upsolver

Analytics-Ready Data Lakes: 6 Guidelines for Evaluating Data Integration 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, organizations find that simply pouring all of the 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, developers need to build **ETL flows** that transform raw data into structured datasets they 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 engineering tools such as Upsolver**. When evaluating such tools, it's important to understand the most material challenges of data lakes 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 companies should consider when evaluating a data transformation platform for your data lake - whether open-source, proprietary, or custom-developed.

1. Complex analytical transformations

Decoupled architectures provide numerous cost and agility advantages for a sophisticated data organization, but when considering crucial data analysis actions (joins, aggregations, and other stateful operations) options can be limited. These abilities play a crucial role when analyzing data from multiple sources, and are readily available in traditional ETL frameworks.

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, relying on a database for every operation defeats the purpose of reducing costs and complexity by decoupling the architecture. When evaluating data lake engineering 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 with a pre-defined schema. However, modern organizations and new data sources often generate large volumes of semi-structured data from streaming sources such as server logs, mobile applications, online advertising and connected devices.

Semi-structured data is often much greater in volume and by definition lacks a consistent schema or structure, making it difficult to query within a traditional data warehouse. This leads to much less data being actively stored and utilized. Hence, data lakes have become the go-to solution for working with large volumes of semi-structured or unstructured data, as they enable organizations to store virtually-unlimited amounts of raw data in its original format.

However, it's impossible to query data without some kind of schema – and hence data transformation 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, in order to continuously make the data available for querying in analytics tools. 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, but can often be seen in sources such as user app activity.

3. Optimizing object storage for improved query performance

Optimizing query performance is crucial in order to ensure data is readily available to answer business-critical questions, as well as to control infrastructure costs. A database optimizes its file system to return query results quickly – but this advantage is missing in 'vanilla' data lake storage, where data is stored as files in folders on cloud object storage such as Amazon S3.

Trying to read raw data directly from object storage will often result in poor 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 in place.

Traditional ETL tools are built around batch processes in which the data is written once to the target database; with continuous event streams stored in a data lake, this approach is typically inadequate. Hence, organizations should opt for data transformation tools that 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 to be able 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 is to store metadata separately from the engine that queries the data. This provides a level of data engine agnosticism which 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, Hive Metastore or AWS Glue Data Catalog can be used to store metadata, which can then be queried using Apache Presto, Amazon Athena and Redshift Spectrum - with all queries running against the same underlying data.

With this in mind, it is essential that any selected data integration tool should support this open architecture working with the metadata catalog – both storing and continuously synchronizing metadata with every change (location of objects, schema, partition) – so that data remains easily accessible 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 engineering tools should reduce the friction of orchestrating such ad-hoc workloads and make it easy to extract a historical dataset, without creating large operational overhead. This is achieved by ensuring multiple copies of the data are stored according to a predetermined retention policy, while data lineage, schema changes and metadata is kept for the purpose of retrieving a previous state of the data.

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 to address regulatory concerns. In order to comply with GDPR or CCPA requests, organizations must be able to accurately retrieve and delete any copy of PII stored on their infrastructure.

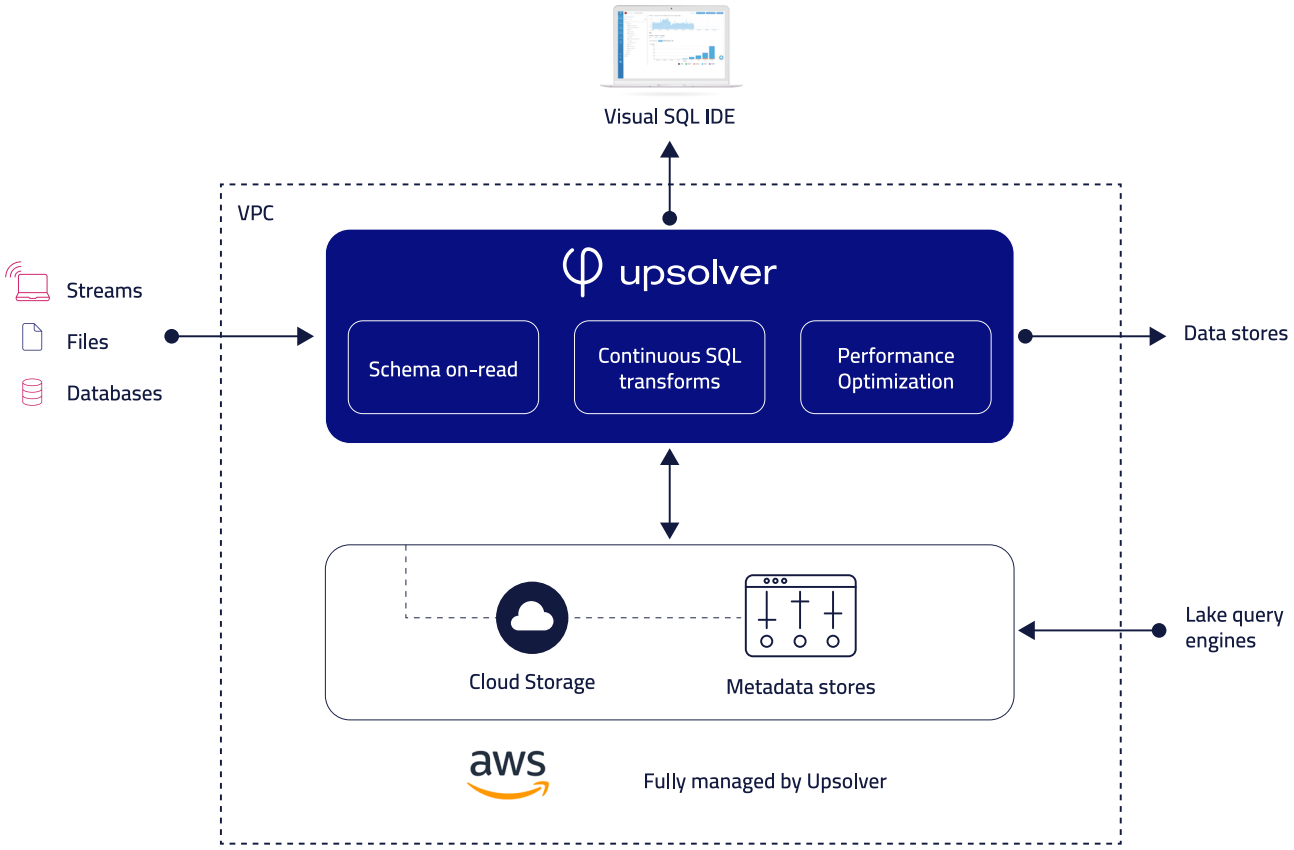
Since data lake storage is not indexed, finding a particular record can be a ‘needle in a haystack’ situation wherein a full table scan is required, which could impact performance and costs and hinder the ability to respond to GDPR / CCPA requests in a timely fashion.

Modern data lake engineering 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. In order to do so, these tools can leverage in-memory indices or key-value stores that enable developers to access and update records in object storage.

Upsolver: Self-service Data Lake Engineering at Scale

As cloud data lake adoption increases, organizations are increasingly looking beyond traditional code-intensive solutions for data processing and integration. In order to reap the benefits of modern cloud architecture, it is essential to reduce the friction of operationalizing data, while ensuring each of the key capabilities described above is met.

Upsolver provides an easy-to-use data lake engineering platform that enables data-intensive organizations to make their data lake analytics-ready 20x faster than alternative solutions. By providing a self-service, high-speed compute layer between your data lake and the analytics tools of your choice, Upsolver automates menial data pipeline work, so companies can focus on developing analytics and real-time applications.



To learn more about Upsolver:

- **Schedule a demo of Upsolver** to see the power of no-code data lake engineering in action.
- Get started with the free-forever **Upsolver Community Edition** to build working data lake engineering solutions in minutes
- Read a case study about how Upsolver helped **Sisense transforms 70bn records into usable data**.

Start now on the AWS Marketplace

Upsolver is an Amazon Web Services Advanced Technology Partner. Data-driven companies such as ironSource, Wix, and The Meet Group use Upsolver to power data lakes and deliver batch and stream processing at unparalleled ease.

Sign up today in AWS Marketplace

aws marketplace