Large-Scale Data Management and Distributed Systems

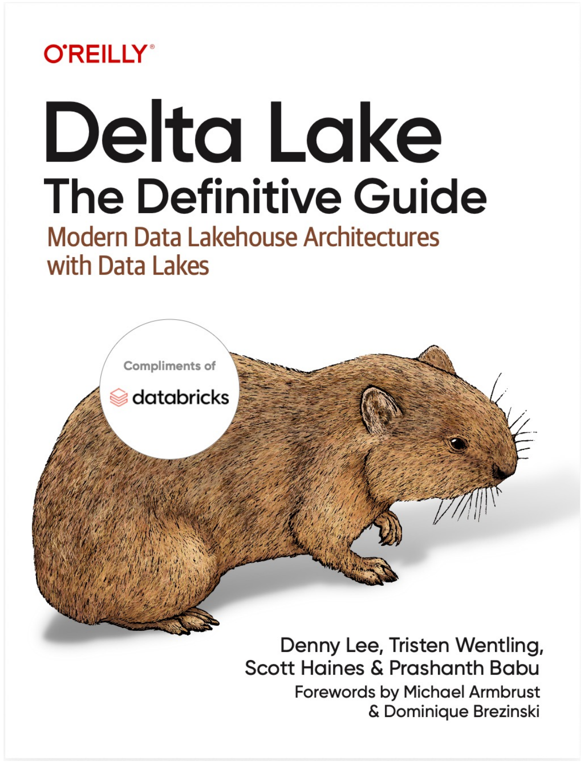
V. Delta Lakehouse

Vania Marangozova

Vania.Marangozova@imag.fr

2025-2026

References


O'REILLY

Delta Lake
The Definitive Guide
Modern Data Lakehouse Architectures with Data Lakes

Compliments of databricks

Denny Lee, Tristen Wentling, Scott Haines & Prashanth Babu
Forewords by Michael Armbrust & Dominique Brezinski

https://delta.io/pdfs/dldg_databricks.pdf

https://docs.databricks.com/aws/en/delta

White Paper

Zaharia, Matei A. et al. "Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics." Conference on Innovative Data Systems Research (2021).

Lakehouse

- Developed to address the limitations of traditional data lakes and data warehouses
- Provides ACID (atomicity, consistency, isolation, and durability) transactions
- Unifies various data analytics tasks, such as batch and streaming workloads, machine learning, and SQL

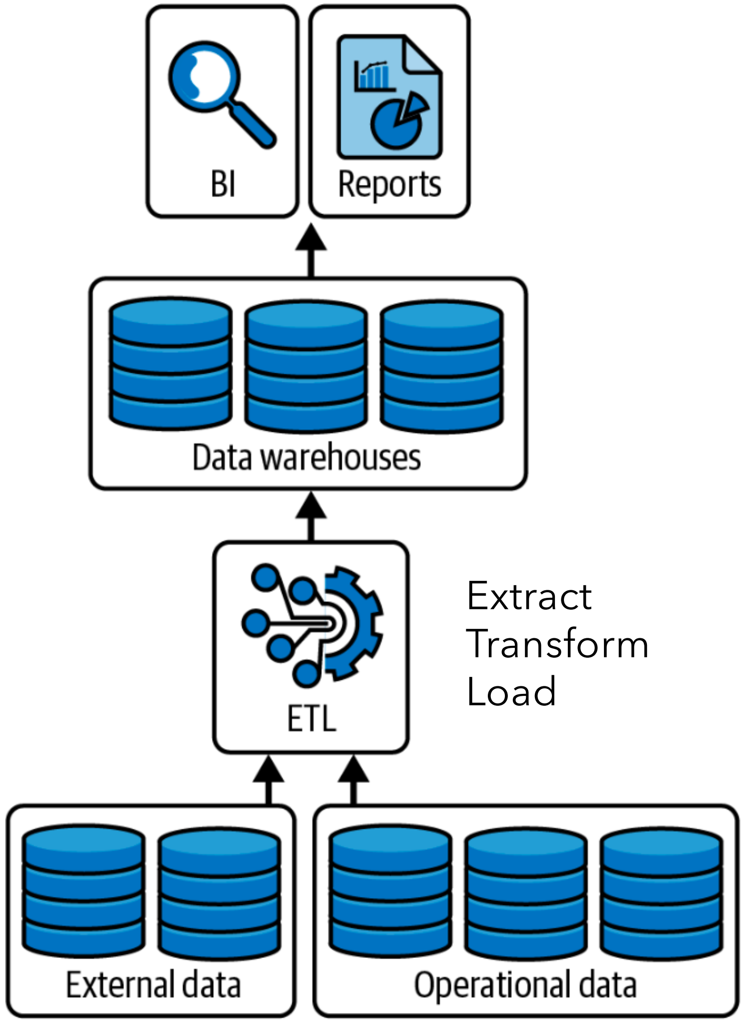
DELTA LAKE ICEBERG Apache Iceberg™ Apache hudi

Data Warehouses

- Aggregate and process structured data
  - Relational databases
  - ACID

☺ Management and optimization functions, **Robustness**

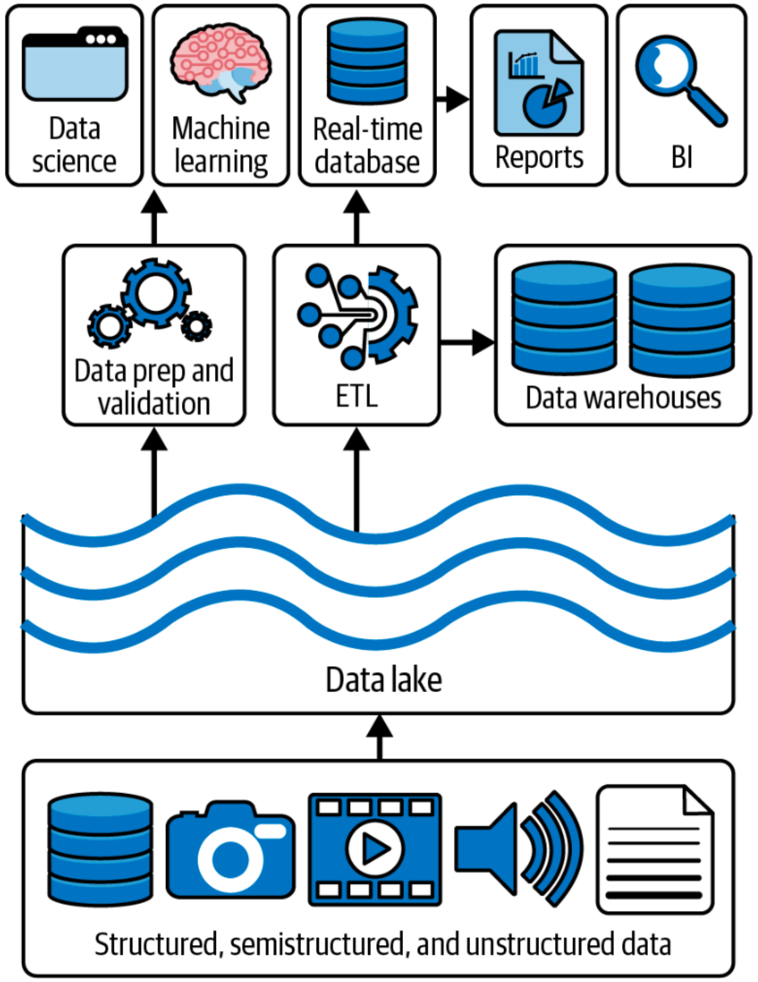
☹ Hard to scale and respond to the Big Data velocity needs



Extract
Transform
Load

Data Lakes

- **Big low-cost** storage repositories
  - Big Data/ Internet scale
- Raw format
- File-based, running on cluster of machines (distributed)
- Unreliable, BASE model
  - Basically available (replication of data)
  - Soft-state (data values may change)
  - Eventually consistent

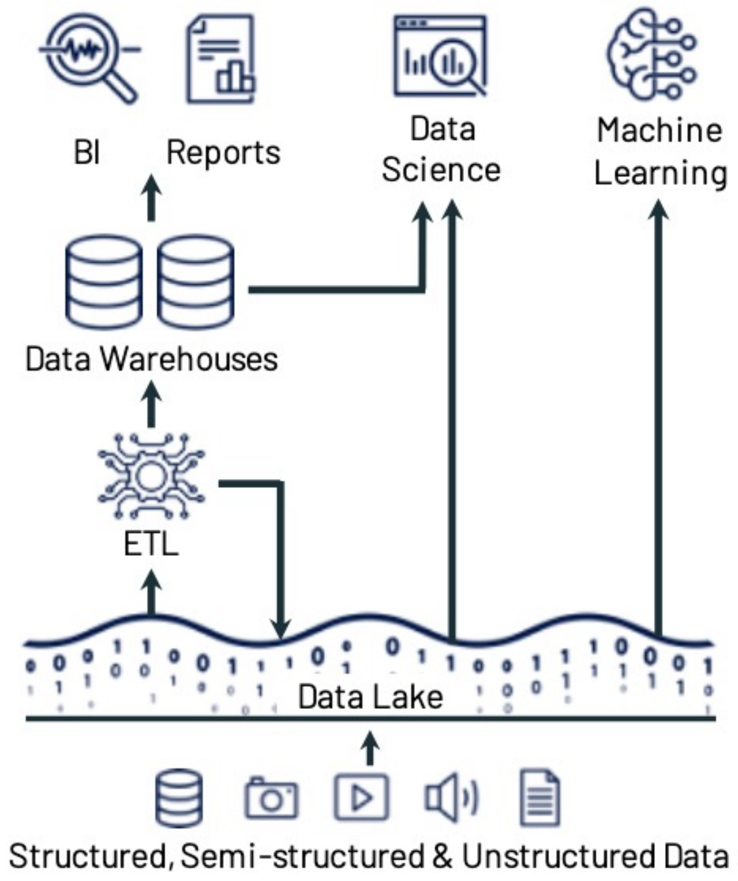


https://aws.amazon.com/fr/compare/the-difference-between-acid-and-base-database/

The Hybrid Solution
(before lakehouses)

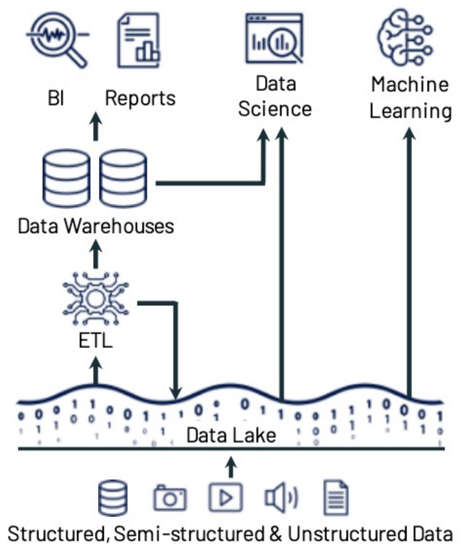
- Predominant (2021 white paper)

- Raw data in the data lake
- ETL to put data in the warehouse
- Analyses on both (lake & warehouse)

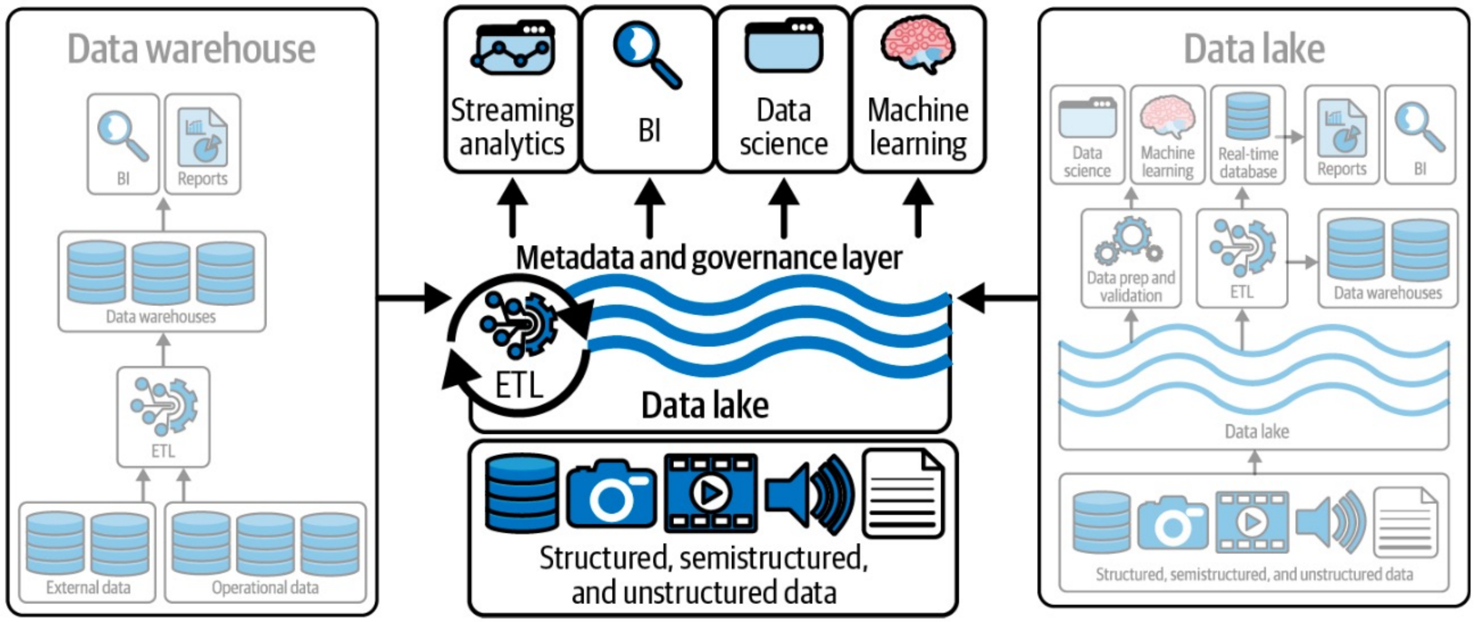


Hybrid Architecture Problems

- Reliability
  - keeping data consistent between data lakes and warehouses is difficult
- Data staleness.
  - The data in the warehouse is stale compared to that of the data lake
- Limited support for advanced analytics.
  - None of the leading ML systems, such as TensorFlow, PyTorch and XGBoost, work well on top of warehouses
- Total cost of ownership.
  - Apart from paying for continuous ETL, users pay double the storage cost

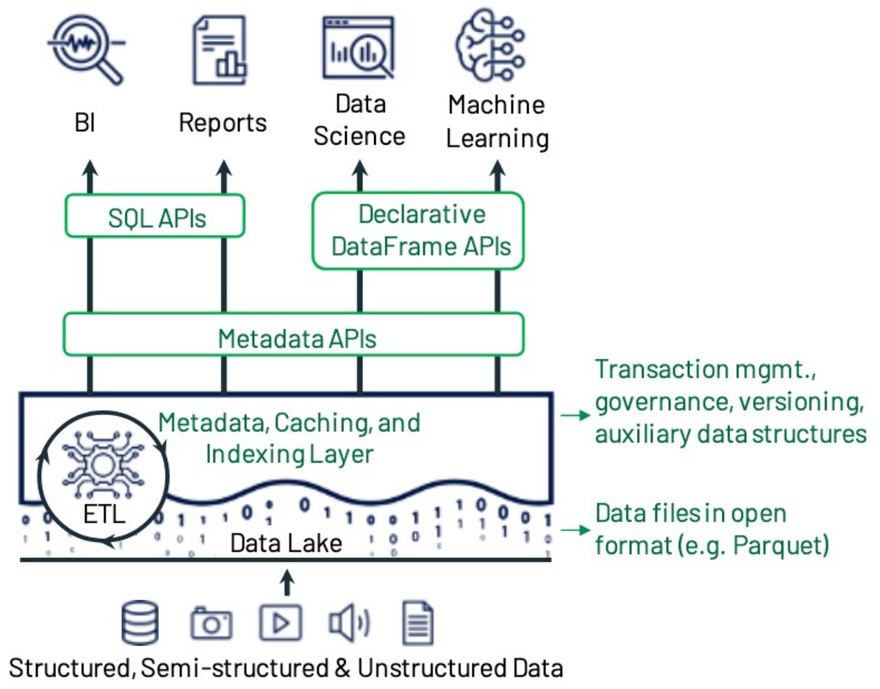


Lakehouse



Data warehouse
BI
Reports
Data warehouses
ETL
External data
Operational data
Streaming analytics
BI
Data science
Machine learning
Metadata and governance layer
ETL
Data lake
Structured, semistructured, and unstructured data
Data lake
Data science
Machine learning
Real-time database
Reports
BI
Data prep and validation
ETL
Data warehouses
Data lake
Structured, semistructured, and unstructured data

The Delta Lake Ideas

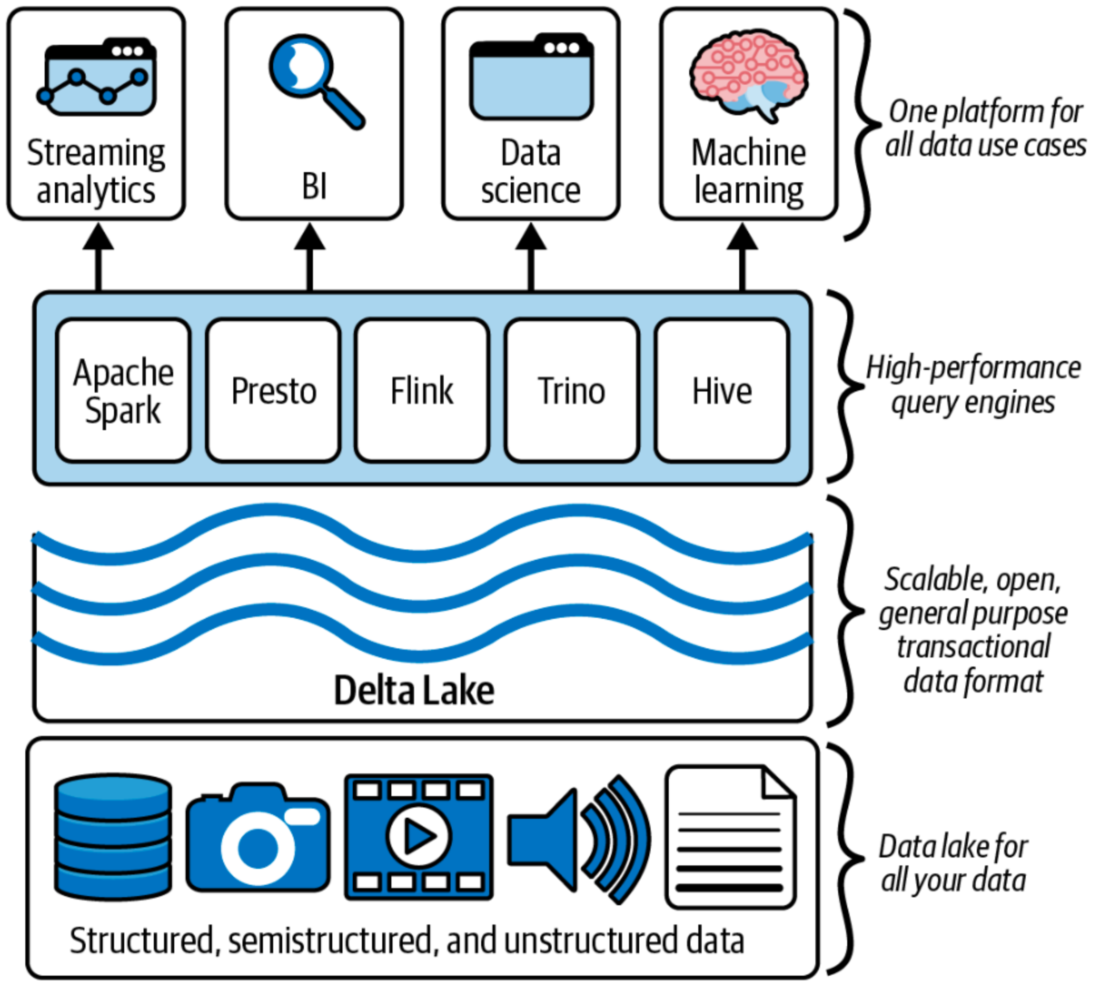


BI
Reports
Data Science
Machine Learning
SQL APIs
Declarative DataFrame APIs
Metadata APIs
Metadata, Caching, and Indexing Layer
Transaction mgmt., governance, versioning, auxiliary data structures
ETL
Data Lake
Data files in open format (e.g. Parquet)
Structured, Semi-structured & Unstructured Data

- Low-cost data storage, open format
- Meta-data transactional layer
  - which objects are part of tables
- SQL performance
  - caching
  - auxiliary data structures such as indexes and statistics
  - data layout optimizations
- ML processing
  - declarative dataframe API

Delta Lake

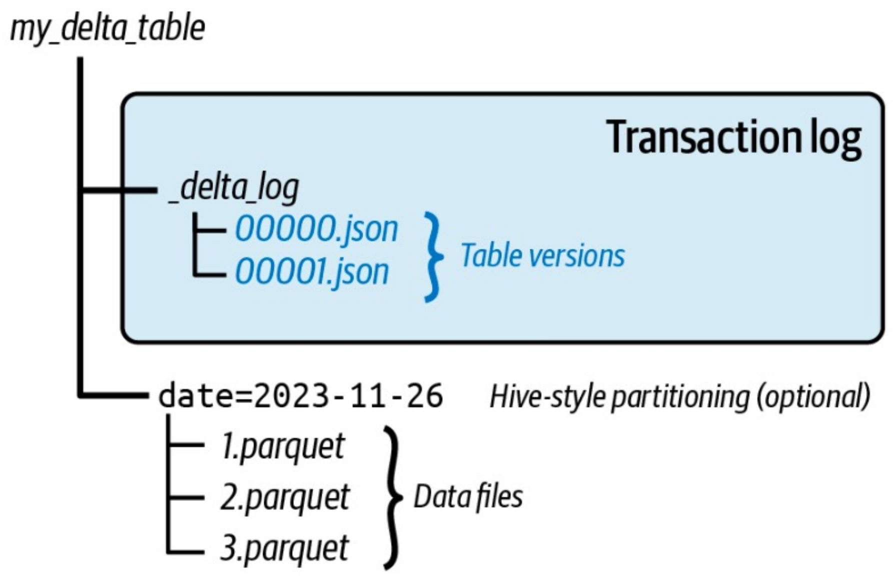
- ACID transactions
- Scalable metadata handling
- Unification of streaming and batch data processing



Streaming analytics
BI
Data science
Machine learning
One platform for all data use cases
Apache Spark
Presto
Flink
Trino
Hive
High-performance query engines
Delta Lake
Scalable, open, general purpose transactional data format
Structured, semistructured, and unstructured data
Data lake for all your data

Delta Lake Table (1)

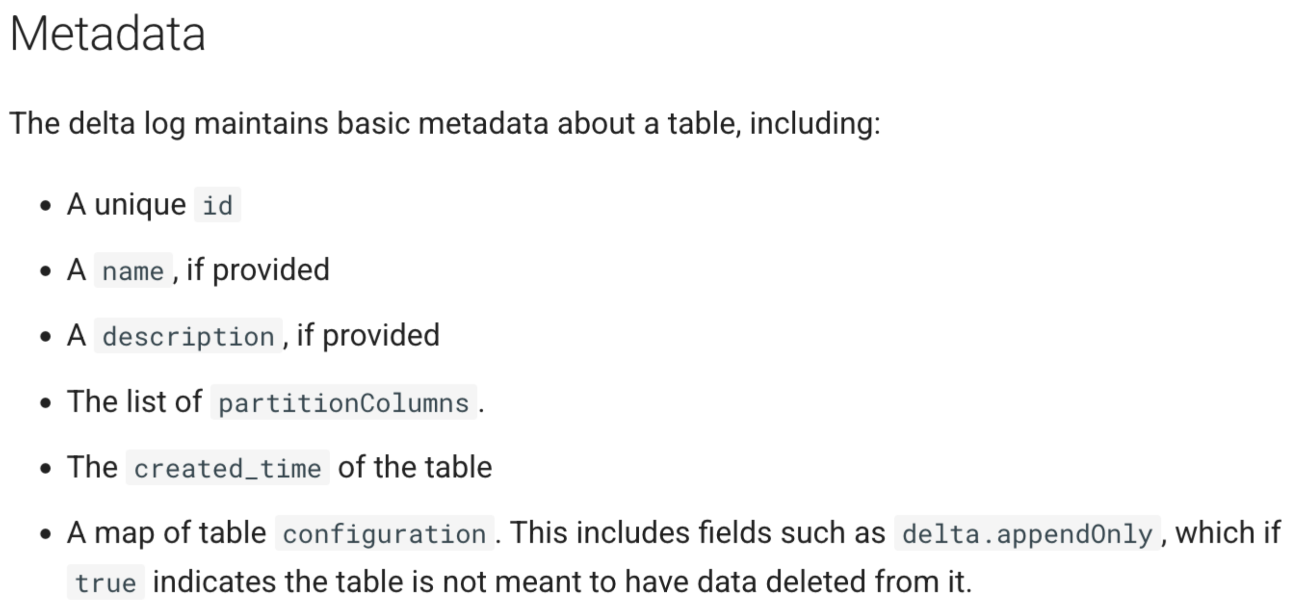
- Data files : Parquet format
- Transaction log
  - Ordered log of all transactions
  - A transaction is encoded in a json file



https://delta-io.github.io/delta-rs/usage/examining-table/#metadata

Delta Lake Table (2)

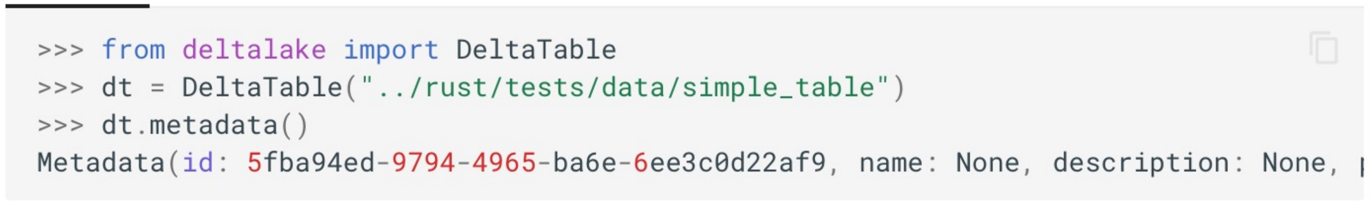
- Metadata
  - table’s schema, partitioning, configuration settings



Metadata

The delta log maintains basic metadata about a table, including:

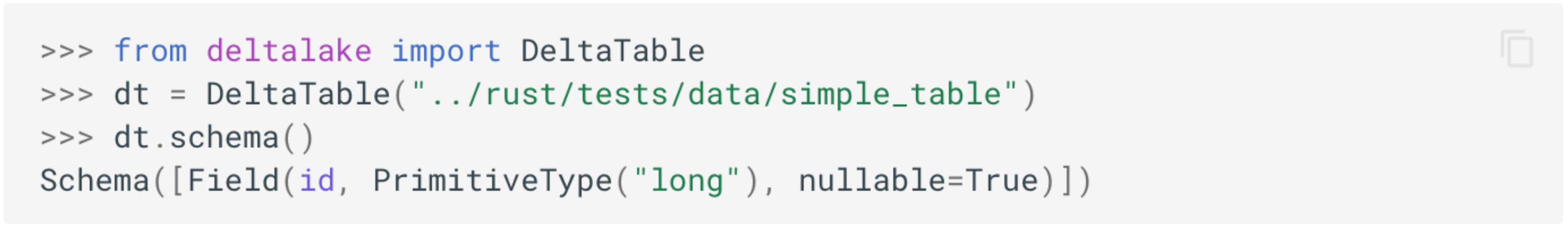
- A unique `id`
- A `name`, if provided
- A `description`, if provided
- The list of `partitionColumns`.
- The `created_time` of the table
- A map of table `configuration`. This includes fields such as `delta.appendOnly`, which if `true` indicates the table is not meant to have data deleted from it.



```
>>> from deltalake import DeltaTable
>>> dt = DeltaTable("../rust/tests/data/simple_table")
>>> dt.metadata()
Metadata(id: 5fba94ed-9794-4965-ba6e-6ee3c0d22af9, name: None, description: None,
```

Delta Lake Table (3)

- Schema
  - the data's structure, columns, data types, etc.

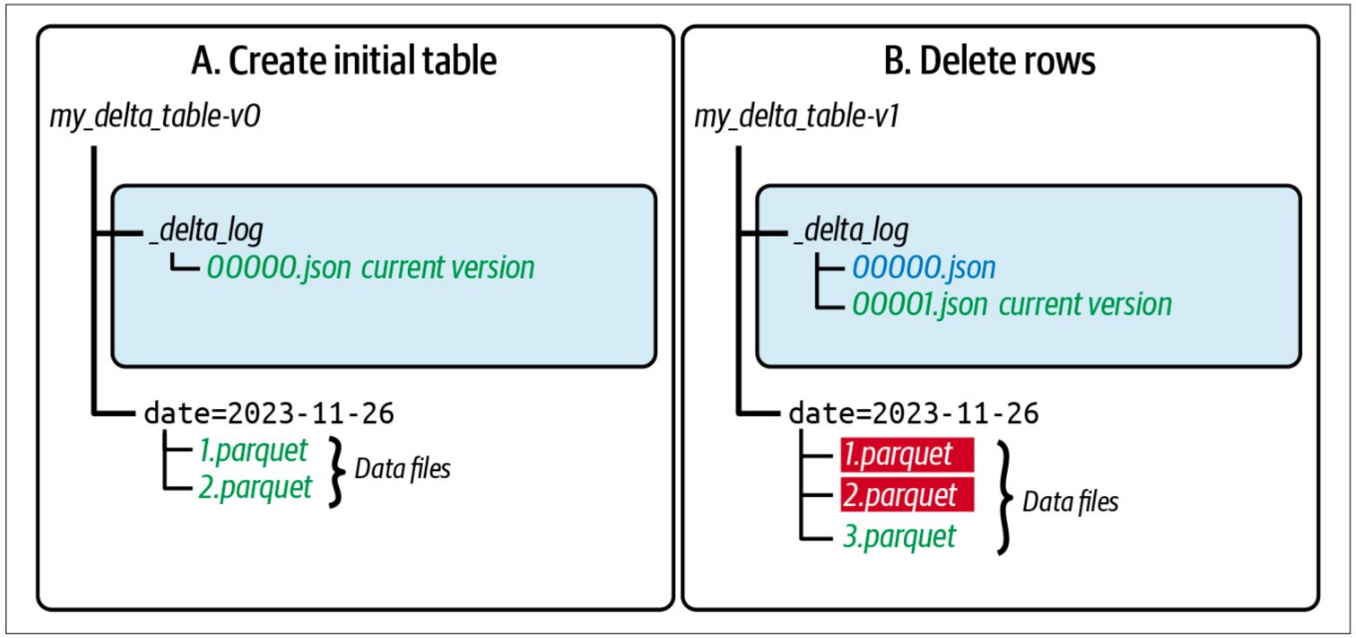


```
>>> from deltalake import DeltaTable
>>> dt = DeltaTable("../rust/tests/data/simple_table")
>>> dt.schema()
Schema([Field(id, PrimitiveType("long"), nullable=True)])
```

- Checkpoints
  - Every 10 transactions
  - For faster recovery

Delta Transaction Protocol

- Any client who wants to read or write to a Delta table must first query the transaction log

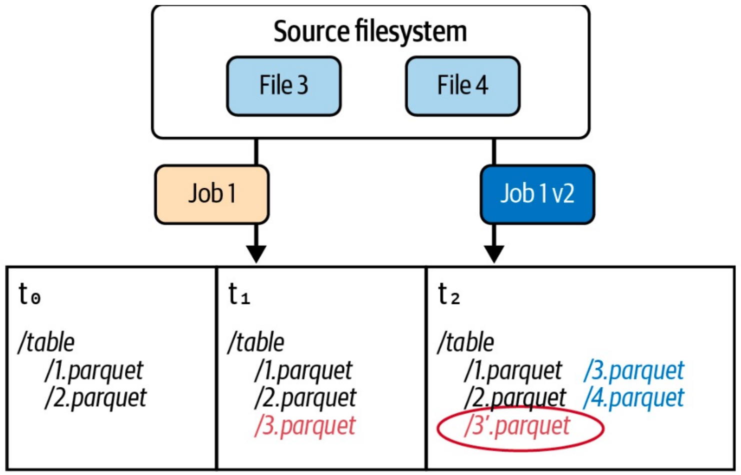


Multiversion Concurrency Control

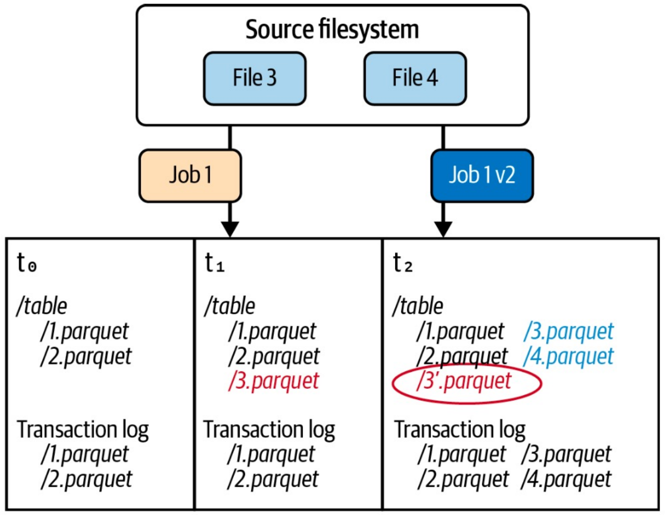
- Observation for **deletes**
  It is faster to create a new file comprising the unaffected rows rather than modifying the existing Parquet file(s)
  - allows data to be safely read and updated concurrently
  - Time travel = multiple versions available

Data and Metadata

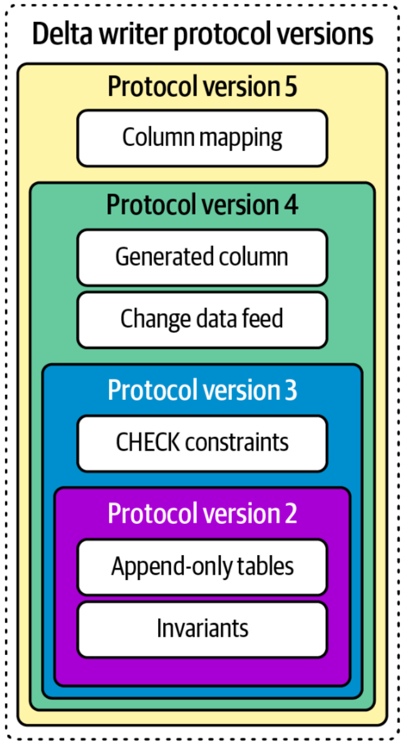
- When modifying tables, some incomplete files may be created



- But the transaction log reflects only committed transactions



Built-In Evolutivity



- Invariants: properties to be enforced on column data
- Append-only: no changes on tables
- CHECK: constraints to be enforced
- Change data feed: register raw data changes
- Generated column: generate additional columns with user-specified treatments
- Column mapping: support different column names

Adoptions

Comcast

- Petabytes of data
- Compute utilization from 640VMs to 64VMs
- 84 to 3 jobs

Apple's information security team

- 300 billion events per day
- writing hundreds of terabytes of data daily