\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* open table format - Apache iceberg \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\* how to update in S3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Apache Iceberg is an open table format designed for huge, petabyte-scale tables.

The function of a table format is to determine how you manage, organize and track all of the files that make up a table. You can think of it as an abstraction layer between your physical data files (written in Parquet or ORC etc.) and how they are structured to form a table.

Note that columnar file formats like parquet, ORC etc that are generally used in Big Data are inefficient for Transactional Operations i.e. writing new records or updating existing ones can be slower since it involves accessing multiple columns scattered across different storage locations. If you have to update a row, in row-based formats, you will identify the row with a single seek and a contiguous read. In column-based formats, assembling a row is much more costlier and inefficient because all values of a row are not stored in contiguous blocks here .. only all values of a column are stored in contiguous blocks. Open table formats like Iceberg try to address this problem. In my own words, Iceberg combines the simplicity of SQL tables with the cost-effectiveness of low-cost storage that columnar file formats provide. Iceberg brings the reliability and simplicity of SQL tables to big data.

Some of the important features of Apache Iceberg are as follows:

1. Transactional consistency between multiple applications where files can be added, removed or modified atomically, with full read isolation and multiple concurrent writes (ACID)

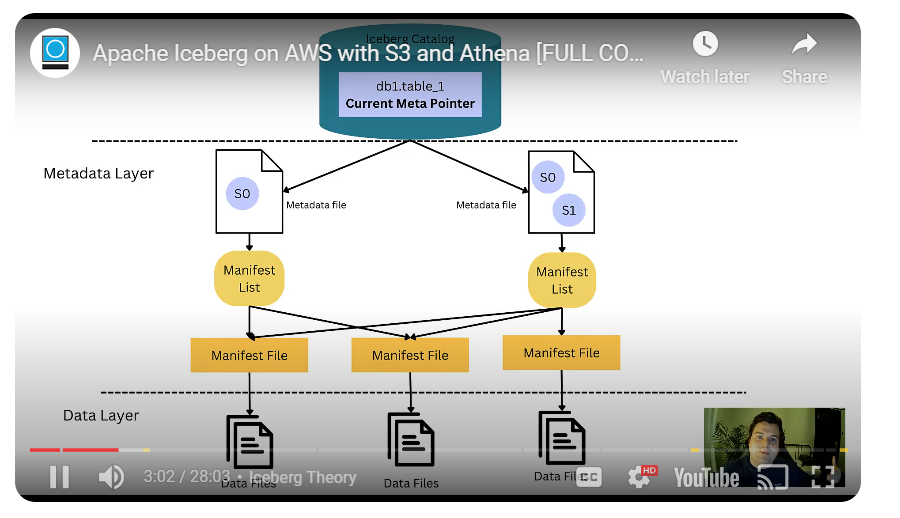
2. Full schema evolution to track changes to a table over time

3. Time travel to query historical data and verify changes between updates

4. Rollback to prior versions to quickly correct issues and return tables to a known good state

5. Advanced planning and filtering capabilities for high performance on large data volumes

**How this achieved ?**



* There is a metadata layer and a data layer. The metadata layer consists of point-in-time snapshots referring to manifest lists. Each manifest list has 1 or many manifest file/files that contain information about the data files at a certain point of time. The info in manifest file could be max, min, distinct values etc. So when a query is executed, through the use of manifest files, iceberg is able to identify the required data quickly. This is what enables point 5 above.
* Multiple snapshots is what enables points 2, 3 and 4 above.
* A Snapshot is committed only when all data files are ready. This is what enables point 1 above.

Note:

* Delta Lake and Hudi are the other examples for open-table format.
* They are called Open table formats because the technology is Open source in nature and allows/encourages collaborative innovation, ensuring users benefit from the latest data management advancements.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Data Lakehouse \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

In a data Lakehouse environment, data from source systems is ingested into a data lake for storage, processing and analysis. That's done through different data integration methods, including batch ETL or ELT processes and real-time ones such as stream processing and change data capture. The data is then organized to support the traditional uses of both data lakes and data warehouses. There is no another hop that moves data from data lake into a data warehouse.

*Data management:*

Data lakes commonly use the Parquet or Optimized Row Columnar (ORC) file formats to organize data but offer limited capabilities for managing it. Data Lakehouses include newer open source technologies, such as Delta Lake, Apache Hudi and Apache Iceberg, that support ACID transactions, indexing, data validation, version history, "time travel" to earlier versions of data, schema enforcement and other features on top of Parquet and ORC.

So the advantages of Data Lakehouses are:

1. Reduce ETL complexity by eliminating one hop
2. Cost saving on infra of second layer
3. Include all the advantages of Open table formats