Eckerson Group | White Paper

The Definitive Guide to the SQL Data Lakehouse

Must-Have Characteristics and Components

BY KEVIN PETRIE

FEBRUARY 2022

RESEARCH SPONSORED BY DREMIO

dremio

About the Author

For 25 years **Kevin Petrie** has deciphered what technology means to practitioners, as an industry analyst, writer, instructor, marketer and services leader. Kevin launched, built and led a profitable data services team for EMC Pivotal in the Americas and EMEA, and ran field training at the data integration software provider Attunity (now part of Qlik). A frequent public speaker and author of two books on data streaming, Kevin also is a data management instructor at eLearningCurve.

About Eckerson Group

Eckerson Group is a global research and consulting firm that helps organizations get more value from data. Our experts think critically, write clearly, and present persuasively about data analytics. They specialize in data strategy, data architecture, self-service analytics, master data management, data governance, and data science. Organizations rely on us to demystify data and analytics and develop business-driven strategies that harness the power of data. **Learn what Eckerson Group can do for you!**

About This Report

This report is sponsored by Dremio, who has exclusive permission to syndicate its content.

Table of Contents

Executive Summary

The SQL Data Lakehouse is a type of cloud data architecture that queries object stores at high speed and provides access to data across multiple sources to support both business intelligence (BI) and data science workloads. Enterprises adopt the SQL Data Lakehouse to streamline their architectures, reduce cost, and simplify data governance. Common use cases include reporting and dashboards, ad-hoc queries, 360-degree customer views, and artificial intelligence/machine learning (AI/ML).

This report explores the architectural components that make the SQL Data Lakehouse unified, simple, accessible, fast, economic, governed, and open. These components span the object store, a data layer, processing layer, semantic layer, communication layer, and client layer. Data teams that select the right components for their environments and establish the right points of integration can modernize their data architecture for analytics and BI.

Take the following steps to build and execute the right strategy to modernize your open data stack.

- ***Identify which use cases matter most to your business users*** *and which of those are hard to support with your current data warehouse as you scale.*
- ***Make sure you need the SQL data lakehouse.*** *Are you struggling to meet BI performance requirements, govern multiple data copies, or integrate data across multiple repositories?*
- ***Design your architecture with must-have characteristics in mind.*** *This means assembling and integrating the right components across your environment—do not look for a standard approach, or assume you need every bell and whistle.*

Rise of the SQL Data Lakehouse

Enterprises have a new option to unify the worlds of business intelligence and data science: the SQL Data Lakehouse.

The SQL Data Lakehouse is a type of cloud data architecture that uses structured query language (SQL) commands to query cloud object stores at high speed. It also provides a semantic layer that consolidates virtual views of the underlying physical data. The SQL Data Lakehouse supports both business intelligence (BI) and data science workloads because it runs SQL queries against both relational data and multi-structured data stored as files.

Enterprises adopt the **SQL data lakehouse** to simplify how they meet exploding business demand for analytics. They see **three primary benefits.**

- By running fast queries directly on the object store within the data lake, enterprises no longer have to copy or move data to meet BI performance requirements. This eliminates the need for data extracts and a data warehouse, which simplifies architectures and reduces cost.
- By consolidating data views into a semantic layer, enterprises simplify data access and governance. Data analysts and data scientists can prototype new analytics approaches without having to copy or move data. They still maintain an open architecture and open formats to interoperate with other tools.
- By supporting both BI and data science, the SQL Data Lakehouse enables enterprises to consolidate workloads. They no longer need two distinct platforms, e.g., a data warehouse for BI and data lake for data science.

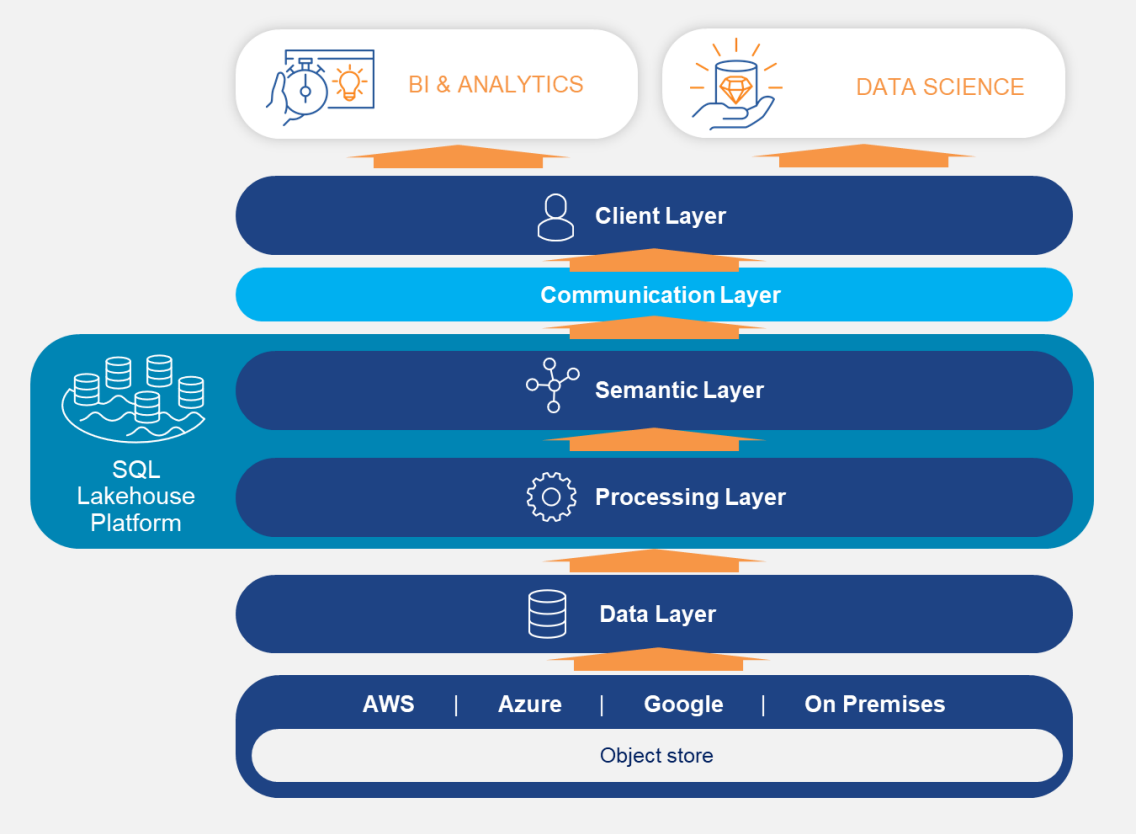
By running fast queries directly on the object store, enterprises no longer have to copy or move data to meet performance requirements

The SQL Data Lakehouse is a new type of architecture. It is not a database, data warehouse, or data lake that stores and transforms physical data. Rather, it is an extension of a data lake that has data warehousing capabilities. Most enterprises use the SQL Data Lakehouse to query an object store in a data lake. The SQL data lakehouse differs from other types of cloud data architectures as well. Unlike the **Snowflake Data Cloud** or **Azure Synapse Analytics,** the SQL data lakehouse queries data directly on the object store, wherever it resides in the cloud or on-premises.

Architectural Overview

The SQL lakehouse includes six architectural layers.

Figure 1. The SQL Data Lakehouse



- **The object store,** as the name suggests, comprises multi-structured data objects that reside in hybrid, cloud, or multi-cloud infrastructure.
- **The data layer** includes table formats, file formats, and metastores that centralize metadata.
- **The processing layer** includes functions, such as columnar processing, parallel processing, and caching, which help optimize queries.
- **The semantic layer** combines virtual data views with a catalog, lineage, and other functions that help govern data.
- **The communication layer** comprises APIs such as REST or ODBC, as well as communication frameworks that client tools use to interact with the semantic layer.
- **The client layer** includes BI products, machine learning libraries, and other tools that analysts use to manipulate and visualize query results.

The SQL data lakehouse also exports data views and query results to data science platforms. Vendors that support the SQL data lakehouse approach include **Dremio** and Starburst. They bundle

the semantic and processing layers into products we can categorize as SQL Lakehouse Platforms. These platforms currently focus mostly on BI use cases, given the popularity of SQL with data analysts. As data scientists grow more familiar with SQL, they will drive adoption of SQL Lakehouse Platforms.

Databricks also offers a SQL lakehouse platform. However, its approach differs from Dremio and Starburst because its Delta Lake maintains multiple physical copies of the same data, including raw and transformed copies. Databricks also differs because it comes from a data lake heritage and therefore has a higher portion of data science users.

Figure 1 illustrates the architectural layers of the SQL Data Lakehouse.

Designing an effective SQL data lakehouse and integrating it into a heterogeneous enterprise environment is no easy task. This report explores use cases and case studies, then defines the must-have characteristics and architectural components of the SQL data lakehouse.

Case Studies

Enterprise data teams rely on the SQL data lakehouse to address a variety of business scenarios. Common use cases include reporting and dashboards, 360-degree customer views, self-service analytics, and artificial intelligence/machine learning (AI/ML). Here are five real-life examples.

Reporting and dashboards: Henkel

The consumer packaged goods firm Henkel needed to synchronize planning and analytics across a global supply chain, but struggled to generate the right insights on a timely basis. To right the ship, they implemented a SQL lakehouse platform from Dremio on Microsoft **Azure Data Lake Storage** (ADLS). This new platform eliminated duplicative data silos, improving data quality, and accelerated performance with caching and columnar processing. Henkel's business analysts and data analysts now monitor interactive dashboards in Tableau and answer queries in seconds. They improved shipment decisions and supply chain efficiency.

360-Degree customer views: AP Intego

AP Intego (now part of Next Insurance) offers a digital platform that recommends insurance plans to small businesses. To enable 360-degree views of each customer, their data team centralized sales records, service records, and affinity partner insurance plans in a Dremio SQL lakehouse platform on an Amazon S3 data lake. AP Intego now accesses customer profiles through Google **Looker** and recommends plans to prospects in real time, improving sales win rates.

Self-service analytics: Vitesco Technologies

Vitesco Technologies, builder of hybrid and electric car components, needed to boost productivity by analyzing millions of data points about their manufacturing process. They implemented a SQL data lakehouse platform on Amazon S3 to provide data analysts and engineers flexible access to one consolidated copy of the data. This architecture improved their security and governance controls, reduced data engineering work, and increased overall efficiency.

Self-service analytics: NCR

NCR Corporation software and services help companies run effective operations. To streamline their quote-to-cash offerings, they need to streamline how they derive supporting metrics from their Hadoop data lake. By deploying a SQL data lakehouse platform on top of the Hadoop file system, NCR delivered real-time views to Tableau dashboards and accelerated data engineering with pipeline automation and reusability. NCR created new analytics value while improving productivity.

AI/ML: Knauf

Knauf builds insulation products that make homes, buildings, and ships more resilient. Their data team analyzes hundreds of millions of data points from factory equipment sensors to ensure product quality. To support fast queries, they implemented a SQL lakehouse platform on a consolidated ADLS data lake and ancillary SQL databases. Data scientists feed query results into the Anaconda data science platform to detect anomalies for preventive maintenance. This makes manufacturing more efficient.

7 Must-Have Characteristics of the SQL Data Lakehouse

These use cases require a SQL data lakehouse with certain must-have characteristics. It must be unified, simple, accessible, high performance, economic, governed, and open. Let's explore these seven characteristics before defining the architectural components.

Unified

The SQL data lakehouse must support both BI and data science use cases. For example, a data scientist might devise an ML model that predicts market prices and give that model to the data analyst so they can forecast future revenue scenarios. This requires a unified repository for data engineers, data analysts, and data scientists. They need to share views of data from sources such as operational databases, applications, IoT sensors, and social media feeds. They need to run multiple concurrent workloads on the same copy of data and share tools and outputs with one another.

Data engineers, data analysts, and data scientists need to share views of data.

Simple

The SQL data lakehouse should automate the configuration and management of its various components to help data teams execute projects with less effort. This includes a graphical interface that helps data engineers and data analysts discover, transform, curate, and query data. They should be able to peruse files within the data store, select one, and preview its contents before filtering or reformatting it for analytics—all via mouse clicks, drop down menus and popup windows rather than manual scripting. A managed service can further simplify things by minimizing software implementation and administration work.

Accessible

The SQL data lakehouse should enable data analysts and data scientists to access data themselves rather than relying on data engineers. Self-service like this requires a catalog with intuitive views of metadata, including file attributes, lineage, and usage history. In addition, data analysts and data scientists need to access these data views without tripping over one another. That is, they need to create consistent data views that rely on the same underlying physical copy of data. A finance report might tabulate paid invoices to measure quarterly revenue and a sales report might tabulate signed contracts to measure bookings. Those reports must derive their distinct numbers from the same consistent records.

High performance

The SQL data lakehouse should meet rigorous Service Level Agreements (SLAs) for key performance metrics. These include low latency to support short query response times, high throughput to query high volumes of data, and high concurrency to support many workloads. A real-time dashboard for retail sales on Cyber Monday might require low latency to help sales leaders adjust market prices. A root-cause analysis of manufacturing defects, meanwhile, might require high throughput to analyze sufficient volumes of IoT sensor data. And both these projects might need to serve many concurrent users.

Economic

Like any other cloud data architecture, the SQL data lakehouse needs to help enterprises control cost by using resources wisely. It should profile workloads prior to execution so users know how many compute cycles they will require, then automatically adjust processing methods along the way to streamline those workloads. It should provide the ability to scale elastic resources up or down, consuming storage

and compute capacity as needed to meet changing workloads requirements. And that scalability must be economic. Like a hybrid car that goes quiet at the stoplight, the SQL data lakehouse should avoid unnecessary compute cycles. Enterprises need economic capabilities like these because most analytics projects, ranging from quarterly financial reports to ad-hoc customer 360 analyses, entail bursts of processing.

Governed

Enterprises must govern data usage in order to reduce risks to data quality and ensure compliance with regulations such as the **General Data Protection Regulation** (GDPR), **California Consumer Privacy Act,** and **Health Insurance Portability and Accountability Act** (HIPAA). This means avoiding unnecessary data copies that might undermine a "single source of truth." It means controlling user actions with role-based access controls, masking sensitive data, and tracking lineage. Finally, it means recording user actions in a comprehensive audit log. Guardrails like these enable data analysts to generate accurate reports and compliance officers to protect personally identifiable information (PII).

Open

The SQL data lakehouse should integrate with the ecosystem of data stores, formats, processors, tools, APIs, and libraries that modern data teams need to innovate. It also should complement and interoperate with alternative cloud data architectures, such as Snowflake or Azure Synapse Analytics. Data teams need an open architecture in which they can change these elements as business requirements evolve. They should not have to write custom scripts in order to move data between object stores, switch processing engines, or import an ML algorithm.

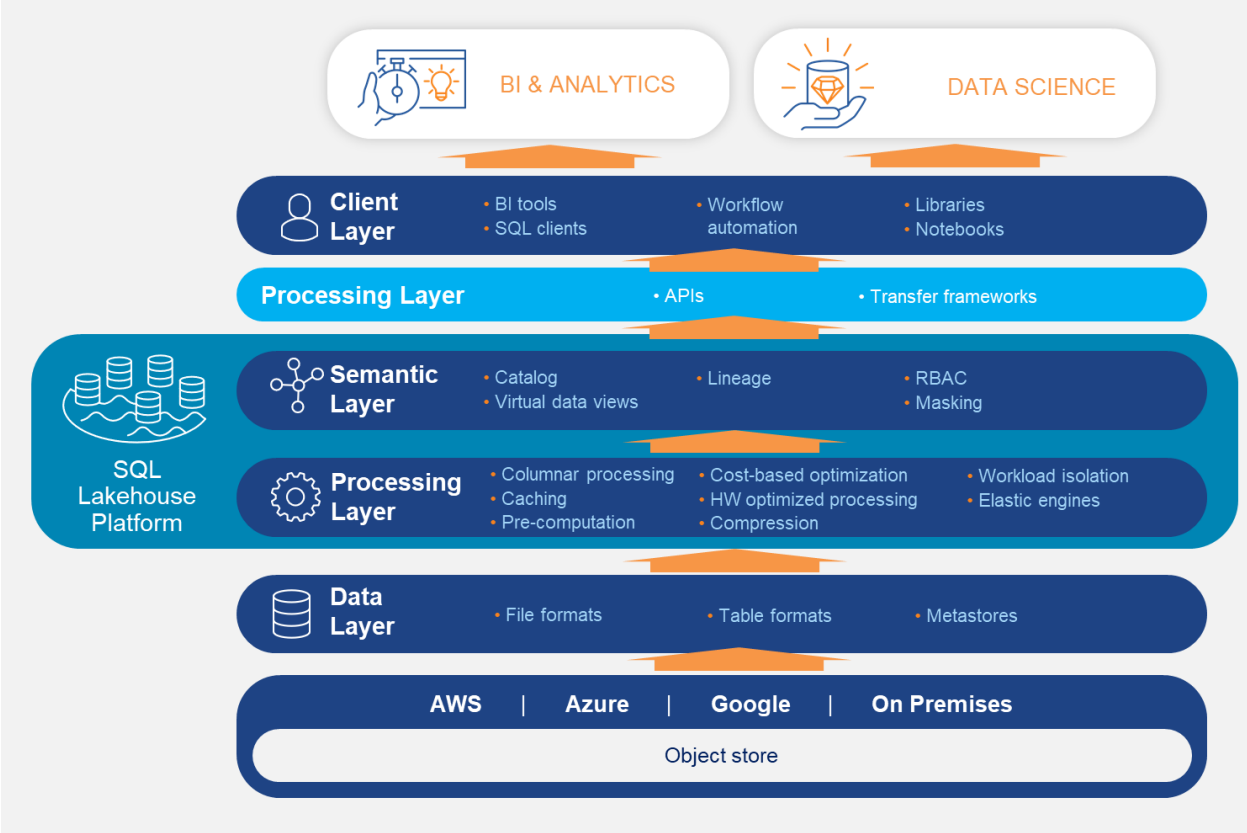
An open architectural approach helps data analysts take iterative steps and provides them with the flexibility to bring multiple engines and their choice of tools directly to the data, leveraging open formats, and not the other way around. This minimizes the need for complex, insecure, or risky data moves and data copy proliferation. For example, analysts might enrich their BI projects with new datasets, query the enriched dataset, and then see whether they can speed performance by swapping in a different query engine—all while retaining the governance integrity of the datasets.

The SQL data lakehouse should integrate with the ecosystem of data stores, formats, processors, tools, APIs and libraries that modern data teams need to innovate.

Architecture and Components

Now let's define the common architectural elements of a SQL data lakehouse, including the SQL lakehouse platform, and ways to ensure they support those must-have characteristics. We'll work from the bottom up in our architecture diagram.

Figure 2. Architecture of the SQL Data Lakehouse



Object Store

This stores all types of data objects—tables, flat files containing plain text, images, you name it—and classifies them for querying with identifiers and metadata. Object stores find a natural home on the cloud, whose elastic storage capacity enables enterprises to scale up or down as needed. Many enterprise data teams now favor object stores such as Amazon Simple Storage Service (**Amazon S3**), **Azure Blob Storage,** and **Google Cloud Storage** because they offer a unified and economic repository for structured, semi-structured, and unstructured data. Enterprise data teams should consider consolidating on cloud object stores as part of any data modernization initiative, provided they select open formats to avoid lock-in.

Data Layer

File formats

Many common file formats, such as comma-separated values (CSV), optimized row columnar (ORC) (for Apache Hive data), **Apache Parquet,** Avro, and JavaScript object notation (**JSON**), interoperate with various tools, processors, and data stores to enable an open architecture. By selecting these formats, enterprise data teams can maintain data portability and flexibility to adapt with the needs of the business. They should avoid converting to proprietary formats that cloud infrastructure providers recommend. While those proprietary formats might enable platform-specific enhancements such as higher performance, they risk locking enterprises into that provider's infrastructure, thereby reducing future portability.

Table formats

The **Apache Iceberg** open table format helps the SQL data lakehouse run fast, concurrent queries on large data volumes. Iceberg maintains metadata that helps datasets evolve and support multiple workloads without having them interfere with one another. For example, it adapts to schema changes such as added or dropped columns. Iceberg partitions data, and updates those partitions as datasets evolve, to help speed queries by avoiding unnecessary reads. It also enables time travel so users can roll back versions to correct issues or run queries on historical snapshots. For example, a data analyst might need to compare query results for 2019 and 2020 to gauge the impact of COVID on customer spending habits.

The Apache Iceberg open table format helps the SQL data lakehouse run fast, concurrent queries on large data volumes.

Delta Lake, also an open-source table format, provides similar capabilities to enable fast, concurrent query workloads on a large dataset. Delta Lake maintains transactional consistency, supports schema evolution, and enables time travel. **Apache Hudi** also offers similar capabilities, along with stream processing and data pipeline management. While each of these open table formats has a slightly different workload focus, enterprise data teams can choose any of them today and still swap out in the future. These open-source table formats help make the SQL lakehouse unified, accessible, and fast.

Metastores

The metastore centralizes the metadata that data engineers need to ingest, transform, and reconcile different versions of data to ensure data quality. For example, the open-source **Project Nessie** offers a metastore that helps data engineers apply DataOps techniques such as code branching and version control to datasets. Based on the metadata Nessie centralizes, data engineers can view, branch, and

merge datasets in an approach akin to GitHub for software development. Nessie supports "extract, transform, and load" (ETL) or ELT sequences for both batch and streaming data pipelines.

Project Nessie centralizes metadata to help data engineers apply DataOps techniques, such as code branching and version control to datasets.

The **AWS Glue** data pipeline also offers a metastore to help data engineers organize and view data so they can ingest and transform it in ETL pipelines that feed Amazon S3 data lakes. In addition, the **Apache Hive** metastore can find and manage metadata for similar purposes in a Hive data warehouse. Of these options, Nessie and Hive metastores offer the more open approaches. Metastores help simplify data management and make data accessible to various users while still governing it.

SQL Lakehouse Platform

Now we enter the SQL lakehouse platform, the category of products that bundle together the processing and semantic layers.

Processing Layer

The processing layer includes a mix of functions that focus precious compute cycles on the data that matters for a given query. The price of cloud compute makes streamlined processing mandatory for any hybrid, cloud, or multi-cloud environment. Each SQL lakehouse platform offers enterprises many or all of the following capabilities.

- **Columnar processing.** By arranging tabular data in columns rather than rows, the SQL lakehouse platform can read only the attributes that relate to a query. Many enterprises use **Apache Arrow,** for example, to support columnar processing. The **Apache Spark** distribution now includes Arrow, as do the **MATLAB** data science tool and **Dremio** SQL lakehouse platform. Other types of cloud data architectures, such as **Google BigQuery** and **Vertica,** also support columnar processing.
- **Caching.** Some platforms identify frequently-queried tables or records and pre-fetch those into cache to prepare for the next query.
- **Query pre-computation.** Some platforms prepare for frequent queries by pre-assembling columns and pre-computing aggregate values.
- **Workload isolation.** Isolating a workload on one or more compute nodes minimizes resource contention, making that workload faster and more predictable.

- **Elastic engines.** The platform should provision flexible "engines," or allotments of parallel compute nodes that scale up or down to support a workload within configurable thresholds.
- **Cost-based optimization.** By profiling workloads and automatically adjusting how it handles tasks—such as the order in which it joins tables—the platform can streamline compute cycles and thereby reduce costs. It also should measure these cycles by workload and user group to support chargeback.
- **HW optimized processing.** Some platforms optimize tasks such as data sorting and filtering for specific runtime environments. For example, Apache Gandiva helps do this for highly parallel workloads on modern CPUs.
- **Compression.** By modifying data structures to consume less disk space, compression frees up valuable capacity, which improves efficiency and reduces cost.

Capabilities like these enable enterprises to maintain or improve query response times while restricting the number of on-premises servers or cloud compute nodes they need underneath. Data teams should seek out and apply these capabilities to ensure they meet business needs for use cases such as rapid reporting and ad-hoc querying of large data volumes. They need a SQL lakehouse platform that is both fast and economic.

Semantic Layer

The semantic layer of the SQL lakehouse platform presents business-oriented data views to data analysts and data scientists. It consolidates these views within the platform rather than relying on semantic layers within BI tools, helping ensure everyone has the same version of the truth. The semantic layer includes a catalog, virtual data views, lineage, role-based access controls, and data masking.

- **Catalog.** The catalog centralizes metadata to create an indexed, searchable inventory of data assets, including tables, files, schema, and queries. This enables data analysts to find the assets they need, while still helping data curators and compliance officers manage data quality and control data access.
- **Virtual data views.** Because users often need many different views of the same physical dataset, the semantic layer presents virtual views that are transformations of physical data underneath.
- **Lineage.** The lineage of a dataset describes its origins, who touches it, and how it changes over time, so users can validate data quality and identify the cause of issues.
- **Role-based access controls.** Like any enterprise software platform, the SQL lakehouse platform authenticates the identities of users and authorizes the actions they take, according to user type, data asset, and category of action. It also integrates with third-party identity management tools.
- **Data masking.** Finally, the semantic layer masks sensitive data such as customers' personally

identifiable information (PII). Masking replaces selected data with dummy values that support queries while still preventing bad actors from compromising real data.

These capabilities of the semantic layer make the SQL lakehouse platform more unified, accessible, and governed. Data engineers can transform and organize multi-sourced data in the object store. Data analysts can serve themselves by creating virtual data views and running queries to feed their BI projects. Data stewards and compliance officers, meanwhile, can control and oversee teams' activities. Data teams should be sure to implement capabilities like these, within either the SQL lakehouse platform or a larger enterprise governance offering such as **Privacera** or **BigID.**

To interact with various client tools, the SQL lakehouse platform relies on a separate communication layer.

Communication Layer

Application programming interfaces (APIs)

Data analysts, data scientists, and data engineers access data through common APIs, such as representational state transfer (REST), SQL, Java database connectivity (JDBC), and open database connectivity (ODBC). For example, a data analyst might use the REST API to connect their BI tool to the SQL data lakehouse, submit SQL queries, and view results. A data scientist, meanwhile, might use a JDBC API to export query results from the SQL data lakehouse to a data science platform. In addition, various other components, such as Amazon S3 and Apache Iceberg, use their own APIs to interoperate with one another.

Enterprises should seek out APIs like these in order to make their SQL data lakehouse open and accessible. They should avoid proprietary APIs—for example, from cloud infrastructure providers—that offer provider-specific enhancements but reduce data portability. There is also the risk of "creeping lock-in," whereby vendors offer APIs that become less open over time.

Data transfer frameworks

Data teams also need to maintain performance levels as they apply various processors to a large dataset or transfer high volumes of data between repositories. Data transfer frameworks, most notably **Apache Arrow Flight,** can help. Arrow Flight offers a high-speed alternative to APIs such as ODBC and JDBC by eliminating the need to serialize and deserialize data as it passes between architectural components. For example, a data analyst might use Arrow Flight to present higher volumes of data to their BI tool or SQL client than might otherwise be possible. A data scientist, meanwhile, might use Arrow Flight to transfer higher volumes of data from the SQL data lakehouse to a data science platform such as **Domino Data Lab** or **DataRobot.** The communication framework, especially Arrow Flight, helps unify and accelerate the SQL data lakehouse.

Arrow Flight offers a high-speed alternative to APIs such as ODBC and JDBC by eliminating the need to serialize and deserialize data as it passes between components of a data architecture.

Client Layer

Tools for BI projects

Data teams access and manipulate data for BI projects using three types of client tools. First, data analysts and some data scientists use BI tools such as **Tableau, Looker,** and **Qlik** to query the virtual data views within the semantic layer and visualize the results. Second, data analysts, developers, and data scientists use SQL client tools such as **DbVisualizer** and **DataGrip** to write and manage SQL commands. Third, developers use workflow tools such as **Airflow** to orchestrate analytical and operational workflows that integrate with query results.

A given BI project might include all three types of tools. A developer might write a complex SQL query in DbVisualizer and pass it over to a data analyst who runs that query and visualizes the results in Tableau. They feed the results back to the developer, who integrates them into application workflows using Airflow. This type of scenario would support various use cases, such as the personalization of web pages for customers based on their most recent purchases and social media posts.

Tools for data science projects

Data scientists use two types of client tools as they manage data science projects. First, they download algorithms from AI/ML libraries such as **PyTorch** or **TensorFlow,** or use tools within those libraries to develop and train models. Second, they transform data, visualize data, and develop models in AI/ML notebooks such as **Jupyter.** AI/ML libraries and notebooks must both integrate easily with the SQL data lakehouse as data scientists move code or data back and forth. Data science projects, iterative by nature, need these tools to interoperate and exchange data with minimal effort.

Defining Your Approach

The SQL data lakehouse can simplify how enterprises meet business demands, foster analyst collaboration, and reduce effort. Data teams that select the right components for their environment and establish the right points of integration can start to redefine how their business uses analytics. They can support new and existing use cases on time and within budget. Of course, this means building and executing on the right plan. Take the following steps to increase your odds of success.

- **Prioritize your use cases.** Define the highest priority use cases for your cloud data architecture, using the categories described in this report as a reference. Most enterprises need to support at least one use case in each category: periodic reports; interactive reports and dashboards; ad-hoc queries, 360-degree customer views; and artificial intelligence/machine learning (AI/ML). Data teams need to identify which use cases matter most to their business users and which of those are hard to support with their current data warehouse.
- **Make sure you need the SQL data lakehouse. I**f your data engineers and data analysts struggle to meet BI performance requirements, govern multiple data copies, or integrate data across multiple repositories, the SQL data lakehouse might make sense for your enterprise. But before leaping to a SQL data lakehouse, determine whether you can reduce these points of pain with your current data warehouse.
- **Design with your must-have characteristics in mind.** If you opt for a SQL data lakehouse, your next step is to figure out how to make it unified, simple, accessible, high performance, economic, governed, and open. This means assembling and integrating the right components across your environment—the data and processing layers within your platform, the data layer they run on, and the client layer they support. Do not look for a standard approach, or assume you need every bell and whistle.

About Eckerson Group

Wayne Eckerson, a globally-known author, speaker, and consultant, formed **Eckerson Group** to help organizations get more value from data and analytics. His goal is to provide organizations with expert guidance during every step of their data and analytics journey.

Eckerson Group helps organizations in three ways:

- **Our thought leaders** publish practical, compelling content that keeps data analytics leaders abreast of the latest trends, techniques, and tools in the field.
- **Our consultants** listen carefully, think deeply, and craft tailored solutions that translate business requirements into compelling strategies and solutions.
- **Our advisors** provide one-on-one coaching and mentoring to data leaders and help software vendors develop go-to-market strategies.

Eckerson Group is a global research and consulting firm that focuses solely on data and analytics. Our experts specialize in data governance, self-service analytics, data architecture, data science, data management, and business intelligence.

Our clients say we are hard-working, insightful, and humble. It all stems from our love of data and our desire to help organizations turn insights into action. We are a family of continuous learners, interpreting the world of data and analytics for you.

Get more value from your data. Put an expert on your side. **Learn what Eckerson Group can do for you!**

Eckerson Group

RESEARCH CONSULTING ADVISORY

GET • MORE • VALUE • FROM • YOUR • DATA

About Dremio

Dremio is the SQL Lakehouse Platform company, enabling companies to leverage open data architectures. Dremio's SQL Lakehouse Platform simplifies data engineering and eliminates the need to copy and move data to proprietary data warehouses or create cubes, aggregation tables and BI extracts, providing flexibility and control for data architects and data engineers, and self-service for data consumers. Founded in 2015, Dremio is headquartered in Santa Clara, CA. Investors include Cisco Investments, Insight Partners, Lightspeed Venture Partners, Norwest Venture Partners, Redpoint Ventures, and Sapphire Ventures. For more information, visit www.dremio.com. Connect with Dremio on GitHub, LinkedIn, Twitter, and Facebook.