upsolver

Whitepaper:

# Getting Data Lake ETL Right: 6 Guidelines for Evaluating Tools

Many organizations gravitate towards data lakes as a means to reduce friction and complexity in their IT infrastructure, and to store large volumes of data without the need for lengthy data transformation on ingest.

However, simply pouring all of your data into object storage such as Amazon S3 does not mean you have an operational data lake quite yet; to actually put that data to use in analytics or machine learning, you will need to build **ETL flows** that transform raw data into structured datasets you can query with SQL.

There are many different options for data lake ETL - from open-source frameworks such as Apache Spark, to managed solutions offered by companies like Databricks and StreamSets, and purpose-built data lake ETL tools such as Upsolver. When evaluating such tools, it's important to understand the unique

challenges of data lake ETL compared to traditional database ETL, and to choose a platform that will be able to address these specific hurdles.

Let's look at the top 6 factors you should consider when evaluating a data lake ETL platform - whether open-source, proprietary, or custom-developed.

*Building a data lake? Get some inspiration by checking out [4 Examples of Data Lake Architectures on Amazon S3](#).*

## 1. Ability to perform stateful transformations - ETL vs ELT

The ability to perform joins, aggregations and other stateful operations plays a crucial role when analyzing data from multiple sources, and is a basic feature available in traditional

ETL frameworks. However, in a decoupled architecture this core functionality is often difficult to implement.

In data warehousing, a common approach is to rely on an ELT (extract-load-transform) process in which data is sent to an intermediary database or data mart. Stateful transformations are then performed using the database's processing power, SQL and historical data already accumulated, before being loaded to the data warehouse table.

In a data lake, we wouldn't want to rely on a database for every operation, as that defeats the purpose of reducing costs and complexity by decoupling the architecture. When evaluating data lake ETL tools, make sure to choose a transformation engine that can perform stateful operations in-memory and support joins and aggregations without an additional database.

## 2. Support for evolving schema-on-read

Databases and SQL are designed around structured tables. However, data lakes are typically used as repositories for raw data in structured or semi-structured form (e.g. log data in JSON format).

Since it's impossible to query data without some kind of schema, data lake ETL tools need to be able to extract schema from raw data and to update it as new data is generated and the data structure changes. One specific challenge to keep in mind in this regard is the ability to query arrays with nested data, which many ETL tools struggle with.

## 3. Optimized object storage for improved query performance

A database optimizes its file system to return query results quickly, but trying to read raw data directly from a data lake will often result will result in terrible performance (up to

100-1000x higher latencies). Data needs to be stored in columnar formats such as Apache Parquet and small files need to be merged to the 200mb-1gb range in order to ensure high-performance, and these processes should be performed on an ongoing basis by the ETL framework you have in place.

Traditional ETL tools are built around batch processes in which the data is written once to the target database; a data lake ETL tools should write the data to the lake multiple times in order to continuously optimize the storage layer for query performance.

## 4. Integration with metadata catalogs

One of the main reasons to adopt a data lake approach is because we want to store large amounts of data now and decide how to analyze it later. Data lakes are meant to be

flexible and open in order to support a wide variety of analytics use cases (see Understanding Data Lakes and Data Lake Platforms).

A core element of this open architecture it to store metadata separately from the engine that queries the data. This makes it easy to replace query engines or to use multiple engines at the same time for the same copy of the data.

For example, we might use Hive Metastore or AWS Glue Data Catalog to store metadata, which we would then query using Apache Presto, Amazon Athena and Redshift Spectrum - with all queries running against the same underlying data.

A data lake ETL tool should support this open architecture by being closely integrated with the metadata catalog - so that metadata is both stored in the catalog and continuously synced with every change (location of objects, schema, partition), so that data remains easily queryable by various services.

# 5. Enabling ‘time-travel’ on object storage

One of the advantages of storing raw data in a data lake is the ability to ‘replay’ a historical state of affairs. This is hard to achieve in databases as they store data in a mutable state, which makes testing a hypothesis on historical impossible in many cases - e.g., if we choose to reduce costs by preprocessing or pruning the data before loading it into the database. Even when it is possible, the performance stress and costs of running such a query could make it prohibitive, creating tension between operations and exploratory analysis.

Data lakes are based on an event sourcing architecture where raw data is stored untouched. When a hypothesis presents itself, historical data is streamed from object storage for quick validation. Data lake ETL should reduce the friction of orchestrating such ad-hoc workloads and make it easy to extract a historical dataset, without creating large operational overhead.

## 6. Ability to update tables over time

While databases typically support updates and deletes to tables, data lakes are composed of partitioned files which predicates on an append-only model. This can create difficulty in storing transactional data, implementing CDC in a data lake, or deleting specific records for GDPR compliance.

Modern data lake ETL tools should provide the means to bypass this limitation by enabling upserts in the storage layer, as well as in the output tables being used for analytic purposes.

## Want to learn more about Data Lake ETL?

- Schedule a demo of Upsolver to see the power of modern data lake ETL in action.
- Check out our guide to improving the performance of Amazon Athena.
- Read a case study of how Sisense transforms 70bn records into usable data.