**Evolution and development of Data Lakes**

An overview of the development of data lakes is provided below.

1. Hadoop & Hive: A MapReduce-based First-Generation Data Lake Table Format SQL expressions are already enabled.

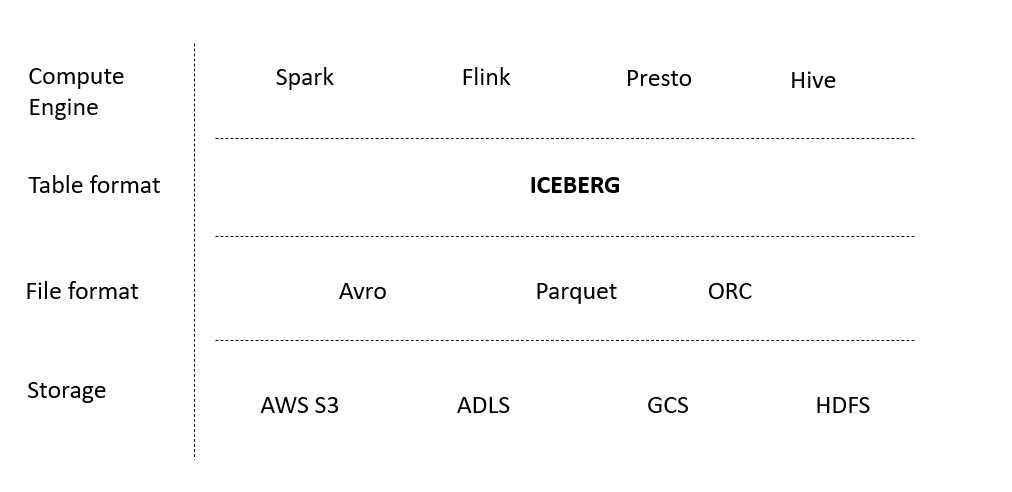
2. The New Generation of Simple Data Lake Storage: AWS S3. There is no function, but there is significantly less maintenance and a great programmable API interface.

3. Data Lake File Format: Appropriate cloud file formats with column-oriented, well-compressed, and Analytics-optimized features. file types like ORC, Apache Avro, and Apache Parquet.

4. Data Lake Table Format: Hudi, Iceberg, and Delta Lake all have comprehensive database-like functionality.

In December 2020, the Databricks article introduced the next-generation architecture, called [Data Lakehouse](https://www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf). The open formats Parquet and ORC can be used, and Data Lake can use cloud object storage, but it lacks the robust management tools found in data warehouses, such as ACID transactions, data versioning, and schema enforcement. In terms of contemporary table formats, we now have more possibilities than ever. They are all designed to address these concerns and support the Lakehouse architecture. The table type you choose to utilise as its foundation is one of the most crucial choices you can make while building a data lakehouse. The decision selected will have an impact on the features, instruments, and workflows that are accessible to you in your lakehouse.

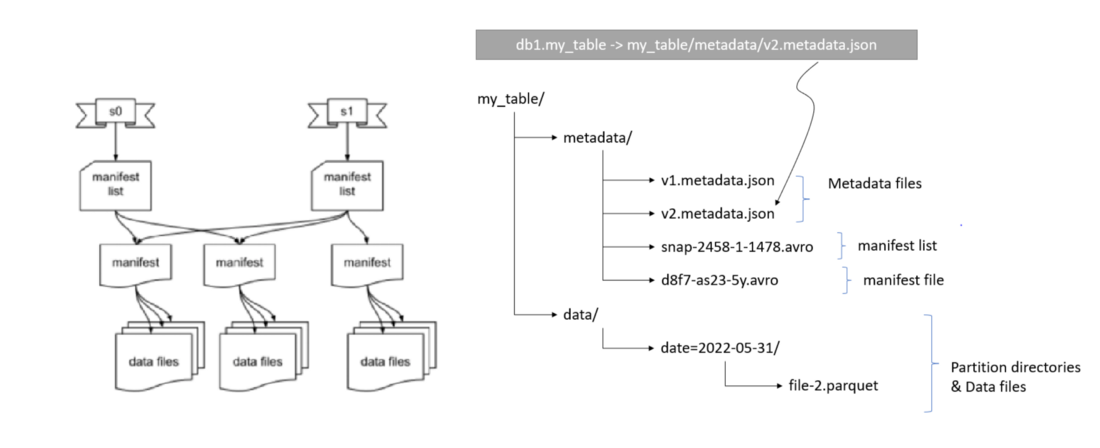
Let's explore what is table formats and what it can offer.

A way to arrange data files is via table formats. They attempt to give the Data Lake features resembling those of databases.

The Apache Iceberg, Apache Hudi, and Delta Lake data lake table formats are covered in detail here to assist you in designing your data lake house with confidence.

**Apache Iceberg**

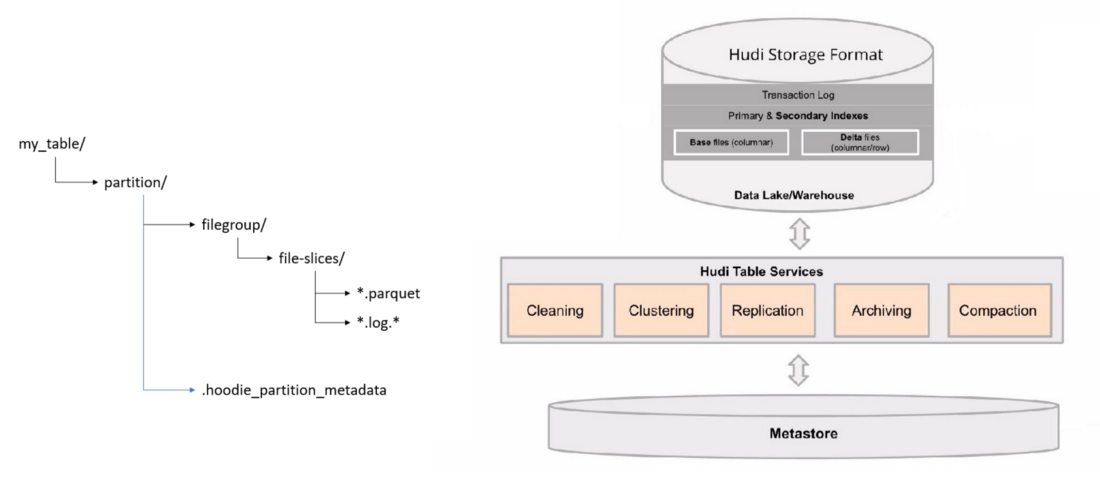
Three different forms of metadata are used by Apache Iceberg to define the table. These kinds include "metadata files" for the table, "manifest lists" for a snapshot of the table, and "manifests" for groups of data files that could be included in one or more snapshots.

The data that supports query optimization and all of Iceberg's features is contained in these three levels of metadata.

Through the metadata tree, which consists of manifest lists, manifests, and metadata files, Iceberg provides ACID support and snapshot isolation. When a query is made, Iceberg always utilises the most recent snapshot, unless otherwise stated. Any writes to a specific table result in a brand-new snapshot that is unaffected by running queries. Concurrent writings are managed using optimistic concurrency (Whoever creates a new snapshot first does so, and subsequent writes are retried.)

In addition to the typical creates, inserts, and merges, Apache Iceberg also supports row-level updates and deletions. Any of these transactions may be carried out using SQL commands.

**Apache Hudi**

All transactions are categorised into many activity kinds that take place along a timetable by Apache Hudi. Timestamped files and log files that track changes to the records in that data file are used by Hudi in a directory-based method. You have the choice to enable a metadata table in Hudi for query efficiency. For large datasets, this table will keep track of a list of files that can be used for query planning rather than file operations, preventing a potential bottleneck. Atomic transactions and support for CREATE TABLE, INSERT, UPDATE, DELETE, and Queries in SQL are also features of Apache Hudi.

### **Delta Lake**

Using two different types of files, Delta Lake tracks metadata:

1. Delta Logs monitor modifications to the table in order.

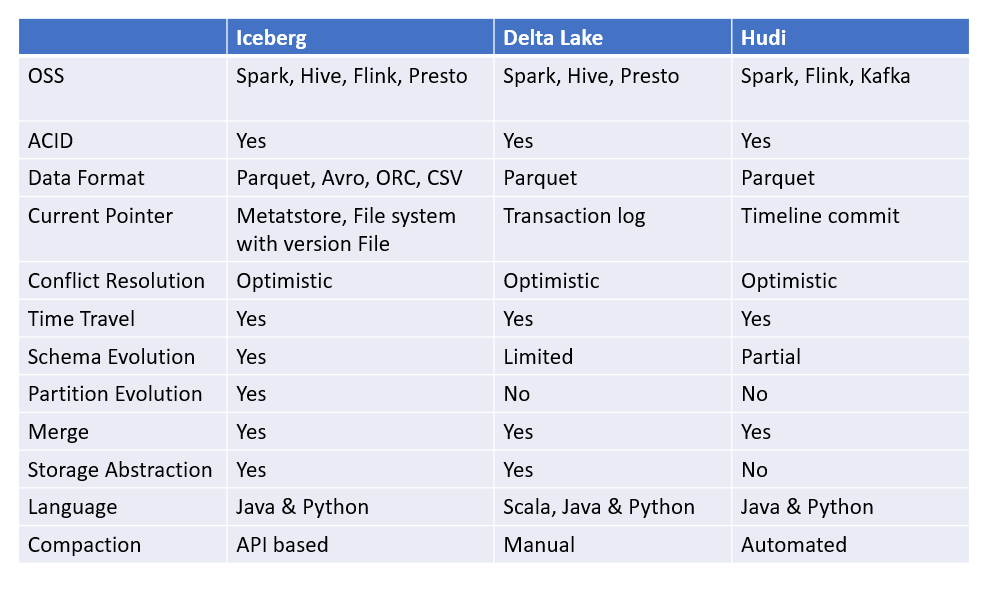
2. Checkpoints compile all table modifications up to that moment, except transactions that cancel one another out.

Along with SQL support for creates, inserts, merges, updates, and deletes, Delta Lake also supports ACID transactions.

A picture containing diagram

Description automatically generated

**Comparison of Data Lake Table Formats and features**

****

**Questions to Ask Before Adopting a Table Format**

Before using a table format, there are some things to consider. Hive will most likely be replaced by one of these three next-generation formats as the preferred method for presenting tables in data lakes. Asking yourself the following questions can help you decide which format to use going forward

* What format contains the most reliable version of the features I require?
* Which format makes it possible for me to use SQL to access the majority of its capabilities and make it available to my data consumers?
* Which format has the advantage in terms of community and engine support?
* Which version-control toolset will I have access to in that format?
* Does the format permit me to read and write using the tools I already use?
* Do I think I may change tools or introduce new tools in the future? How important is broad tool compatibility for me?
* How often will my schema evolve? Does this table format allow me to evolve my schema in the way I need?
* Might I need to evolve a table’s partitioning at some point? If I do, do I want to avoid a rewrite of the table?
* Does the format enable an intuitive and easy to use SQL Syntax for creating and querying tables to avoid unnecessary full table scans?
* Is there a large and diverse developer community behind the project to avoid vendor lock -in and a potential future lack of support for tools and use cases that don’t suit dominant contributors?

These inquiries should assist you in future-proofing your data lake and supplying it with the cutting-edge capabilities offered by more recent table formats.

**Snowflake and Apache Iceberg**

The features of Snowflake's platform are extended to open formats, Apache Iceberg and Apache Parquet, in storage that is managed by customers thanks to a new type of Snowflake table known as an Iceberg Table. Without sacrificing the interoperability that an open table format offers, you may interact with Iceberg Tables in the same way that you would with any other Snowflake table, including being able to implement native column-level security.

Using a variety of file formats, such as Parquet, Avro, ORC, JSON, and XML, the Snowflake Data Cloud makes it simple to perform big data operations. Despite the fact that establishing a data lake, mesh, or other storage pattern for data that is directly stored in Snowflake is made much simpler by the internal tables of Snowflake, some businesses cannot or prefer not to keep all of their data in Snowflake due to regulatory or other restrictions. The Snowflake Data Cloud now offers Iceberg Tables and External Tables as two options to use Apache Iceberg. Iceberg Tables combine customer-managed cloud storage with the Snowflake tables' performance and well-known query semantics. Iceberg Tables are the best choice for use cases demanding complete DML, quick performance, several Snowflake platforms features, as well as data stored on external storage. When it comes to use cases that call for quick, read-only access to data that can't be moved from cloud storage but still needs to be governed and shared, external tables are ideal.

**Diagram

Description automatically generated**