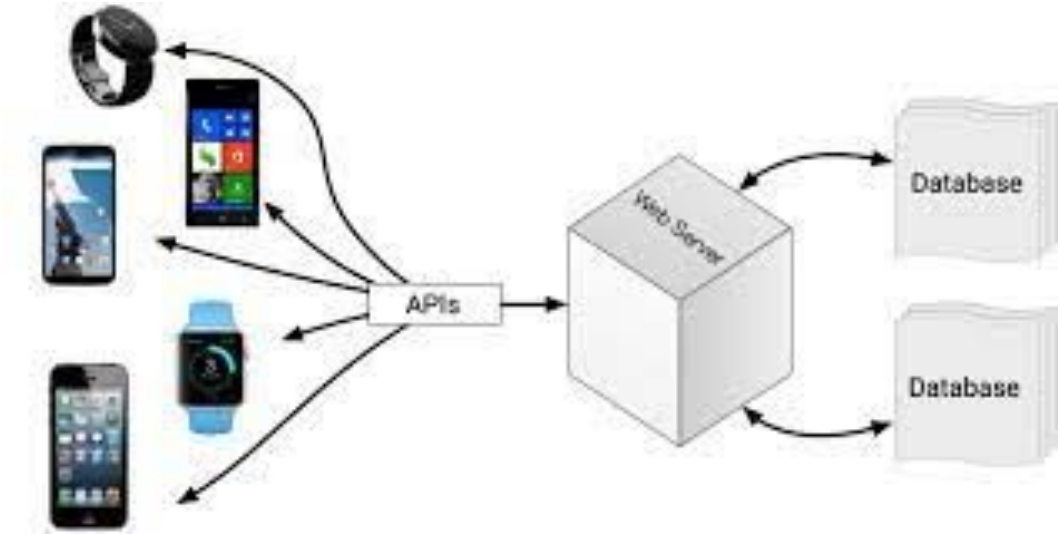
ADALTAS

Types of data storages

Data Warehouse - Data Lake - Lakehouse

First there was a database...

- …and it still is :)
- to collect the data
- **GOAL:** to analyse this data
  - can we do it in-place?



Time	UsrID	GPS coords

Time	UsrID	Heart rate

Time	UsrID	Steps

Time	UsrID	Phone call

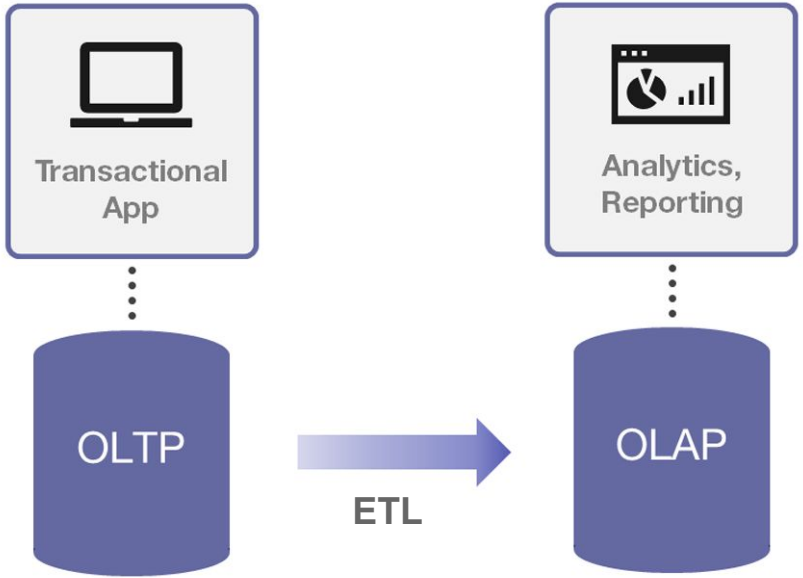
Time	UsrID	App usage

Difference between OLTP and OLAP

ADALTAS

- High volume of transactions
- Fast processing
- Normalized data
- Many tables
- *"Who bought X?"*

ACID compliant



- High volume of data
- Slow queries
- Denormalized data
- Fewer tables
- *"How many people bought X?"*

https://www.guru99.com/oltp-vs-olap.html

Imagine this company-wise

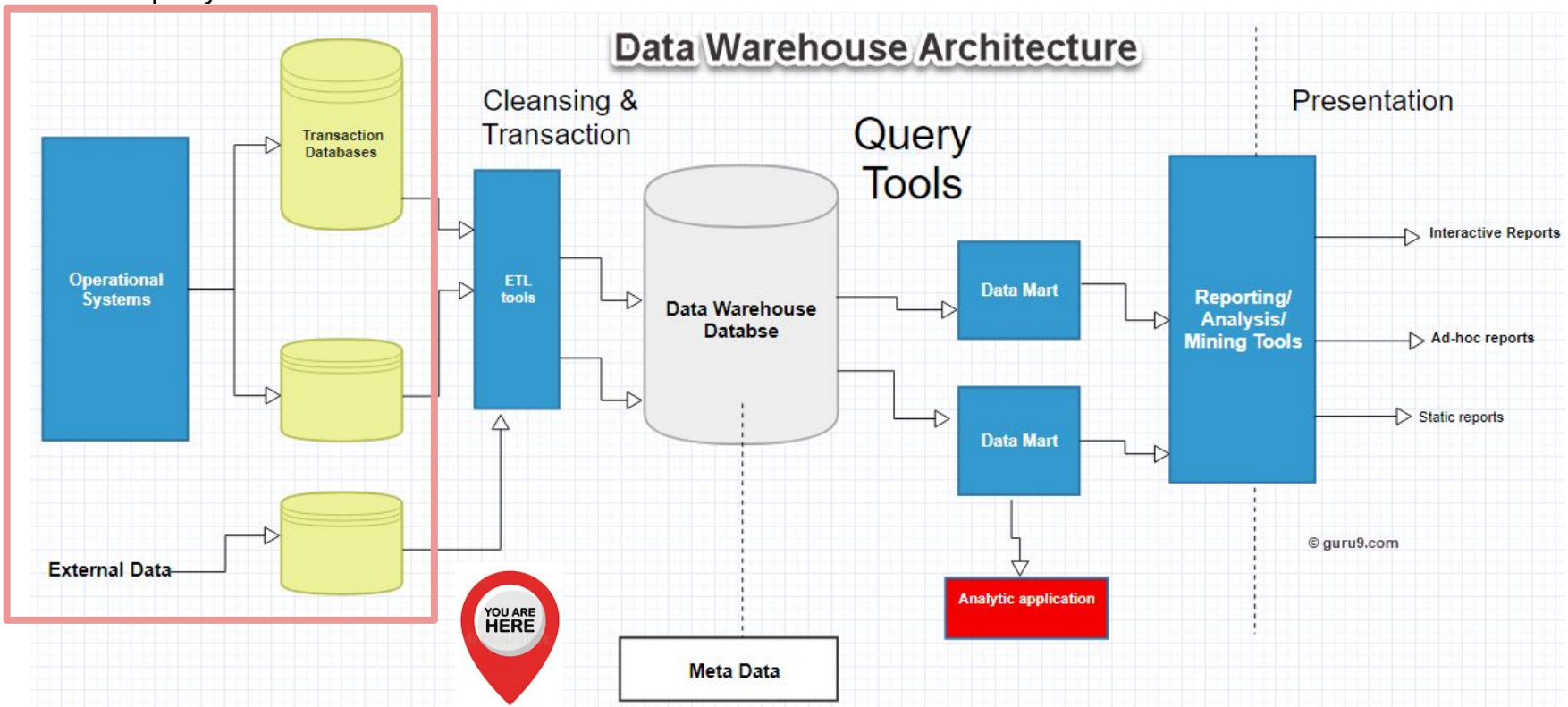
ADALTAS

- Many departments collect or buy the data
- Everybody would like to get insights from these data
- **BUT:**
  - The data might be (or, probably is) messy
  - Different bits of information reside in different tables (joins)
- Does every department manage the collection, cleaning and maintenance by themselves?

Data warehouse

ADALTAS

On a company level:



https://www.guru99.com/data-warehouse-architecture.html

ETL (Extract-Transform-Load) Process

ADALTAS

- Extract the data from its source
- Transform it into a pre-defined form
  - Remove corrupted data (schema-on-write)
  - Consolidate the date format, gender notation…
- Load into the data warehouse
- Benefit:
  - The data is clean and ready to use
- Issues:
  - Proprietary solutions (expensive, data lock-in)
  - Lots of work
  - What if the data is never exploited?
  - What if the data is not structured?
    - Not very ML-friendly

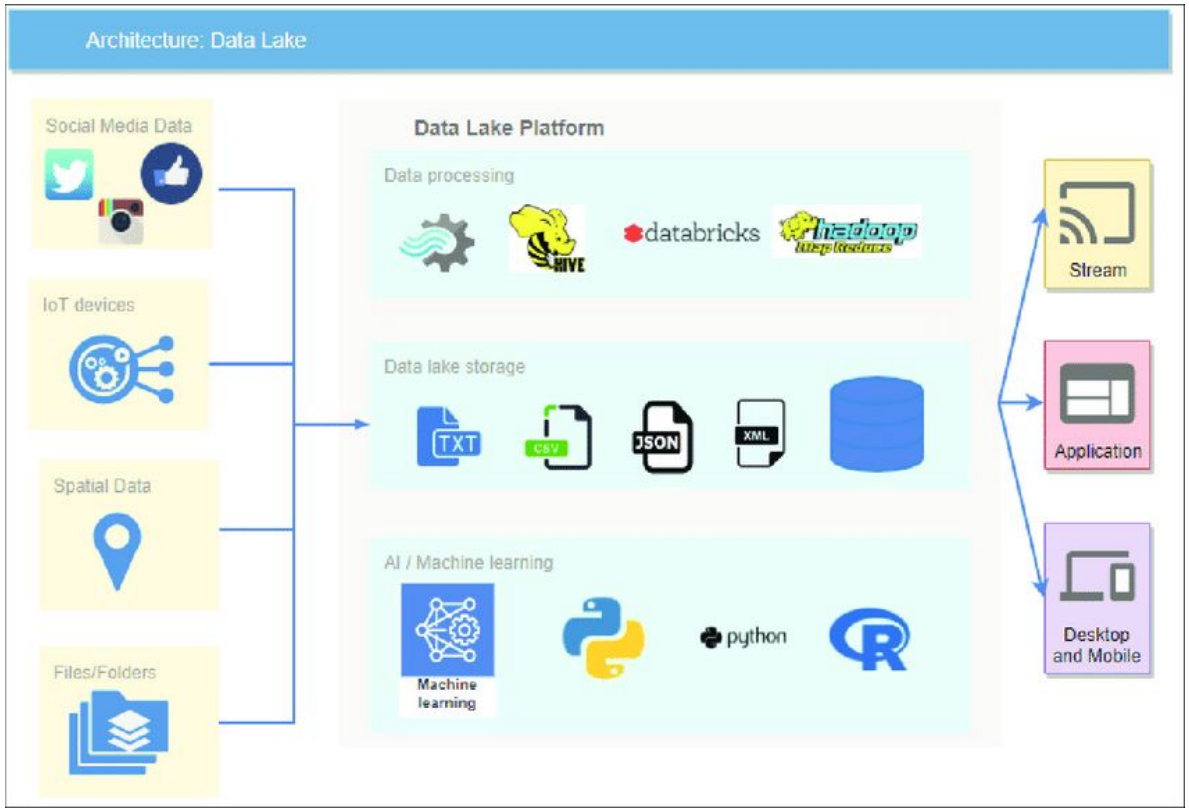
Upgrade

ADALTAS

Idea:

- Store the data as it comes
- Prepare it only if / when needed

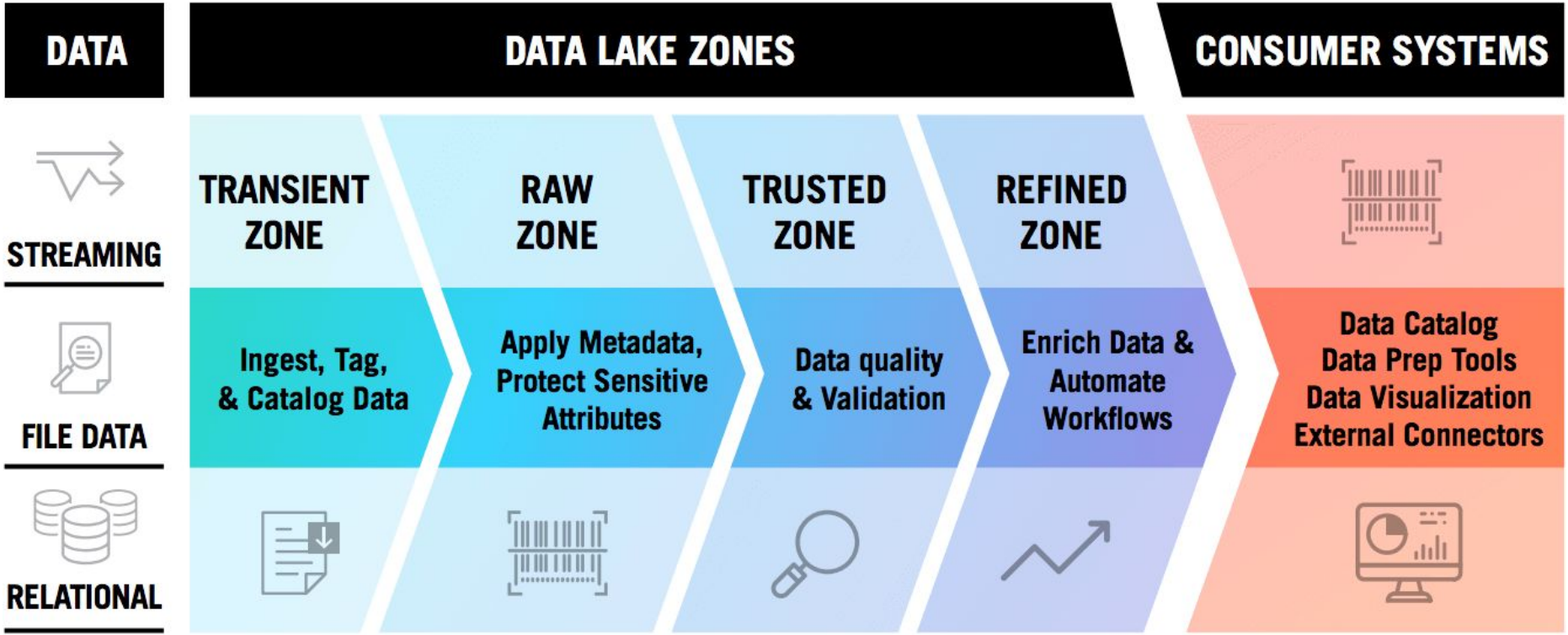
Data Lake (two-tier architecture)



- Cloud object stores
  - Key/value pairs
  - Schema-on-read
- Data Warehouse

Data Lake: Governance

ADALTAS



Data Lake: Pros and Cons

ADALTAS

Pros:

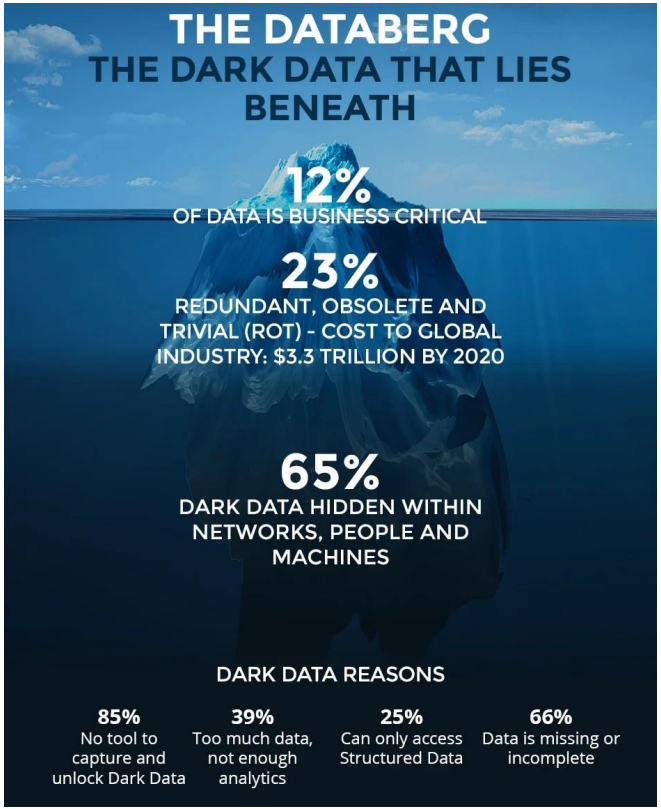
- many different types of data
- data doesn’t need to be structured
- lesser workload for transformations: ELT instead of ETL

Cons:

- how to find anything back?!?
  - cataloging
  - lake-to-swamp
- staleness of data due to “the zones”
- “data over-acquisition”
  - do we really need all of it? => dark data

Intermezzo: Dark data

ADALTAS



Can we simplify? => Lakehouse

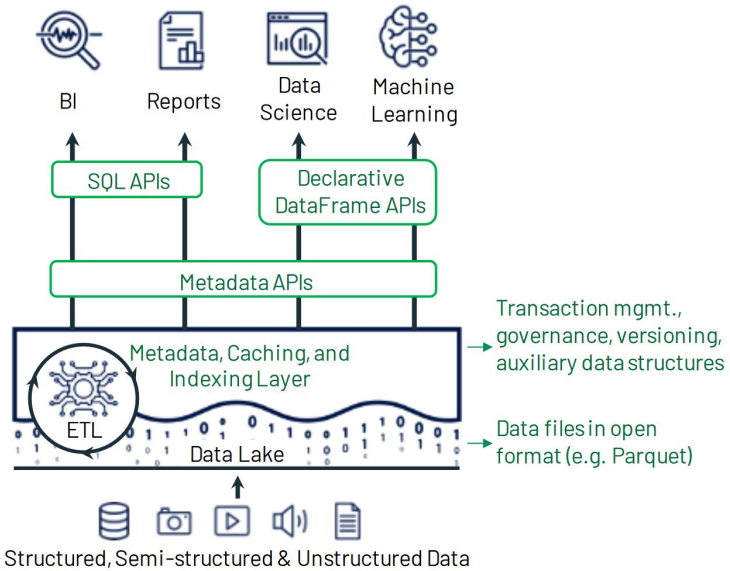
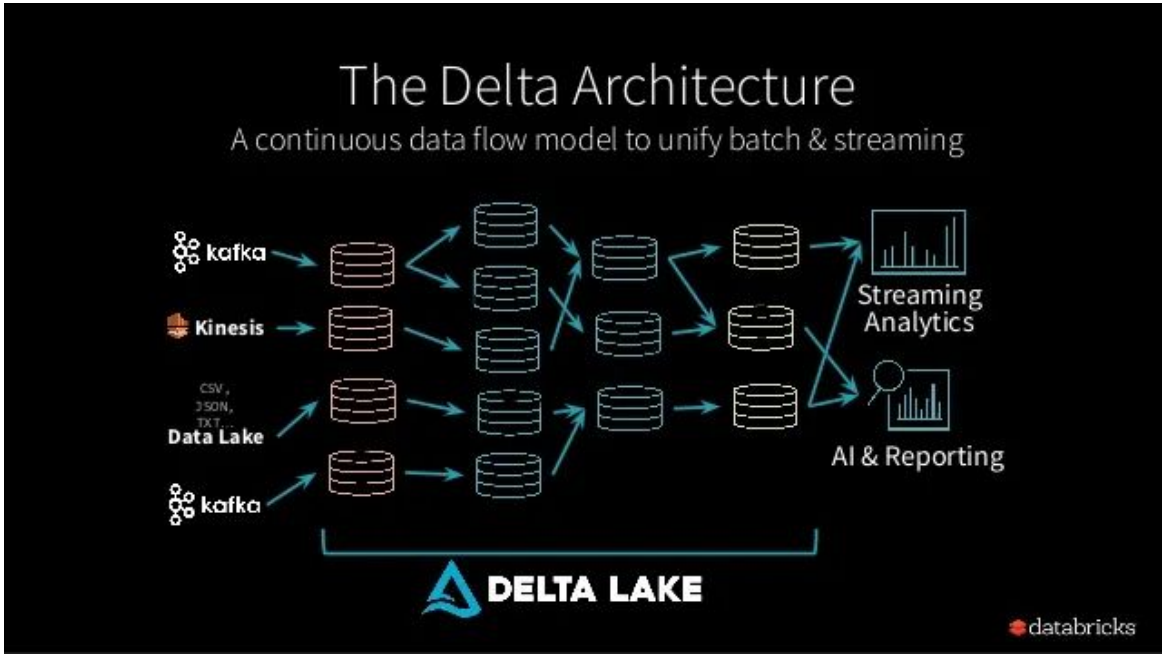


Figure 2: Example Lakehouse system design, with key components shown in green. The system centers around a metadata layer such as Delta Lake that adds transactions, versioning, and auxiliary data structures over files in an open format, and can be queried with diverse APIs and engines.

- "Data Lake behaving like a database"
  - ACID transactions
  - Management on the level of files
    - Logs

https://databricks.com/fr/wp-content/uploads/2020/12/cidr_lakehouse.pdf

Implementation of Lakehouse: Delta Lake



Based on this transactional design, we were also able add multiple other features in Delta Lake that are not available in traditional cloud data lakes to address common customer pain points, including:

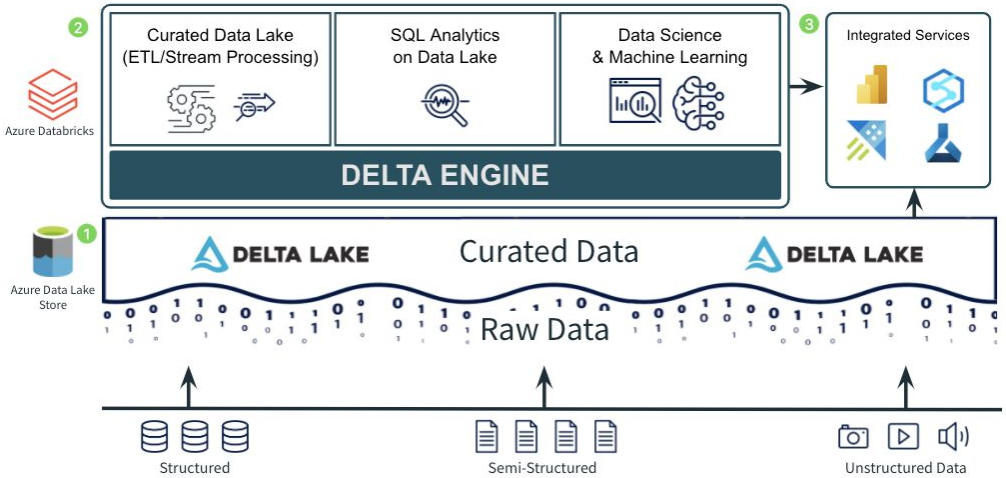
- **Time travel** to let users query point-in-time snapshots or roll back erroneous updates to their data.
- **UPSERT, DELETE and MERGE operations,** which efficiently rewrite the relevant objects to implement updates to archived data and compliance workflows (e.g., for GDPR [27]).
- **Efficient streaming I/O,** by letting streaming jobs write small objects into the table at low latency, then transactionally coalescing them into larger objects later for performance. Fast "tailing" reads of the new data added to a table are also supported, so that jobs can treat a Delta table as a message bus.
- **Caching:** Because the objects in a Delta table and its log are immutable, cluster nodes can safely cache them on local storage. We leverage this in the Databricks cloud service to implement a transparent SSD cache for Delta tables.
- **Data layout optimization:** Our cloud service includes a feature that automatically optimizes the size of objects in a table and the clustering of data records (e.g., storing records in Z-order to achieve locality along multiple dimensions) without impacting running queries.
- **Schema evolution,** allowing Delta to continue reading old Parquet files without rewriting them if a table's schema changes.
- **Audit logging** based on the transaction log.

https://towardsdatascience.com/the-fundamentals-of-data-warehouse-data-lake-lake-house-ff640851c832 (contains a Delta Lake paper from Databricks)

Important Building Blocks of Delta Lake

ADALTAS

- Delta **tables** - contain the data, enable time-traveling
- Delta **transaction logs** - it stores every executed transaction -> single source of truth for delta table changes
- Delta **engine** - performance optimization for SQL and DataFrames



Delta Bookkeeping

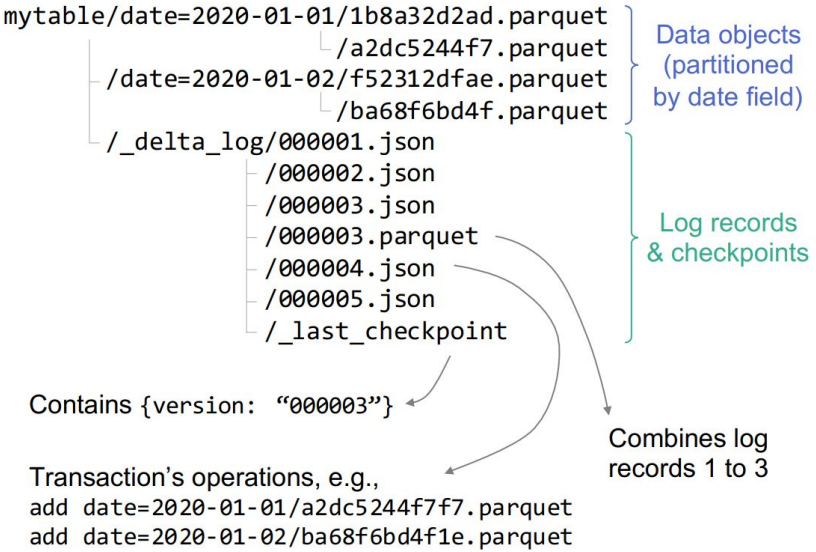


Figure 2: Objects stored in a sample Delta table.

```
1  DESCRIBE HISTORY flightdelays
```

▸ (1) Spark Jobs

version	timestamp	userId	userName	operation	notebook
7	2019-10-08T16:47:22	101543	...@databricks.com	MERGE	▸ {"notebookId":"25"}
6	2019-10-08T16:44:16	101543	...@databricks.com	MERGE	▸ {"notebookId":"25"}
5	2019-10-06T19:26:53	101543	...@databricks.com	UPDATE	▸ {"notebookId":"25"}

Figure 3: `DESCRIBE HISTORY` output for a Delta Lake table on Databricks, showing where each update came from.

Data lakes vs. Data lakehouses vs. Data warehouses

	Data lake	Data lakehouse	Data warehouse
Types of data	All types: Structured data, semi-structured data, unstructured (raw) data	All types: Structured data, semi-structured data, unstructured (raw) data	Structured data only
Cost	$	$	$$$
Format	Open format	Open format	Closed, proprietary format
Scalability	Scales to hold any amount of data at low cost, regardless of type	Scales to hold any amount of data at low cost, regardless of type	Scaling up becomes exponentially more expensive due to vendor costs
Intended users	Limited: Data scientists	Unified: Data analysts, data scientists, machine learning engineers	Limited: Data analysts
Reliability	Low quality, data swamp	High quality, reliable data	High quality, reliable data
Simplicité d'utilisation	Difficult: Exploring large amounts of raw data can be difficult without tools to organize and catalog the data	Simple: Provides simplicity and structure of a data warehouse with the broader use cases of a data lake	Simple: Structure of a data warehouse enables users to quickly and easily access data for reporting and analytics
Performance	Poor	High	High

https://databricks.com/fr/discover/data-lakes/introduction

ADALTAS

Extras

ADALTAS

- Conflict resolution (https://docs.databricks.com/delta/concurrency-control.html)
- Very good technical talks with many details on Databricks YouTube channel: https://www.youtube.com/c/Databricks/playlists

Other Lakehouse implementations

ADALTAS

- Apache Hudi (https://hudi.apache.org/)
- Apache Iceberg (https://iceberg.apache.org/)

To go further...

ADALTAS

- Storage on-premise vs. storage in cloud
  - Different technology
  - Different behavior
- Data processing architectures (maybe off-topic)
  - How you treat streaming and batch data?
  - https://jixjia.com/delta-architecture/ (cool animations)