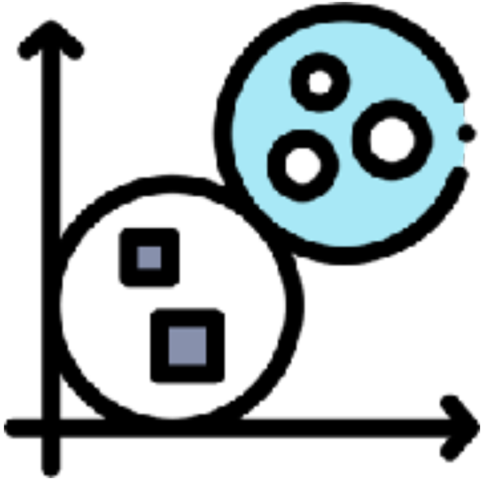
Compaction (Bin Packing)

Z-ordering

Liquid Clustering

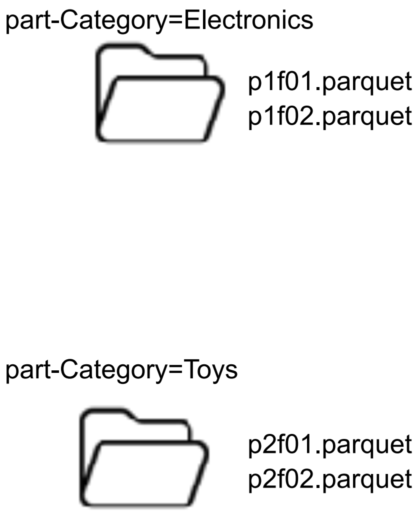
Z-Ordering

- Colocate related information in the same set of files
- Effectively clusters the data
- Minimizes file hits and maximizes data skipping
- Reduces number of files scanned and data transferred
- Applied using the OPTIMIZE command



Z-Ordering

ID	Product	Category	Brand
1	Phone	Electronics	XY
2	Teddy Bear	Toys	ABC
3	Tablet	Electronics	XY
7	Charger	Electronics	ZZ



part-Category=Electronics

p1f01.parquet
p1f02.parquet

part-Category=Toys

p2f01.parquet
p2f02.parquet

Data partitioned by category

Z-Ordering

ID	Product	Category	Brand
1	Phone	Electronics	XY
2	Teddy Bear	Toys	ABC
3	Tablet	Electronics	XY
7	Charger	Electronics	ZZ

part-Category=Electronics

p1f01.parquet
p1f02.parquet

part-Category=Toys

p2f01.parquet
p2f02.parquet

Products of the same brand may be in separate files

Z-Ordering

ID	Product	Category	Brand
1	Phone	Electronics	XY
2	Teddy Bear	Toys	ABC
3	Tablet	Electronics	XY
7	Charger	Electronics	ZZ

part-Category=Electronics

p1f01.parquet
p1f02.parquet

part-Category=Toys

p2f01.parquet
p2f02.parquet

Searches by category and brand may scan more files than needed

Z-Ordering

ID	Product	Category	Brand
1	Phone	Electronics	XY
2	Teddy Bear	Toys	ABC
3	Tablet	Electronics	XY
7	Charger	Electronics	ZZ

part-Category=Electronics

p1f01.parquet
p1f02.parquet

part-Category=Toys

p2f01.parquet
p2f02.parquet

Z-ordering will locate products of the same brand close together

Liquid Clustering

Disk Caching

Data Skipping

Partitioning

Compaction (Bin Packing)

Z-ordering

Liquid Clustering

Liquid Clustering

- Delta Lake liquid clustering replaces table partitioning and `ZORDER` to simplify data layout decisions and optimize query performance

- Tables often filtered by high cardinality columns.
- Tables with significant skew in data distribution.
- Tables that grow quickly and require maintenance and tuning effort.
- Tables with concurrent write requirements.
- Tables with access patterns that change over time.
- Tables where a typical partition key could leave the table with too many or too few partitions.

Poll 5

Which of the following optimizations organizes data so that related data is placed close together?

- Delta Caching
- Z-ordering
- Compaction
- Serialization

Poll 5

Which of the following optimizations organizes data so that related data is placed close together?

- Delta Caching
- Z-ordering
- Compaction
- Serialization

O'REILLY®

Hands-on Demos: Delta Optimizations