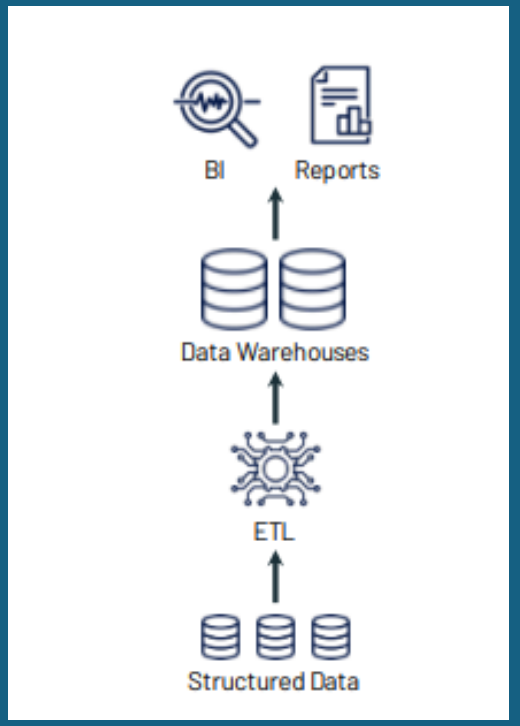
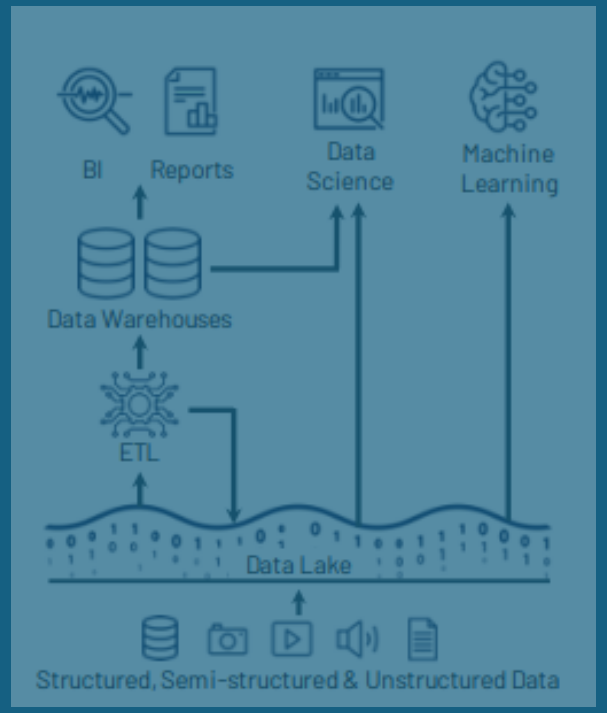
LAKEHOUSE WITH DELTA LAKE

PART 2: HOW DELTA LAKE SOLVES ALL YOUR DATA PROBLEMS ;)

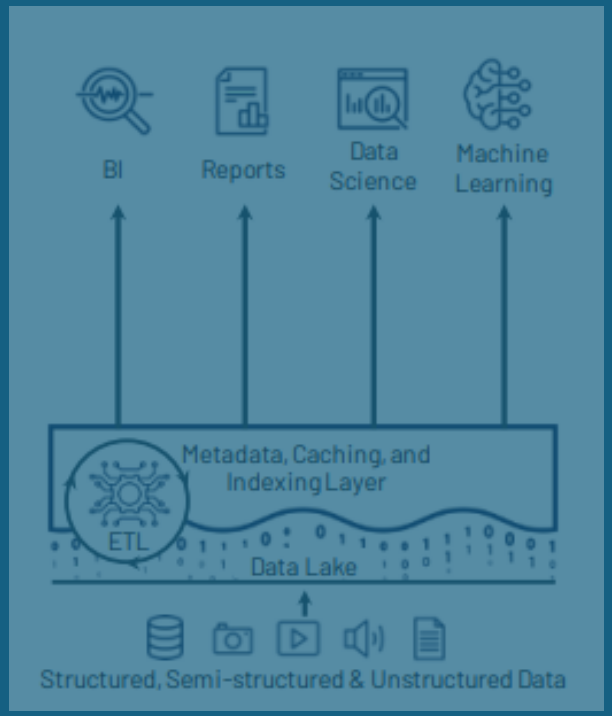
Data Warehouse



Two tier Architecture



Data Lakehouse

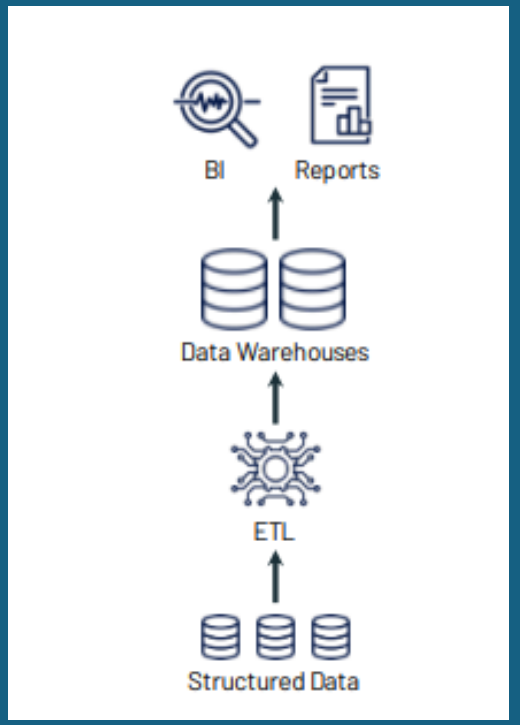


- DML** transactions and Analytical queries
- Central data store with central access control, governance and ACID*
- High costs, storage and compute coupling, missing scalability
- No unstructured data support,

*Atomicity, Consistency, Isolation, and Durability

** Data Manipulation Language: Insert, Update, Delete

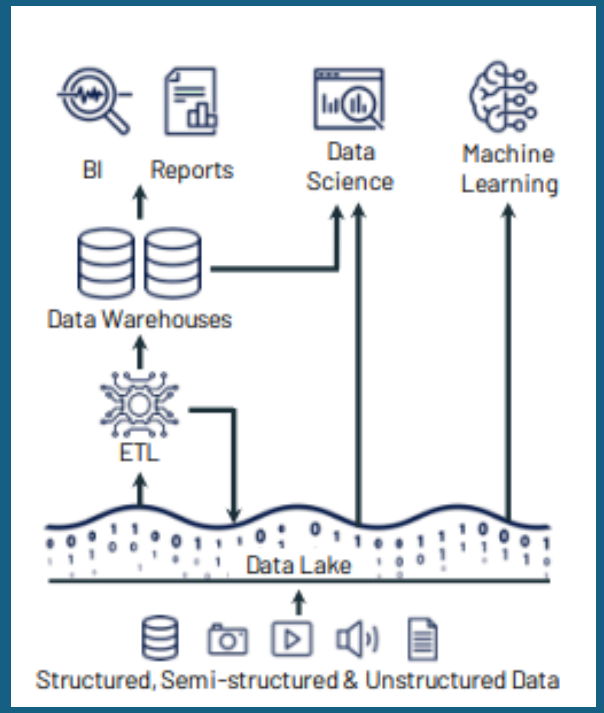
Data Warehouse



- DML** transactions and Analytical queries
- Central data store with central access control, governance and ACID*
- High costs, storage and compute coupling, missing scalability
- No unstructured data support,

*Atomicity, Consistency, Isolation, and Durability

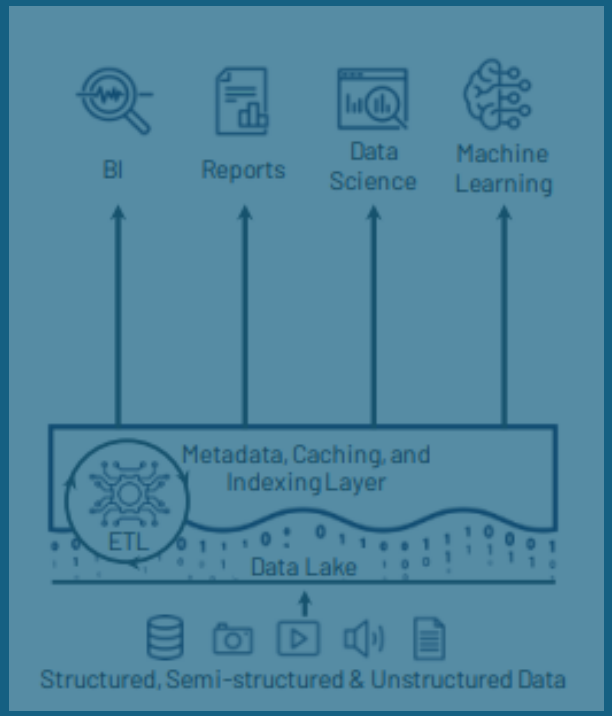
Two tier Architecture



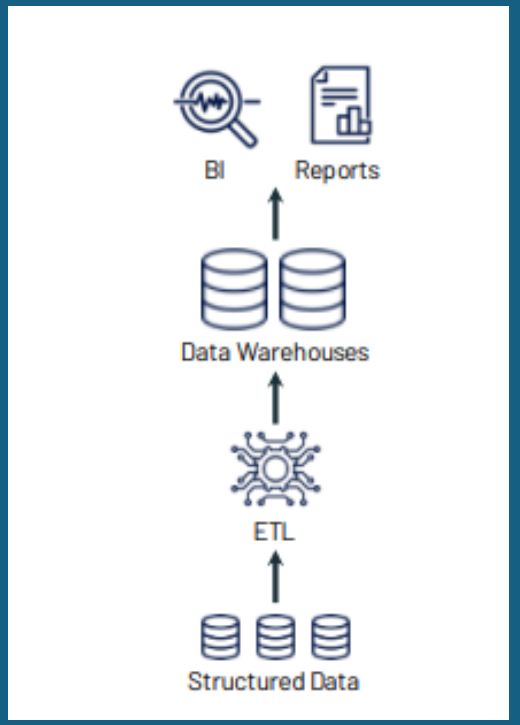
- Data saved in low-cost storage like HDFS and later on cloud like ADL2
- Subset of data moved to Data Warehouse
- Spark and Hadoop helped
- ML possible with Dataframe APIs consuming storage data
- Data replication for ETL and ML
- Issues: governance, out of sink, costs, swamps

** Data Manipulation Language: Insert, Update, Delete

Data Lakehouse



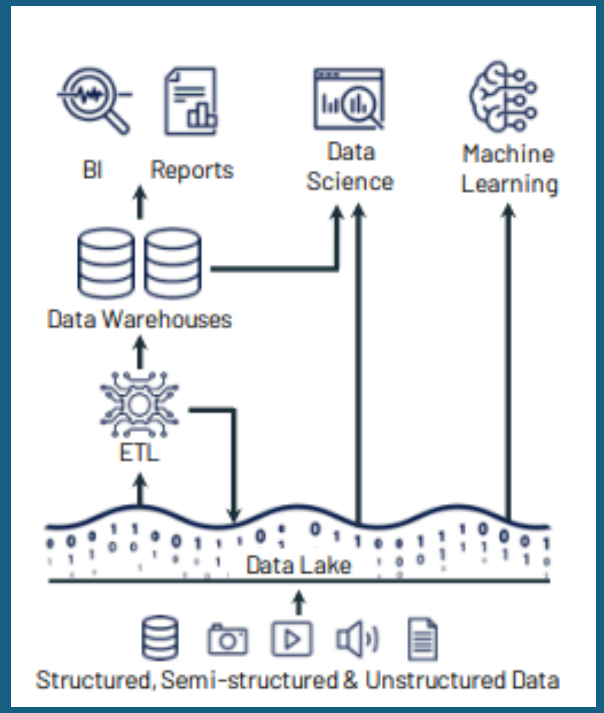
Data Warehouse



- DML** transactions and Analytical queries
- Central data store with central access control, governance and ACID*
- High costs, storage and compute coupling, missing scalability
- No unstructured data support,

*Atomicity, Consistency, Isolation, and Durability

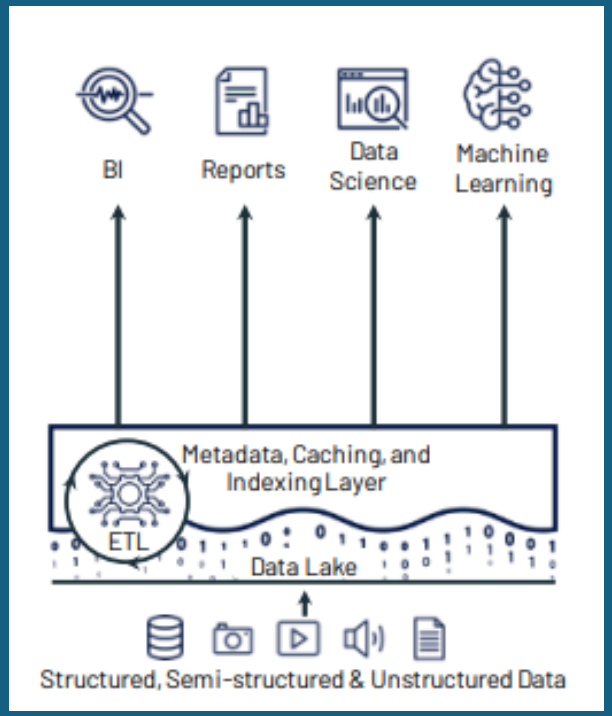
Two tier Architecture



- Data saved in low-cost storage like HDFS and later on cloud like ADL2
- Subset of data moved to Data Warehouse
- Spark and Hadoop helped
- ML possible with Dataframe APIs consuming storage data
- Data replication for ETL and ML
- Issues: governance, out of sink, costs, swamps

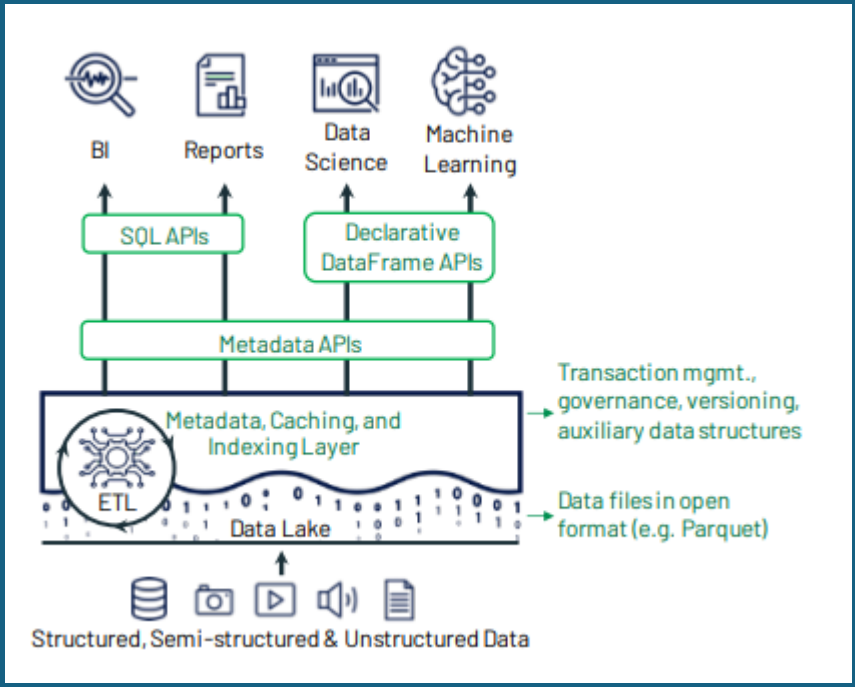
** Data Manipulation Language: Insert, Update, Delete

Data Lakehouse



- ETL and DML using Dataframe APIs
- BI Support via SQL and JDBC Support for Machine Learning
- Open Standards like Parquet
- Schema enforcement, Governance, Audit logs, ACID
- Indexing, Data Versioning
- Separate Compute and Storage

Key Components of the Lakehouse



Metadata layer:

- File format, transactional meta layer on top of parquet
- Schema enforcement and Governance incl Audit logs and Data Integrity (ACID)
- Versioning, Indexing, Transaction history
- Indexing and Clustering
- Maintain and leverage statistics saved in the meta data layer for file skipping

SQL API:

- Caching in SSDs and RAM as faster storage and partially decompress data
- Backed by Spark Engine as powerful engine, Spark SQL, Hive and open source jdbc

Machine Learning:

- Dataframe APIs like Pandas and Spark used for ML modules like XGBoost can leverage the saved data e.g. in parquet

Storage Frameworks serving as Meta Data Layer

Created by Databricks, available 2017

Created by Netflix, available 2017

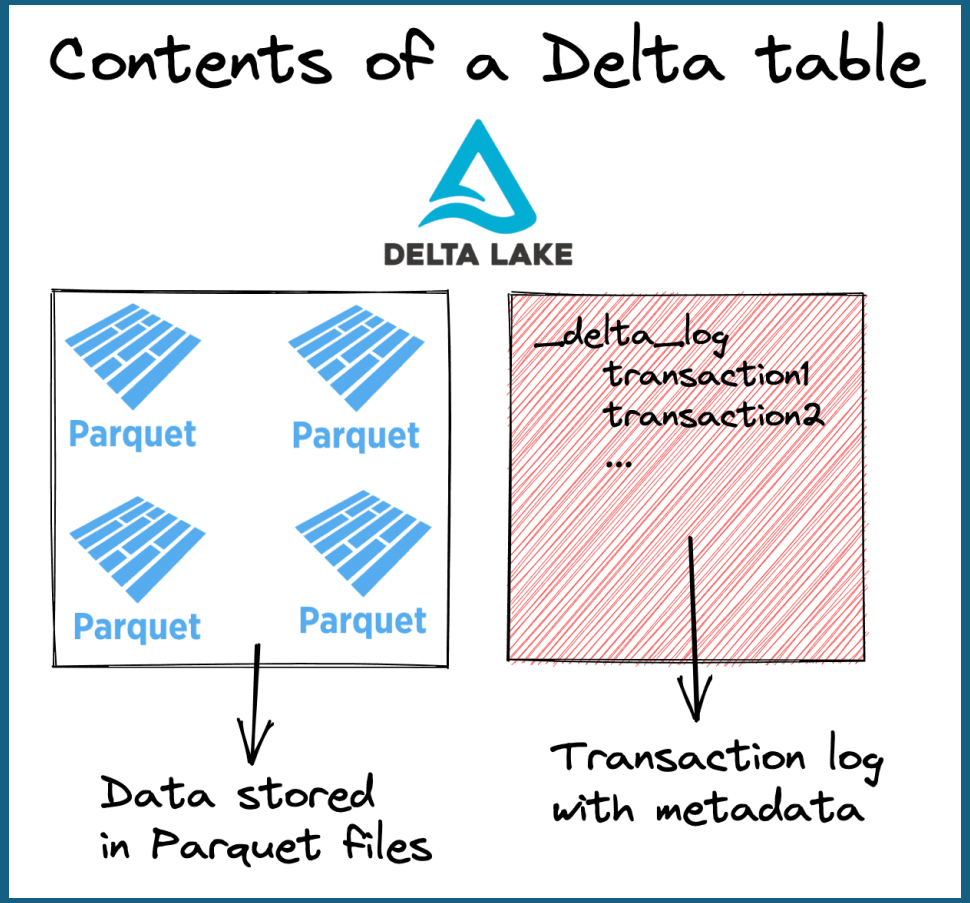
Created by Uber, available 2016

+

+ **Metadata**

Delta Lake

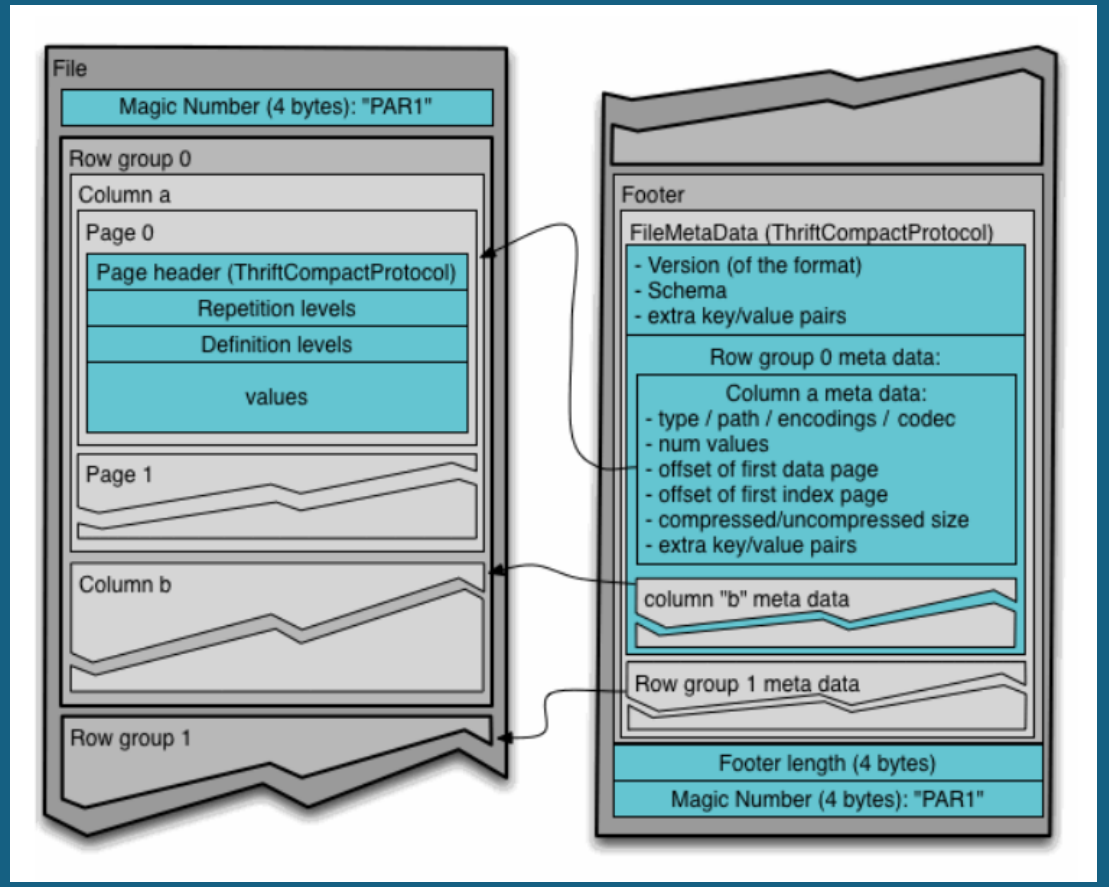
Created by Databricks,
available 2017



Delta Lake vs. Parquet Comparison | Delta Lake

- Launched 2013, developed by Cloudera and Twitter
- Allows complex data types
- Hybrid approach columnar but also row-level via so called row groups. Queries cane use predicate pushdown to load relevant rows and column filtering
- Compressible, Snappy as default with the aim of high speed
- Optimized columnar storage and efficient compression and encoding makes it also fast in writing data and saving significant amount of size
- Schema (column names, type and null value) and other meta data are saved in the footer. Thus, self-describing and efficient for serialisation
- Updates require a recreation as files are immutable
- Splitable for parallel reads

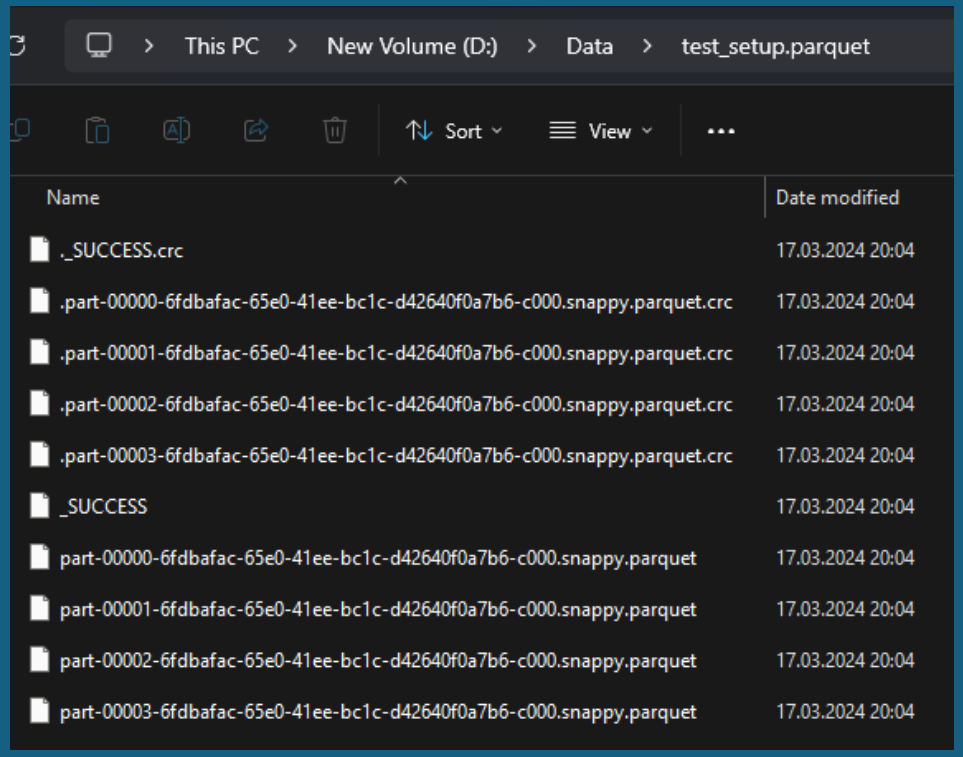
- Row Groups are a logical division on row level of a parquet defaulting to 128 MB
- Column part relates to column chunk of row groups
- Pages are invisible units where the encoding and compression happens
- Footer containing file metadata which can be used for predicate pushdown:
  - File level: num rows/ columns, schema
  - Row group: num rows/ columns
  - Column level: min, max, null count, distinct values, page indexes etc.

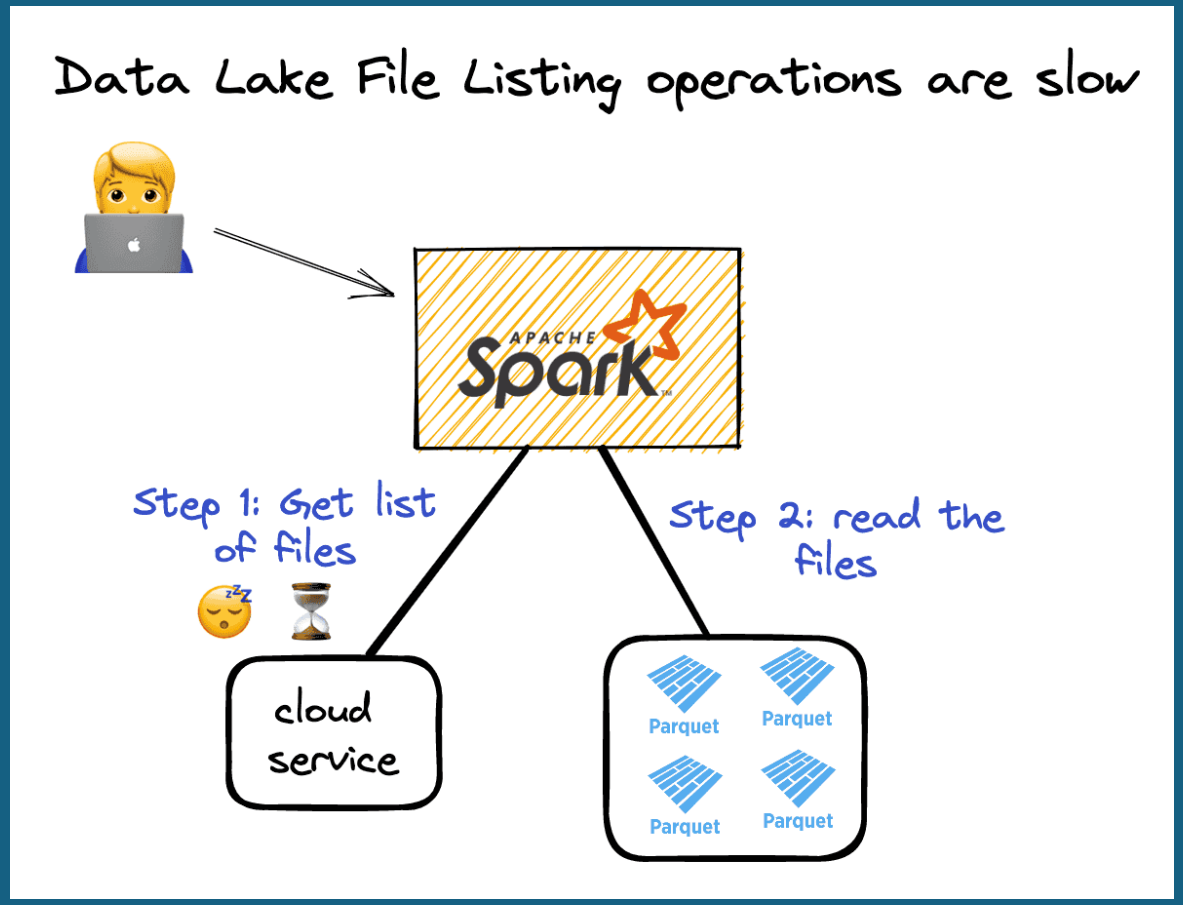


Parquet format: Documentation | Parquet (apache.org)
Parquet Page Indexes: Speeding Up SELECT Queries with Parquet Page Indexes - Cloudera Blog

Usually multiple parquet files with File Listing

```
data_folder.parquet/
    file1.parquet
    file2.parquet
    ...
    fileN.parquet
```





Delta Lake vs. Parquet Comparison | Delta Lake

- Columnar format allows loading relevant columns
- Saved schema reduces time for schema interference
- Columnar format allows high compression
- Statistics allow skipping row groups or pages which boosts loading time using predicate pushdown

- No ACID transaction
- Operations like Delete, Update, Merge
- Expensive footer reads to skip a whole file and collect statistics
- Rename columns, reorder, drop not possible
- File listing overhead
- …

Delta Lakes makes managing Parquet better

ACID Transactions

Protect your data with serializability, the strongest level of isolation

Scalable Metadata

Handle petabyte-scale tables with billions of partitions and files with ease

Time Travel

Access/revert to earlier versions of data for audits, rollbacks, or reproduce

Open Source

Community driven, open standards, open protocol, open discussions

Unified Batch/Streaming

Exactly once semantics ingestion to backfill to interactive queries

Schema Evolution / Enforcement

Prevent bad data from causing data corruption

Audit History

Delta Lake log all change details providing a fill audit trail

DML Operations

SQL, Scala/Java and Python APIs to merge, update and delete datasets

Home | Delta Lake

Delta Lakes makes managing Parquet better

ACID Transactions

Protect your data with serializability, the strongest level of isolation

- Databases & Data Warehouses support ACID to prevent errors in data
- Parquet does not support ACID which can lead to e.g. a half-written file during an append operation
- As Delta Lake supports ACID you will never be able to corrupt Delta

Unified Batch/Streaming

Exactly once semantics ingestion to backfill to interactive queries

Schema Evolution / Enforcement

Prevent bad data from causing data corruption

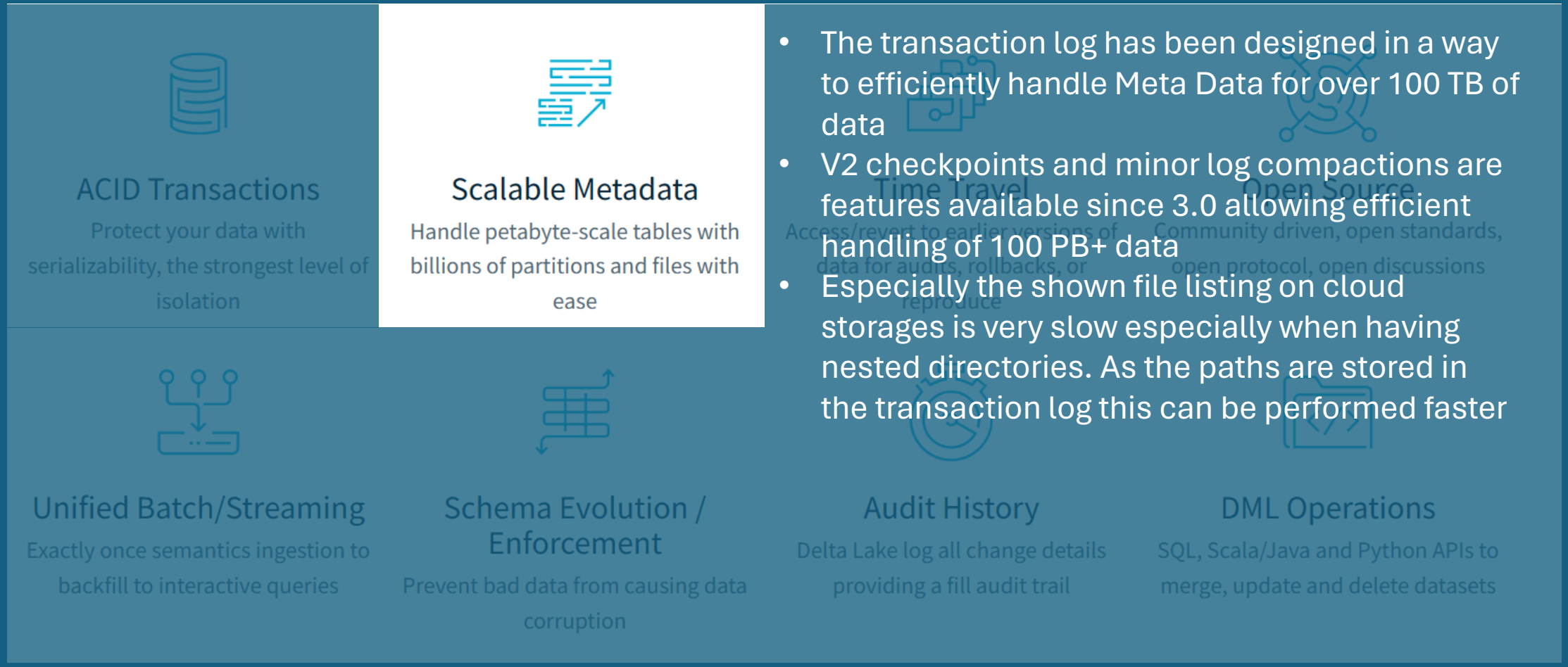
Audit History

Delta Lake log all change details providing a fill audit trail

DML Operations

SQL, Scala/Java and Python APIs to merge, update and delete datasets

Delta Lakes makes managing Parquet better



- The transaction log has been designed in a way to efficiently handle Meta Data for over 100 TB of data
- V2 checkpoints and minor log compactions are features available since 3.0 allowing efficient handling of 100 PB+ data
- Especially the shown file listing on cloud storages is very slow especially when having nested directories. As the paths are stored in the transaction log this can be performed faster

Delta Lakes makes managing Parquet better

- Deleting or overwriting data in Parquet meant a physical delete
- In Delta data is physically deleted using Vacuum
- Together with the transaction log it allows you to inspect and roll back to earlier versions of a table

Time Travel

Access/revert to earlier versions of data for audits, rollbacks, or reproduce

Open Source

Community driven, open standards, open protocol, open discussions

Unified Batch/Streaming

Exactly once semantics ingestion to backfill to interactive queries

Schema Evolution / Enforcement

Prevent bad data from causing data corruption

Audit History

Delta Lake log all change details providing a fill audit trail

DML Operations

SQL, Scala/Java and Python APIs to merge, update and delete datasets

Delta Lakes makes managing Parquet better

- Since Delta 2.0 in 2017 Delta Lake is fully open-source
- A fast-growing community around Delta has led to a lot of developments in the Delta Ecosystem

Open Source

Community driven, open standards, open protocol, open discussions

Unified Batch/Streaming

Exactly once semantics ingestion to backfill to interactive queries

Schema Evolution / Enforcement

Prevent bad data from causing data corruption

Audit History

Delta Lake log all change details providing a fill audit trail

DML Operations

SQL, Scala/Java and Python APIs to merge, update and delete datasets

Delta Lakes makes managing Parquet better

ACID Transactions

Protect your data with serializability, the strongest level of isolation

Scalable Metadata

Handle petabyte-scale tables with billions of partitions and files with ease

Time Travel

Access/revert to earlier versions of data for audits, rollbacks, or reproduce

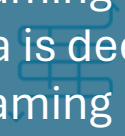
Open Source

Community driven, open standards, open protocol, open discussions

Unified Batch/Streaming

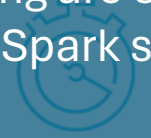
Exactly once semantics ingestion to backfill to interactive queries

- Streaming and Batch processing are supported
- Delta is deeply integrated into Spark structured Streaming



Schema Evolution / Enforcement

Prevent bad data from causing data corruption



Audit History

Delta Lake log all change details providing a fill audit trail

DML Operations

SQL, Scala/Java and Python APIs to merge, update and delete datasets

Delta Lakes makes managing Parquet better

ACID Transactions

Protect your data with serializability, the strongest level of isolation

Scalable Metadata

Handle petabyte-scale tables with billions of partitions and files with ease

Time Travel

Access/revert to earlier versions of data for audits, rollbacks, or reproduce

Open Source

Community driven, open standards, open protocol, open discussions

Unified Batch/Streaming

Exactly once semantics ingestion to backfill to interactive queries

Schema Evolution / Enforcement

Prevent bad data from causing data corruption

- Another Data Warehouse feature enforcing the schema in your table to avoid corruption
- Schema evolution based on a logical layer on top of parquet:
  - Rename, add, drop, re-order columns
  - Schema changes require re-write
  - Type widening in Delta 4.0 will also allow some type changes without re-write

Delta Lakes makes managing Parquet better

ACID Transactions

Protect your data with serializability, the strongest level of isolation

Scalable Metadata

Handle petabyte-scale tables with billions of partitions and files with ease

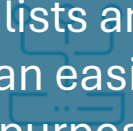
Time Travel

Access/revert to earlier versions of data for audits, rollbacks, or reproduce

Open Source

Community driven, open standards, open protocol, open discussions

- Delta lists any operation on the table
- You can easily query the transaction log for audit purposes



Unified Batch/Streaming

Exactly once semantics ingestion to backfill to interactive queries

Schema Evolution / Enforcement

Prevent bad data from causing data corruption

Audit History

Delta Lake log all change details providing a fill audit trail

DML Operations

SQL, Scala/Java and Python APIs to merge, update and delete datasets

Delta Lakes makes managing Parquet better

ACID Transactions

Protect your data with serializability, the strongest level of isolation

Scalable Metadata

Handle petabyte-scale tables with billions of partitions and files with ease

Time Travel

Access/revert to earlier versions of data for audits, rollbacks, or reproduce

Open Source

Community driven, open standards, open protocol, open discussions

Unified Batch/Streaming

Exactly once semantics ingestion to backfill to interactive queries

- Despite the immutability of parquet files delta supports efficient ways to re-write the parquet files in the background when executing delete, update and merge
- Throughout Delta 2.4, 3.0 and 3.1 the added deletion vector features brought the performance even to a next level

DML Operations

SQL, Scala/Java and Python APIs to merge, update and delete datasets

Low Level features

File Skipping

Column pruning

Predicate pushdown

Optimize (bin packing)

Indexing

Home | Delta Lake

Low Level features

File Skipping

- Delta stores statistics about the underlying parquet files in the transaction logs
- This allows skipping whole parquet files without even load the parquet meta data which can be slow

Column pruning

Predicate pushdown

Optimize (bin packing)

Indexing

Low Level features

File Skipping

Column pruning

Predicate pushdown

Optimize (bin packing)

Indexing

- As with Parquet we can also prune columns
- Loading less data is always faster
- This is a Parquet only feature and supported by design

Low Level features

File Skipping

- Parquet is saving min/max statistics in the footer which allows skipping non needed row groups or pages
- With file skipping we can speed predicate push down even further. As we have not only one but multiple parquet files this is big improvement as only a subset of parquet file metadata need to be read

Column pruning

Predicate pushdown

Optimize (bin packing)

Indexing

Low Level features

File Skipping

Column pruning

Predicate pushdown

- Spark can’t handle a lot of small files very well due to the generated overhead
- What you want to achieve is to compact the files into bigger chunks
- The optimize command is doing this for you in a secure way without interfering with new data coming in while the compaction is running

Optimize (bin packing)

Indexing

Low Level features

File Skipping

Column pruning

Predicate pushdown

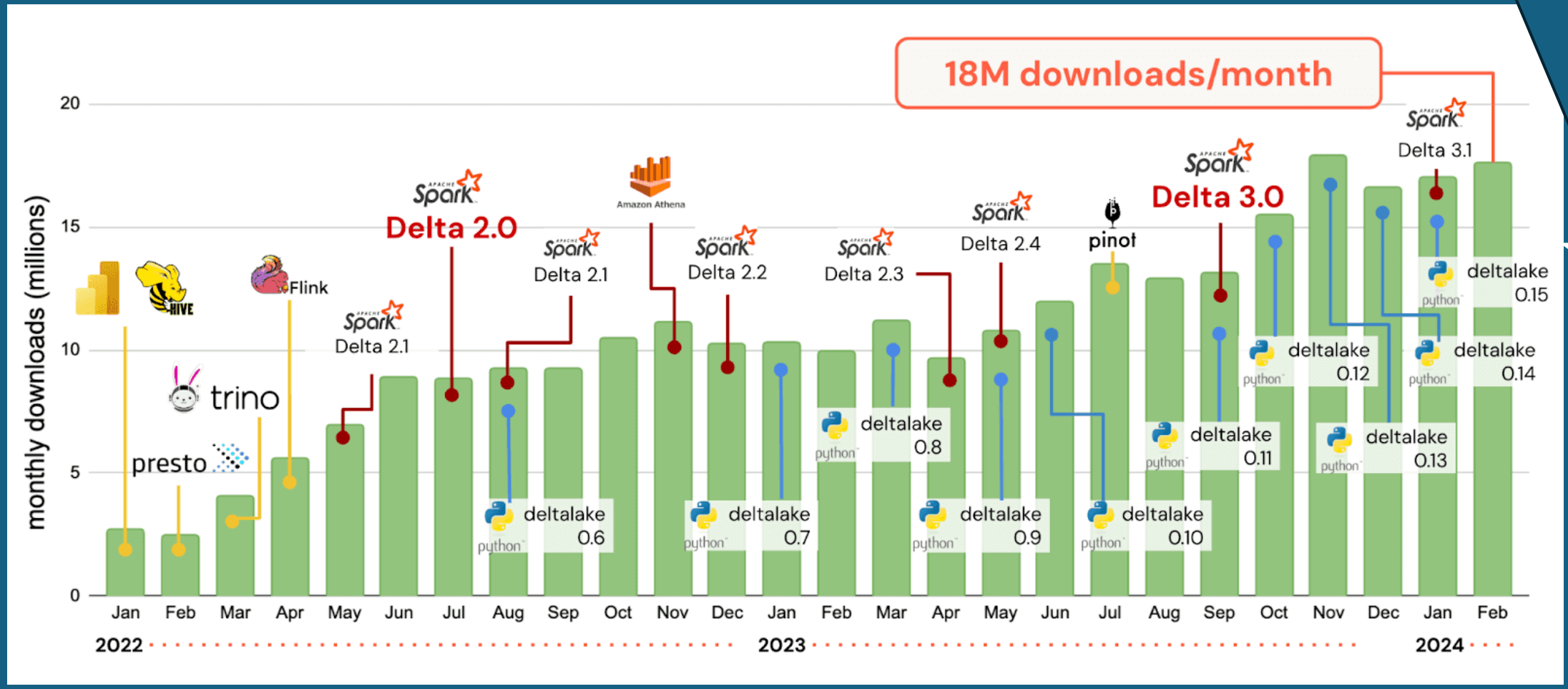
- File skipping becomes especially efficient if similar data is in the same parquet file
- That's what Delta is doing using Z-Order or since Delta 3.1 Liquid Clustering

Optimize (bin packing)

Indexing

Delta Lake Development

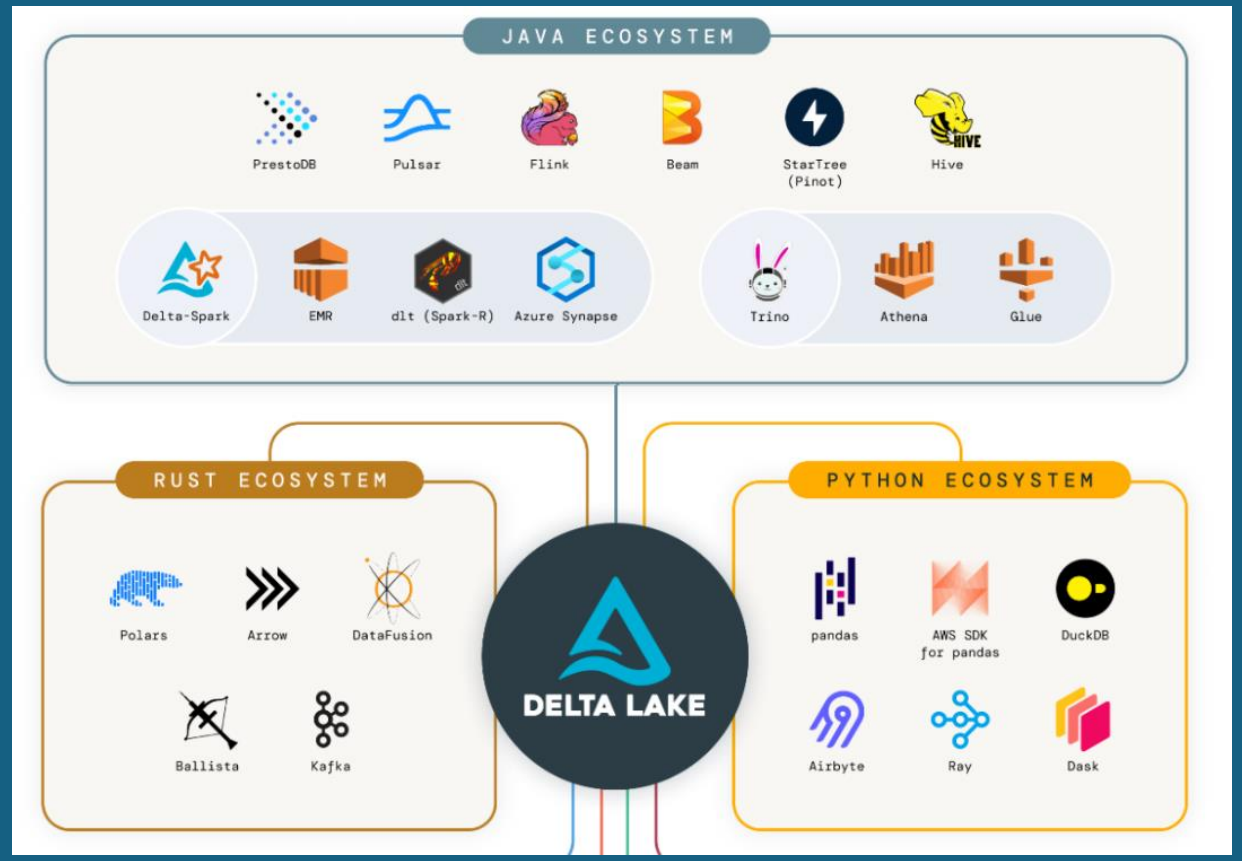
May 2024:
Delta Spark 3.2



Delta Lake - State of the Project - Part 1 | Delta Lake

The Delta Lake Ecosystem

Delta Lake Connectors & Kernel



Delta Sharing Protocol & Server

Delta Uniform

Home | Delta Lake

Summary

- The Lakehouse combines the best of Data Warehouses and Data Lakes from one single place
- Delta Lake provides mature features to maintain data efficiently for ETL, Machine Learning and Reporting on a Cloud Storage
- Delta Lakes community and eco system is significantly growing