Decoding Data Lakehouse: A Technical Breakdown

Big Data Warsaw, Dipankar Mazumdar

Speaker Bio

Dipankar Mazumdar - Staff Developer Advocate

Open Source Contributor: Apache Hudi, Iceberg, XTable, Arrow

Prev Work: BI, ML, Data Architecture

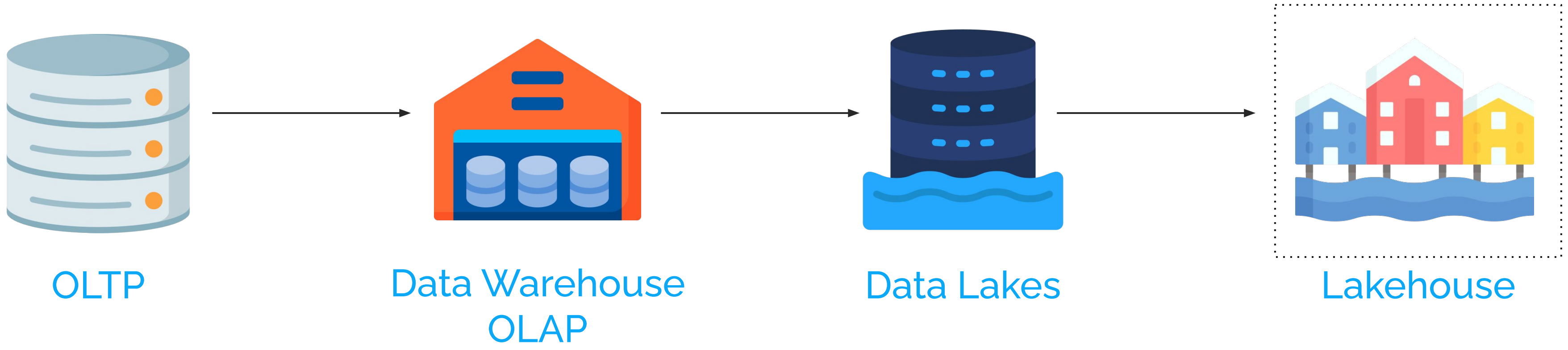
in/dipankarmazumdar/

@dipankartnt

Agenda

- Evolution of Data Architecture
- Research - Hypothesis
- Data Warehouse
- Requirements of Data Warehousing
- Data Lakehouse & Open Table Formats

Evolution of Data Architecture

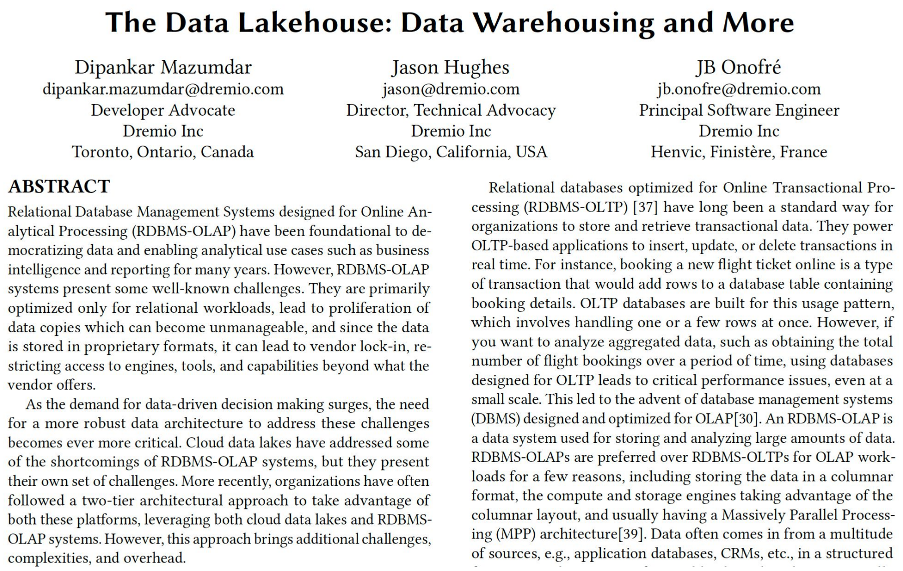


Store, Retrieve Data
Centralized & Reliable Data Platform
Democratize data

Hypothesis

1. Lakehouse - more than a marketing term
2. Data Warehouse means different things
3. Data Lakehouse = Data Warehouse + Lake + (Additional Values)
4. Advantages with a Open Lakehouse

Research Paper



The Data Lakehouse: Data Warehousing and More

Dipankar Mazumdar
dipankar.mazumdar@dremio.com
Developer Advocate
Dremio Inc
Toronto, Ontario, Canada

Jason Hughes
jason@dremio.com
Director, Technical Advocacy
Dremio Inc
San Diego, California, USA

JB Onofré
jb.onofre@dremio.com
Principal Software Engineer
Dremio Inc
Henvic, Finistère, France

ABSTRACT

Relational Database Management Systems designed for Online Analytical Processing (RDBMS-OLAP) have been foundational to democratizing data and enabling analytical use cases such as business intelligence and reporting for many years. However, RDBMS-OLAP systems present some well-known challenges. They are primarily optimized only for relational workloads, lead to proliferation of data copies which can become unmanageable, and since the data is stored in proprietary formats, it can lead to vendor lock-in, restricting access to engines, tools, and capabilities beyond what the vendor offers.

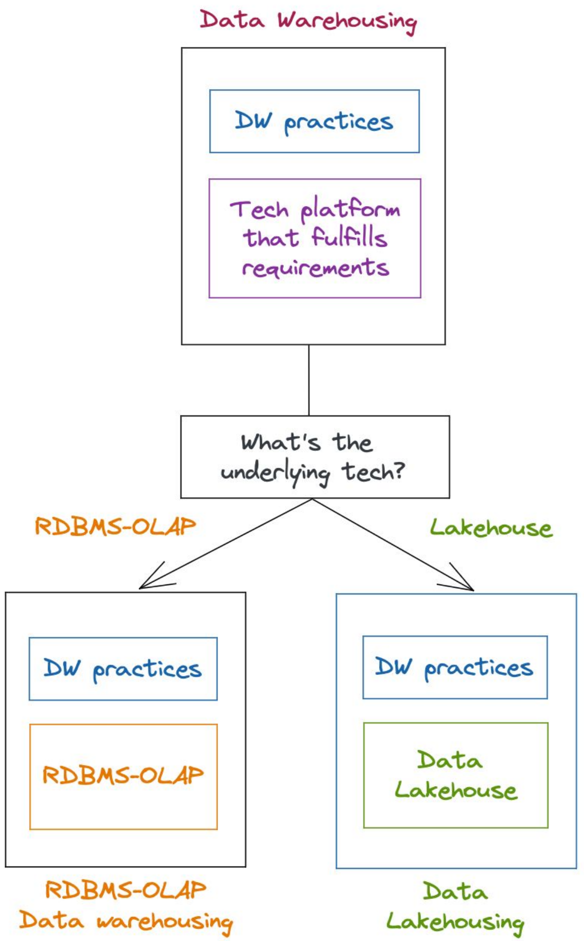
As the demand for data-driven decision making surges, the need for a more robust data architecture to address these challenges becomes ever more critical. Cloud data lakes have addressed some of the shortcomings of RDBMS-OLAP systems, but they present their own set of challenges. More recently, organizations have often followed a two-tier architectural approach to take advantage of both these platforms, leveraging both cloud data lakes and RDBMS-OLAP systems. However, this approach brings additional challenges, complexities, and overhead.

Relational databases optimized for Online Transactional Processing (RDBMS-OLTP) [37] have long been a standard way for organizations to store and retrieve transactional data. They power OLTP-based applications to insert, update, or delete transactions in real time. For instance, booking a new flight ticket online is a type of transaction that would add rows to a database table containing booking details. OLTP databases are built for this usage pattern, which involves handling one or a few rows at once. However, if you want to analyze aggregated data, such as obtaining the total number of flight bookings over a period of time, using databases designed for OLTP leads to critical performance issues, even at a small scale. This led to the advent of database management systems (DBMS) designed and optimized for OLAP[30]. An RDBMS-OLAP is a data system used for storing and analyzing large amounts of data. RDBMS-OLAPs are preferred over RDBMS-OLTPs for OLAP workloads for a few reasons, including storing the data in a columnar format, the compute and storage engines taking advantage of the columnar layout, and usually having a Massively Parallel Processing (MPP) architecture[39]. Data often comes in from a multitude of sources, e.g., application databases, CRMs, etc., in a structured

Data Warehouse

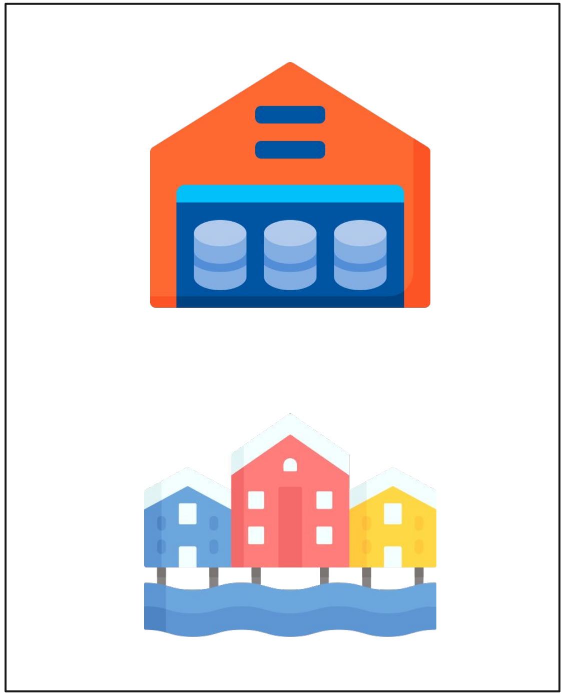
Data Warehousing (DWH)

- Data Warehouse is an ‘overloaded’ term
- Usually refers to 2 different things:
  - Technology
  - Tech-independent Practices
- DWH = Technology + Practices



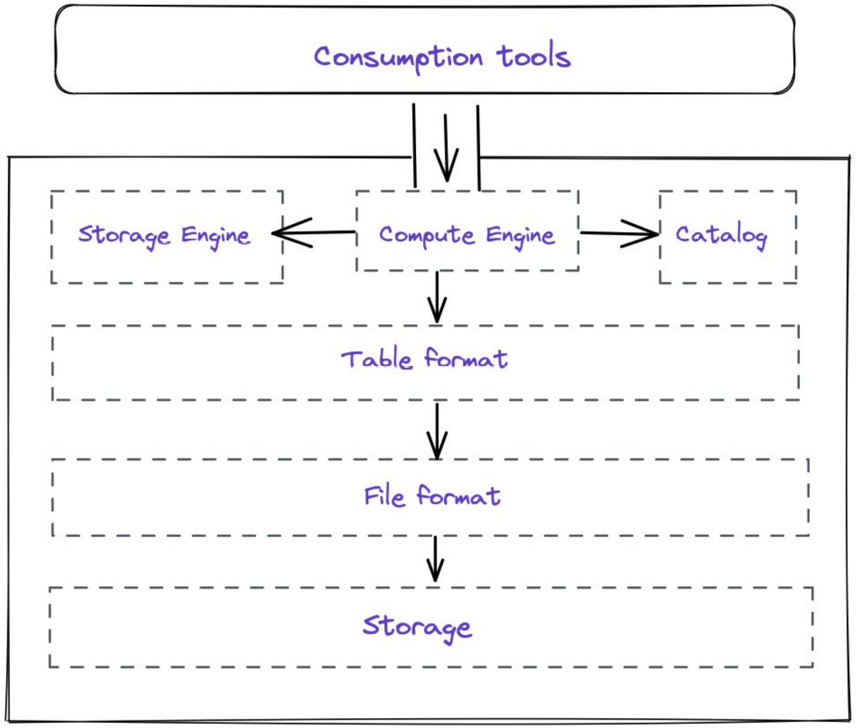
Defining the Requirements

- How do we compare?
- Distilling down into 3 aspects:
    - Technical Components
    - Technical Capabilities
    - Technology-independent Practices



Technical Components (DWH)

- Storage
- File Format
- Table Format
- Storage Engine
- Compute Engine
- Catalog

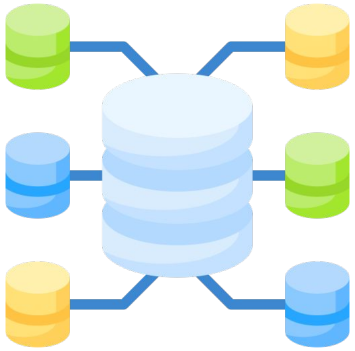


Technical Capabilities (DWH)

- Governance & Security
- High Concurrency
- Low Query Latency
- Ad hoc queries
- Workload Management (WLM)
- Schema & Physical Layout Evolution
- ACID-compliant Transactions

Tech-Independent Practices (DWH)

- Data Modeling
- ETL/ELT
- Data Quality
    - Master Data Management (MDM)
    - Referential Integrity
    - Slowly Changing Dimensions (SCD)



Challenges (DWH)

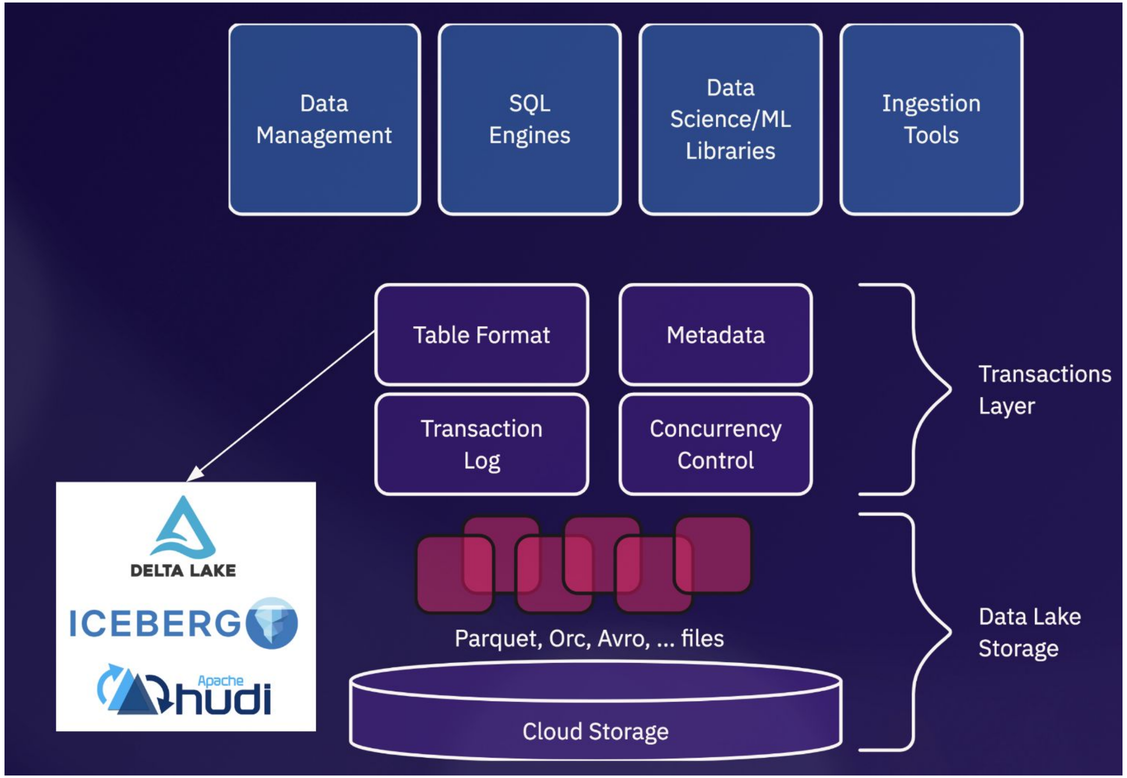
Structured Workloads

Vendor Lock-in

High Costs

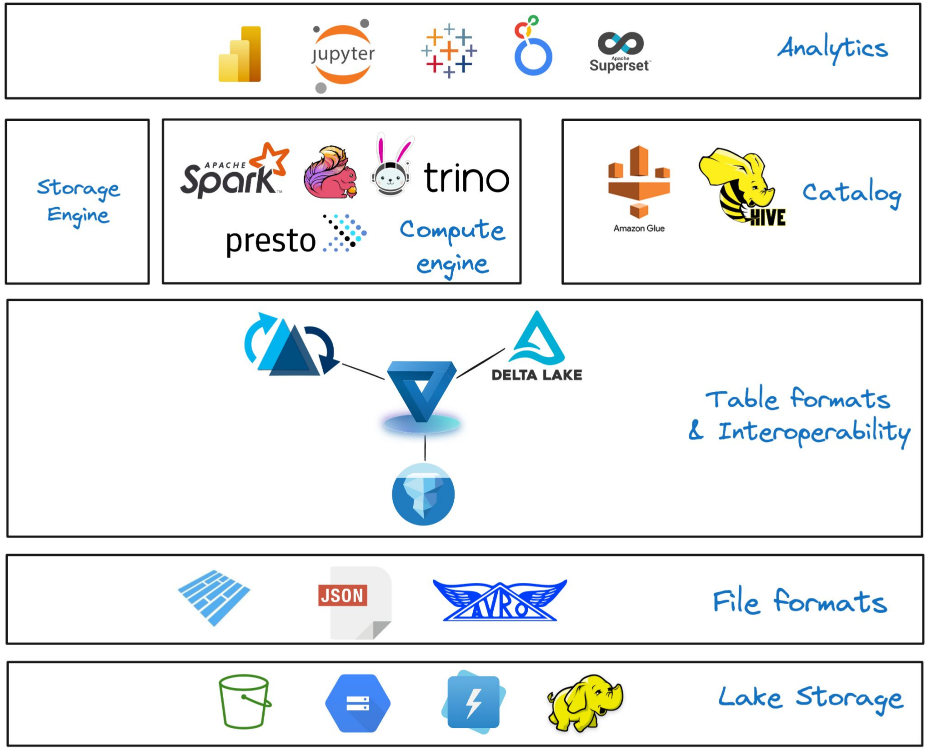
Data Lakehouse

Data Lakehouse (DLH)

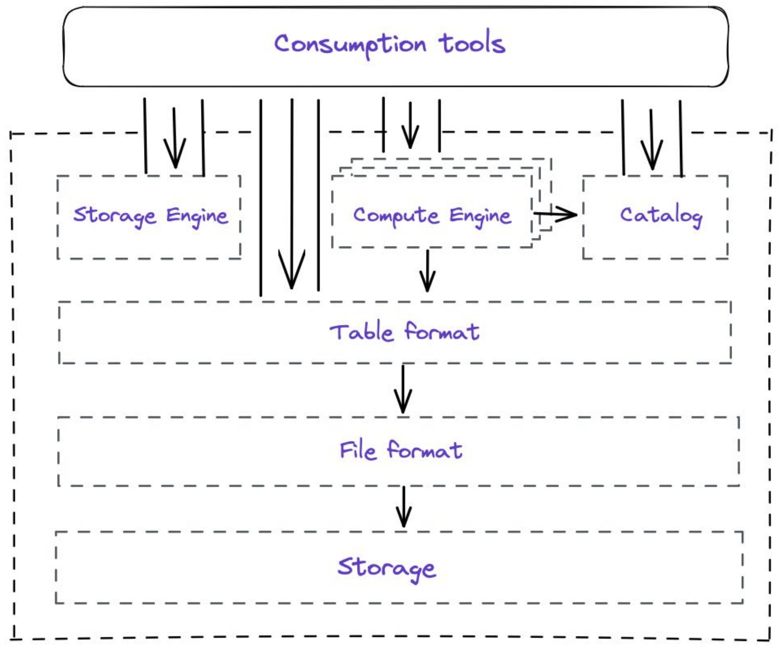


Data Lakehouse Characteristics

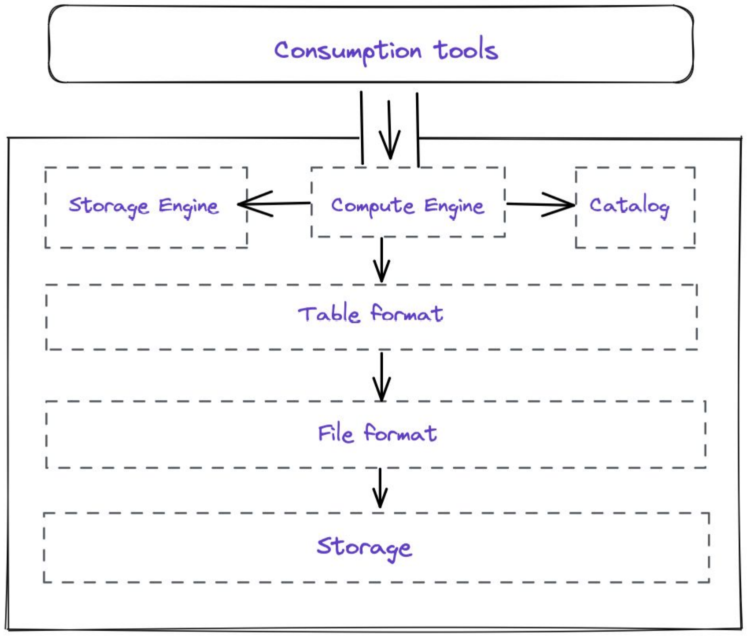
- Transactional Support (ACID)
- Open Data Architecture
- Schema Management
- Scalability
- Less data movement



Technical Components (DLH)

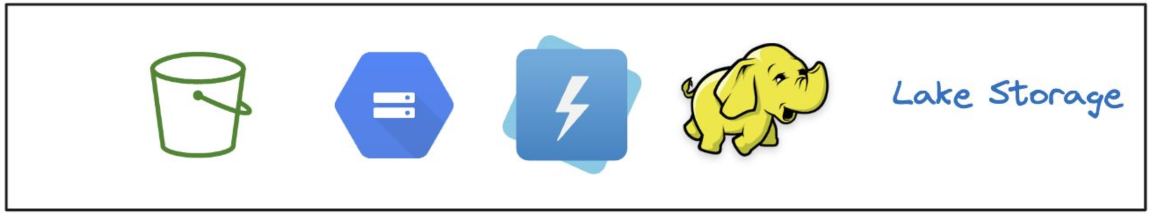


Lakehouse



Warehouse

Storage



- Data lands after ingestion from operational systems
- Data files (such as Parquet) stored
- Supports storing data of any type (structured, unstructured)
- Cloud object stores: AWS S3, GCS, Azure; On-Prem: HDFS

File Format

- Holds raw data & are physically stored on Data Lakes
- Usually in columnar formats but can be row-based
- Open file formats allows access to different compute

Table Format

- Organize the data files (Parquet) as a single ‘table’ - an abstraction
- File Layout, schema, metadata

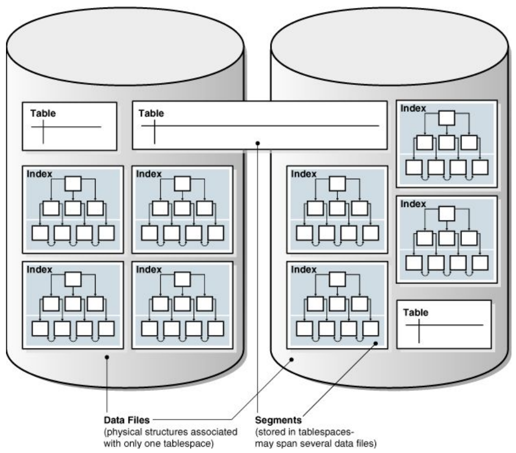


Table
Table
Index
Index
Index
Index
Index
Index
Index
Index
Table
Data Files
(physical structures associated with only one tablespace)
Segments
(stored in tablespaces- may span several data files)

ORACLE

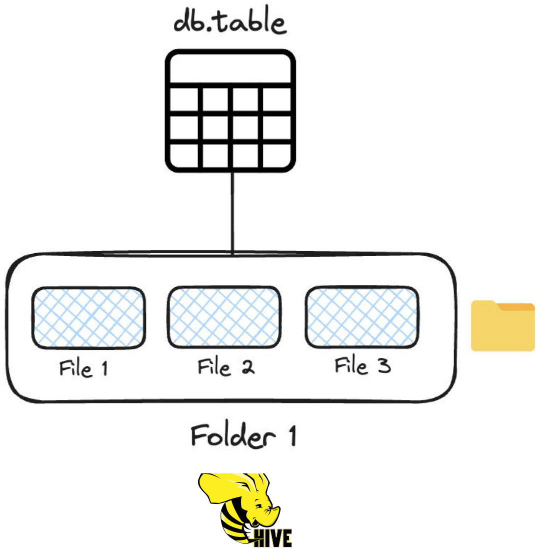
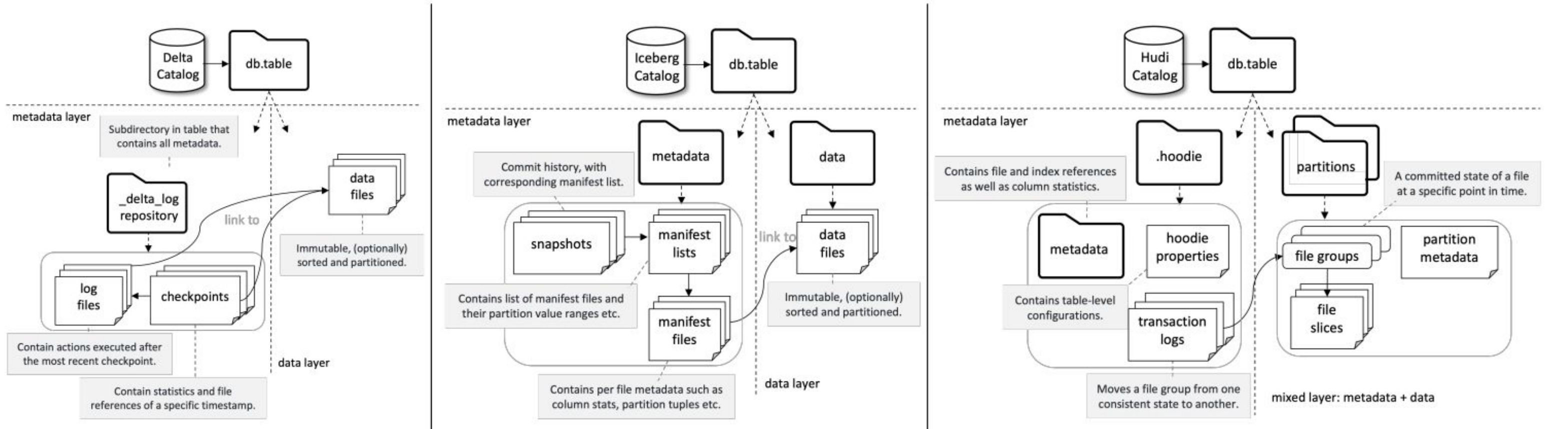


Table Format: Under the Hood

- Tables with SQL semantics and schema evolution
- ACID transactions
- Updates and deletes (merge/upsert)
- Data layout optimizations for performance tuning


Delta Catalog
db.table
metadata layer
Subdirectory in table that contains all metadata.
_delta_log repository
link to
data files
Immutable, (optionally) sorted and partitioned.
log files
checkpoints
Contain actions executed after the most recent checkpoint.
Contain statistics and file references of a specific timestamp.
data layer
Iceberg Catalog
db.table
metadata layer
metadata
data
Commit history, with corresponding manifest list.
snapshots
manifest lists
link to
data files
Contains list of manifest files and their partition value ranges etc.
manifest files
Immutable, (optionally) sorted and partitioned.
Contains per file metadata such as column stats, partition tuples etc.
data layer
Hudi Catalog
db.table
metadata layer
.hoodie
partitions
Contains file and index references as well as column statistics.
A committed state of a file at a specific point in time.
metadata
hoodie properties
file groups
partition metadata
file slices
Contains table-level configurations.
transaction logs
Moves a file group from one consistent state to another.
mixed layer: metadata + data

(a) Delta (b) Iceberg (c) Hudi

Table Format: Under the Hood

- Fundamentals of table formats Hudi, Delta, Iceberg are not that different
- Each has a special metadata layer on top of Parquet

Apache hudi

```
s3_bucket/my_table/
 |- .hoodie/
 |     |- hoodie.properties
 |     |- metadata/
 |- file_1.parquet
 |- file_2.parquet
 |- file_N.parquet
```

DELTA LAKE

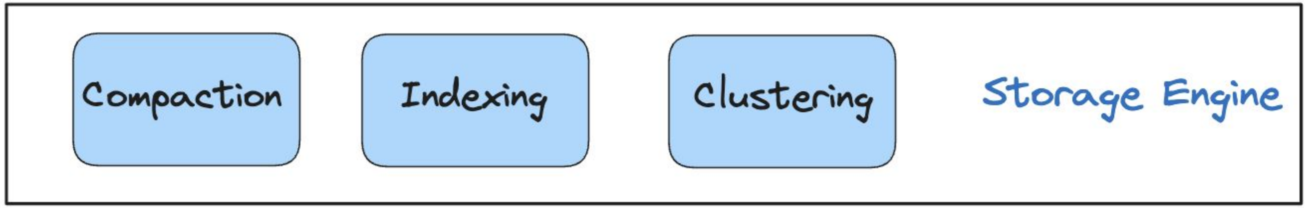
```
s3_bucket/my_table/
 |- _delta_log/
 |     |- 000000.json
 |- file_1.parquet
 |- file_2.parquet
 |- file_N.parquet
```

ICEBERG

```
s3_bucket/my_table/
 |- metadata/
 |     |- v1.metadata.json
 |     |- snap-9fa1-2-16c3.avro
 |     |- 0d9a-98fa-77.avro
 |- file_1.parquet
 |- file_2.parquet
 |- file_N.parquet
```

Amazon S3 Google Cloud Storage Azure Data Lake Storage HDFS

Storage Engine

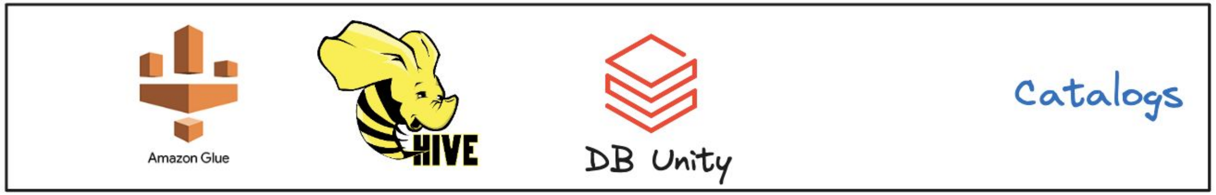


- Keeps data layout optimized for performance
- Table Management tasks - Compaction, Clustering, Cleaning, Indexing
- Enabled by Table formats with Compute engine

Compute Engine

- Responsible for processing data
- Interacts with Open Table Formats' APIs
- Cater to different types of workloads - Ad hoc SQL, Distributed ETL, Streaming engines

Catalog



- Logical separation of metastore
- Efficient search & data discovery via metadata
- Governance, Security & Data Federation

Technical Capabilities (DLH)

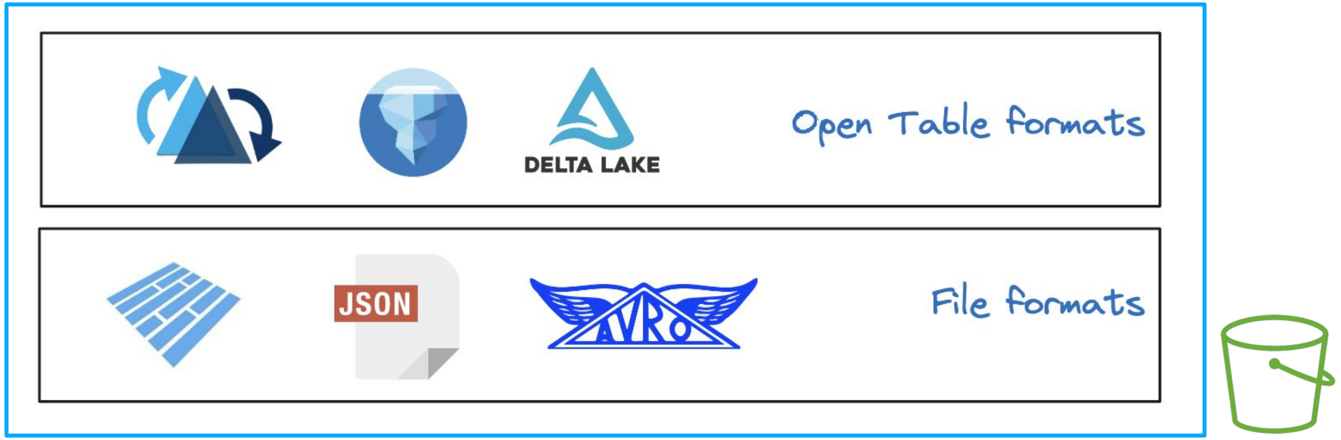
Capability	Data Lakehouse
Governance & Security	Apache Ranger, Lakehouse Platforms
High Concurrency	Concurrency Control, Engines Scalable
Low Query Latency	Clustering, Partitioning, Indexing
Ad hoc Queries	Compute engines (Presto, Trino), BI tools
Workload Management (WLM)	Isolated workloads for different users
Schema & Physical Layout Evolution	Evolve schema with Table formats
ACID Transactions	Table formats brings consistency

Tech-Independent Practices (DLH)

Practice	Data Lakehouse
Data Modeling	Various modeling techniques; Different Layers (Bronze, Silver, Gold)
ETL/ELT	Schema-on-Read
Data Quality	MDM, SCD, Pre-commit checks, WAP

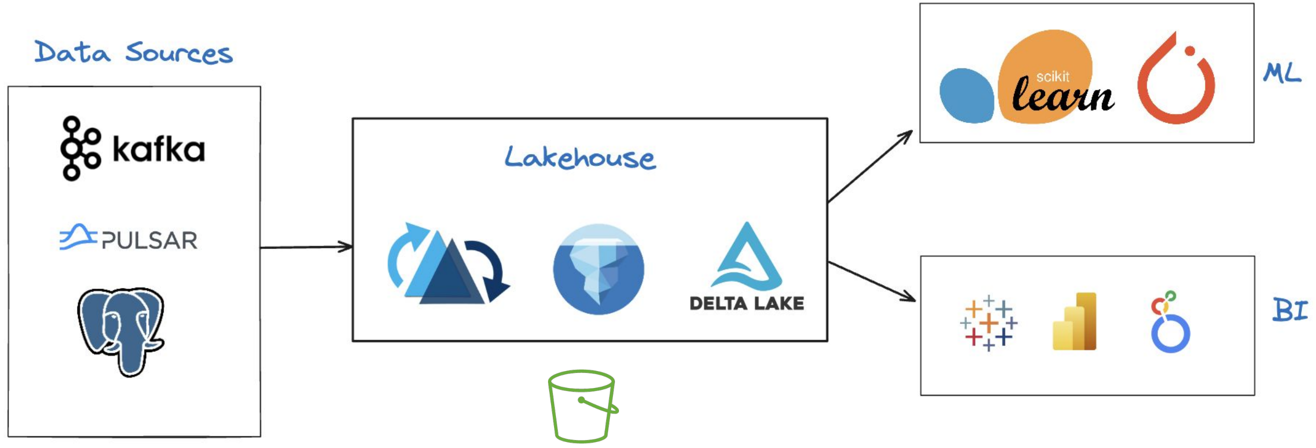
Additional Values

Open Data Architecture



- Data stored as an open & independent tier
- Open to multiple engines
- Eliminates vendor lock-in

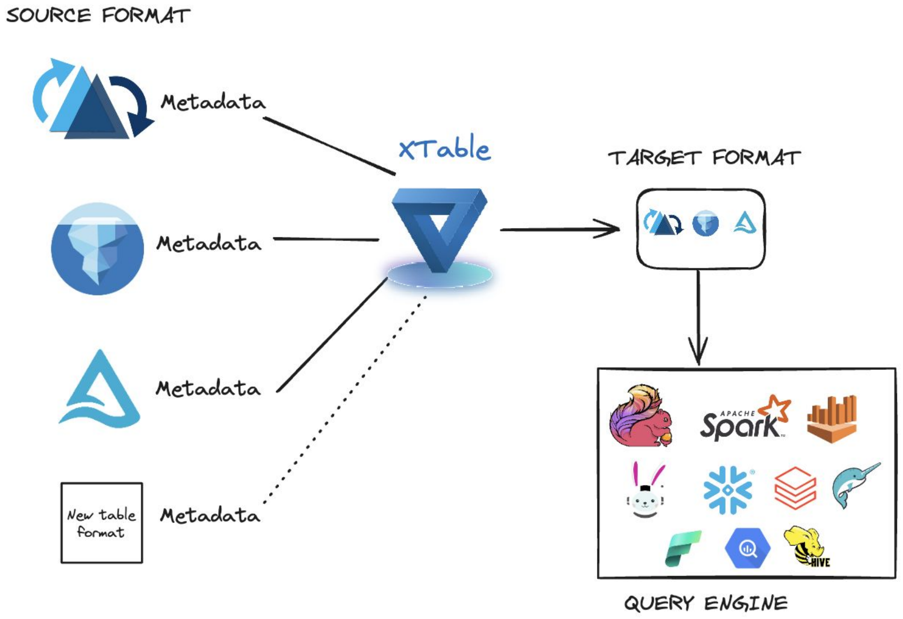
Fewer Data Copies



- Less Data movement, More Governance (unlike a 2-tier architecture)
- Query directly on the lake using various technologies

Interoperate between Formats

- Choosing a table format maybe tough
- Each project has rich features that may fit different use-cases
- Newer use cases requires formats to be interoperable
- **Apache XTable** (incubating) for interoperability



Apache XTable

- An omni-directional interop of lakehouse table formats
- NOT a *new or separate* format
- XTable provides abstractions and tools for the translation of metadata

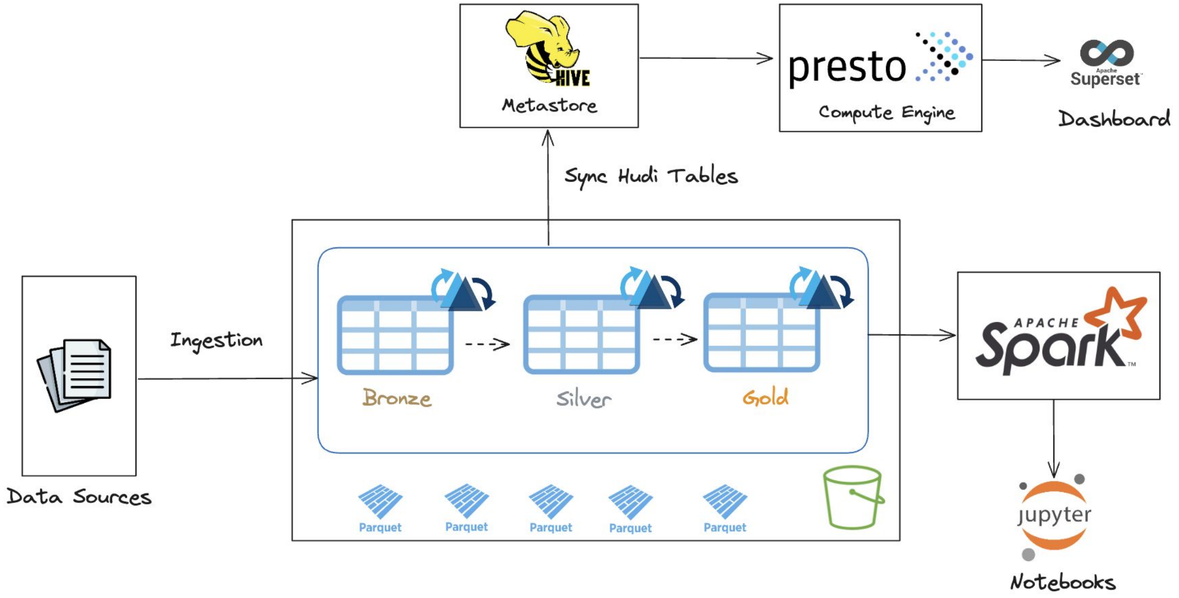
Read your table as any of the formats:

```
# any of these work on the same table
spark.read.format("hudi")
spark.read.format("delta")
spark.read.format("iceberg")
```

1: Choose your "source" format
2: Choose your "destination" format(s)
3: XTable will translate the metadata layers

```
s3_bucket/my_table/
 |- .hoodie/
 |     |- hoodie.properties
 |     |- metadata/
 |- _delta_log/
 |     |- 000000.json
 |- metadata/
 |     |- v1.metadata.json
 |     |- snap-9fa1-2-16c3.avro
 |     |- 0d9a-98fa-77.avro
 |- file_1.parquet
 |- file_2.parquet
 |- file_N.parquet
```

Simple Lakehouse Implementation



Revisiting Hypothesis

Going beyond the jargons

Data Warehousing = Tech + Practices

Data Lakehouse = DWH + DL + Additional Stuff

Advantages (Open architecture, Interoperability)

Q&A