**Difference between Data Warehouse , Data Lake and lakehouse**

Historically, you’ve had two primary options for a data repository: [data lake or data warehouse](https://www.qlik.com/us/data-lake/data-lake-vs-data-warehouse). To support BI, data science and machine learning, it’s likely that you’ve had to maintain both of these structures simultaneously and link the systems together. This often leads to data duplication, security challenges and additional infrastructure expense. Data lakehouses can overcome these issues.

<https://www.qlik.com/us/data-lake/data-lakehouse>

What is Delta Lake ,Delta Tables ?

Delta Lake is an open-source storage layer that brings ACID (Atomicity, Consistency, Isolation, Durability) transactions to big data workloads. It's built on top of Apache Spark and is fully compatible with Spark APIs. Delta Lake stores data as Parquet files in DBFS, Azure Blob storage, Azure Data Lake Storage, or other storage systems, and maintains a transaction log to keep track of all data modifications.

Here's a high-level overview of how Delta Lake works in Azure Databricks:

\*\*1. Storing Data:\*\*

When you write data into a Delta Lake table, the data is stored as a collection of Parquet files in a directory of your choice. You can use DBFS (Databricks File System), Azure Blob Storage, or Azure Data Lake Storage to store these files.

```python

data.write.format("delta").save("/mnt/delta/my\_table")

```

\*\*2. Transaction Log:\*\*

Along with the Parquet files, Delta Lake also maintains a transaction log. This log records details about every change made to the data, including added/removed files, changes to the metadata, etc. This transaction log is crucial as it allows Delta Lake to provide ACID transactions and time travel capabilities.

\*\*3. Reading Data:\*\*

When you read data from a Delta Lake table, Delta Lake uses both the Parquet files and the transaction log to provide a consistent view of the data.

```python

df = spark.read.format("delta").load("/mnt/delta/my\_table")

```

\*\*4. ACID Transactions:\*\*

With the transaction log, Delta Lake can handle multiple concurrent reads and writes, ensuring that each reader gets a consistent and isolated view of the data. This means you can have multiple jobs concurrently reading and writing to the same Delta Lake table without worrying about inconsistencies.

\*\*5. Schema Enforcement:\*\*

When data is written into a Delta Lake table, the schema of the data is checked against the table's current schema. This helps prevent corrupt or malformed data from being written into the table.

\*\*6. Time Travel:\*\*

With the transaction log, Delta Lake provides snapshots of the data at different points in time, allowing you to access previous versions of the data (time travel).

Delta Lake tables are stored in a directory of your choice, and you can use standard Spark APIs or Spark SQL to read/write data. The data in Delta Lake tables is stored in Parquet format, a columnar storage file format that is optimized for use with big data processing frameworks like Apache Spark.