Data Lakehouse vs Data Warehouse vs Data Lake - Comparison of data platforms

Mariusz Kujawski · Follow
16 min read · Jul 24

262 4

For decades, data warehouses have been the dominant architectural approach for building data platforms in enterprises. However, with the advent of technologies such as Cloud, Big Data, and Hadoop the evolution of modern data platforms has accelerated, leading to the emergence of various options such as the data lake, and the data lakehouse.

According to articles published by leading cloud providers, the data lakehouse represents a new generation of data platforms. But the questions every data platform architect should be asking themselves are: Is data lakehouse the ideal architecture for my particular use case? Or should I opt for a data lake, or data warehouse?

In this post, I will explore the differences between these architectures and analyze which works best in which scenarios.

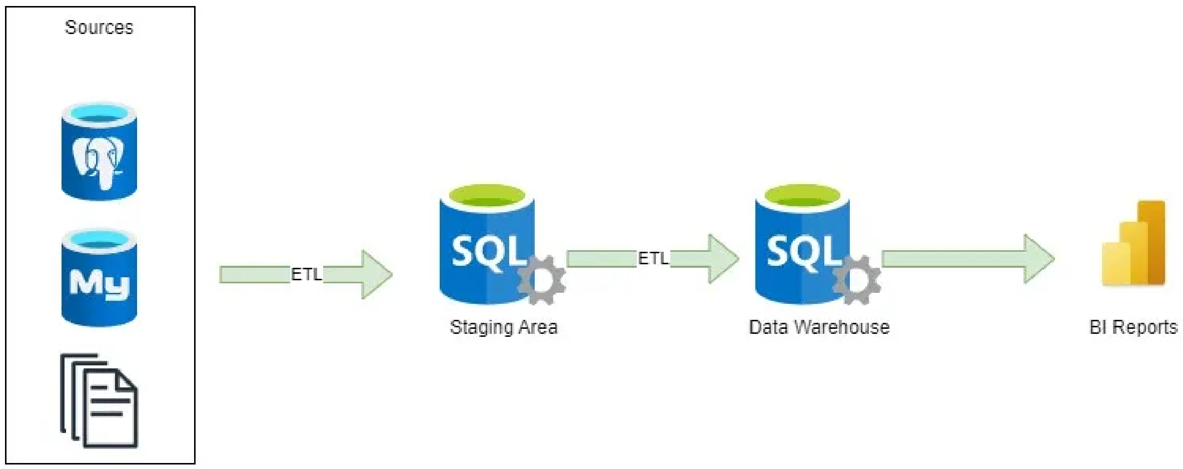
Data Warehouse

The Data Warehouse architecture (DW, DWH), aka Enterprise Data Warehouse (EDW), has been a dominant architectural approach for decades. A data warehouse serves as a central repository for structured business data, enabling organizations to gain valuable insights. It is important to define the schema before writing data into the warehouse. Typically, the data warehouse is populated through batch processing and consumed by BI tools and ad-hoc queries. The design of a data warehouse is specifically optimized for processing BI queries, although it cannot handle unstructured data. It consists of tables, constraints, keys, and indexes that support data consistency and enhance performance for analytical queries. The tables are often divided into dimensional and fact tables to further improve performance and utility.

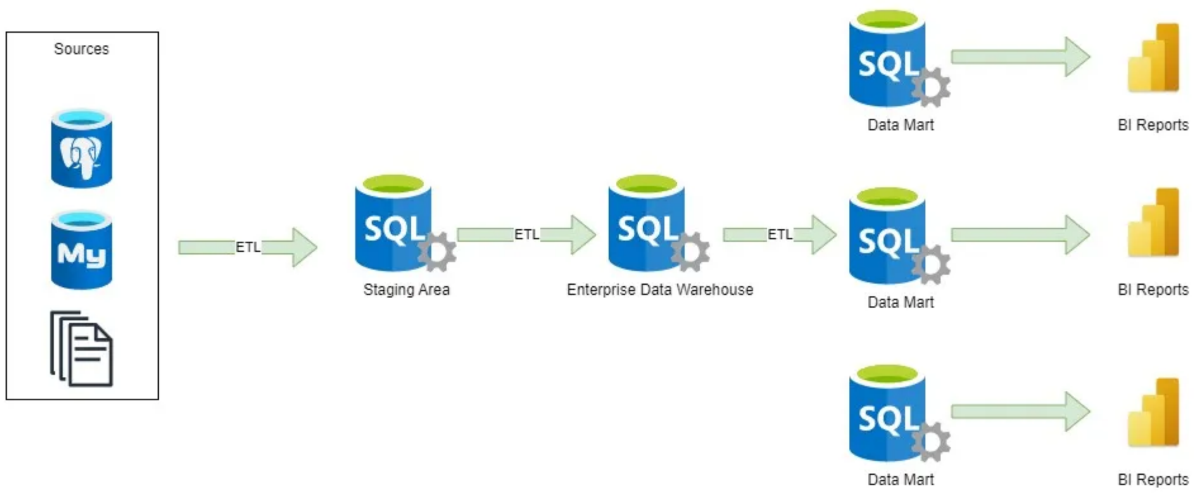
The common design of data warehouses incorporates a staging area where raw data is extracted from various sources using ETL tools like Informatica PowerCenter, SSIS, Data Stage, Talend, and more. The extracted data is then transformed into a data model within the data warehouse, either using an ETL tool or SQL statements. There are several architectural approaches to design a data warehouse, including the Inmon, Kimball, and Data Vault methodologies.

Data Warehouse design patterns

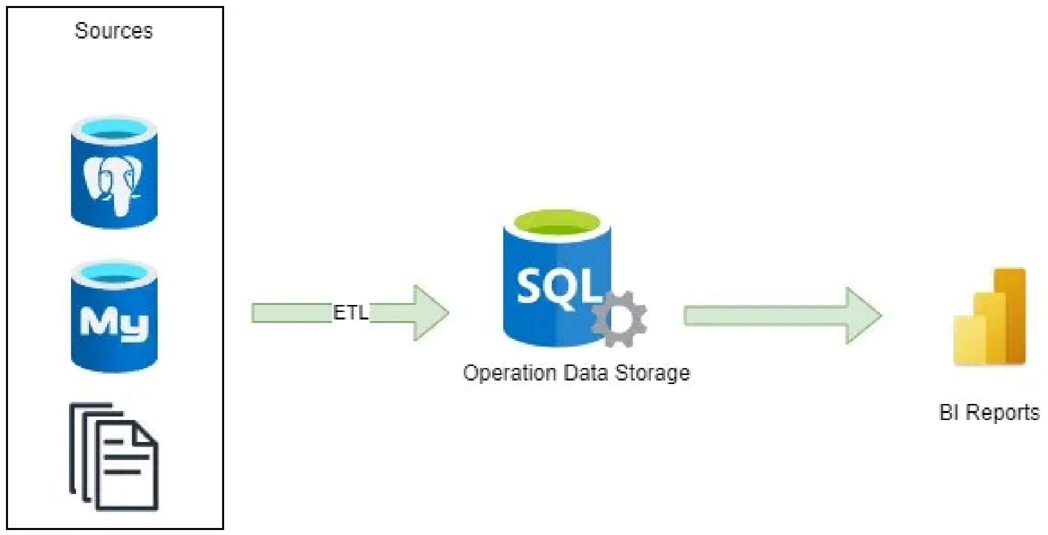
Now let's examine some different data warehouse implementation patterns. There are several architectural approaches to design a data warehouse. In this post, we will provide a brief overview of these architectures.



In the context of an enterprise data warehouse, where multiple departments have distinct reporting requirements, the implementation of a data mart becomes necessary. A data mart, as defined, is a data model specifically optimized to supply to the unique needs of specific business domains or departments. For example, the reporting needs of the marketing department will significantly differ from those of the accounting department.



Another essential component of the data platform was the Operation Data Storage, specifically designed to store the most recent data from a transactional system. Its primary distinction from the Data Warehouse (DWH) lies in the absence of historical data. The Operation Data Storage is intended to facilitate more frequent imports of data from source systems.



Modern Data Warehouse

Do we need the traditional data warehouse in a modern data platform? The answer is: it depends.

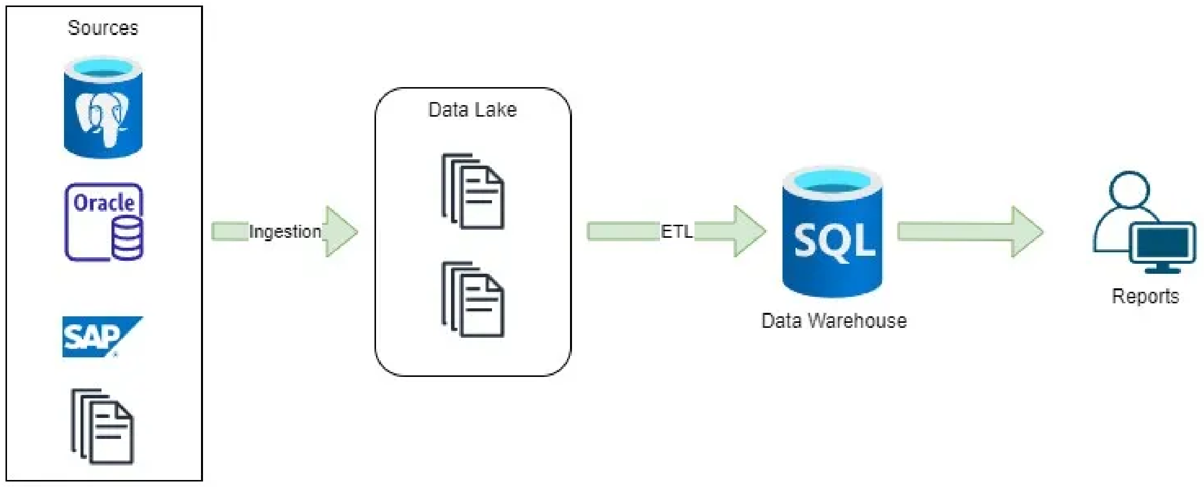
If we are developing a platform for a bank or an insurance company where regulatory reporting or utilization of BI tools is crucial, the data warehouse remains the optimal approach. It ensures that data is structured, prepared, and optimized for these specific purposes.

Technologies such as Azure Synapse, Redshift, BigQuery, and Snowflake enable the creation of a standard data model that can be utilized as a data warehouse. Integration with popular BI tools like Power BI, Tableau, or Qlik allows for generating required reports or extracting data into formats such as XLSX, XML, or JSON files. To eliminate the need for extensive ETL tools, an ELT (Extract, Load, Transform) process can be established to consume data directly extracted from source systems. Moreover, all the aforementioned database engines support External tables, which facilitate the importation of data into the data warehouse from the data lake. Creating the two-tier architecture.

I will focus a little bit more about it in the later part of the article, but briefly this approach empowers the creation of a data warehouse using platforms like BigQuery, Redshift, or Snowflake, leveraging familiar SQL and tools such as DBT (Data Build Tool) and GCP Dataform framework to establish ETL processes that cater to your organization's reporting requirements.

However, it's important to note that popular machine learning systems such as TensorFlow, PyTorch, XGBoost, and scikit-learn do not seamlessly integrate with the data warehouse. Additionally, there is often a need to analyze data that may not be implemented within the existing data model.

Two-tier Data Warehouse: Pros and Cons



Advantages:

- Data is structured, modeled, cleaned, and prepared.
- Easy access to data.
- Optimized for reporting purposes.

- Column and row-level security, data masking.
- ACID transaction support.

Disadvantages:

- Complexity and time-consuming for changes in implementation in the data model and the ETL process.
- Schema must be defined.
- Costs of the platform (depends on the database provider and type).
- Database vendor dependence (Complex in a migration from, for instance, Oracle to SQL Server).

Data Lake History

With the advent of new generations, the practice of collecting raw data has emerged. It is used, for example, in the data lake model.

These data lakes utilized low-cost storage with a file API, built on Apache Hadoop (2006) and its Hadoop Distributed File System (HDFS). Data was stored in various formats, including open file formats like Avro and Parquet, known for their efficient compression ratios and query performance. The data lake offered flexibility through its schema-on-read approach.

The introduction of new Hadoop and MapReduce technologies provided a cost-effective solution for analyzing large volumes of data. However, an alternative option still existed, allowing for the loading of a subset of data into a data warehouse for analysis using BI tools.

The use of open-source formats made the data lake accessible to a wide range of analytics tools, including machine learning and AI. Nonetheless, this gave rise to new challenges such as data quality, data governance, and the complexity of MapReduce jobs for data analysts. To address these challenges, Apache Hive (2010) was introduced, providing SQL access to data. The open-source nature of big data platforms and data lakes fostered the development of numerous tools. While this platform offered flexibility, it also introduced complexity due to the diverse range of tools utilized. As a result, a quiz titled 'Is it Pokemon or Bing data?' can be found here, allowing users to test their knowledge of the platform.

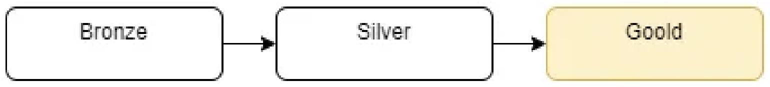
The next stage in data lake development came with the introduction of cloud environments and data stores such as S3, ADLS, and GCS, which gradually replaced HDFS. These new data stores were serverless and offered cost benefits, along with additional features like archiving. Cloud providers introduced new types of data warehouses, including Redshift (2012), Azure Data Warehouse, BigQuery (2011), and Snowflake (2014). This facilitated the implementation of a two-tier architecture, combining the data lake and the data warehouse. Another significant advancement was Apache Spark, which enabled large-scale data processing and supported popular programming languages such as Python, SQL, and Scala.

Data Lake Architectures

Medalion

I have come across several data lake designs, with the most popular one currently being the Medallion architecture. However, there are other patterns that can be encountered depending on the specific use case.

In the Medallion architecture, the bronze layer is responsible for ingesting raw data from sources and transforming it into the Silver layer, where it is stored in a common data format like Parquet. Finally, the data is aggregated and stored in the golden layer.



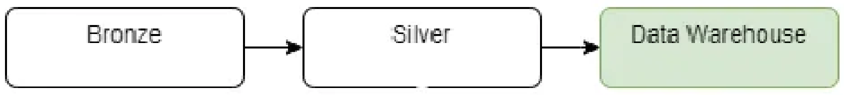
Bronze/Raw zone stores data in its original format(such as JSON, CSV, XML, etc). We keep data as we ingest from source systems. This data is immutable, files here are read-only. Files here should be organized in a folder per source system. We should prevent users from accessing this layer.

Silver/Cleansed zone stores data that may be enriched, cleaned, and converted to common formats like Parquet, Avro, and Delta(these formats provide a good compression ratio and read performance). We can validate data here, standardize, and harmonize. We have also defined a data type and a schema. This layer is accessible to data scientists and other users.

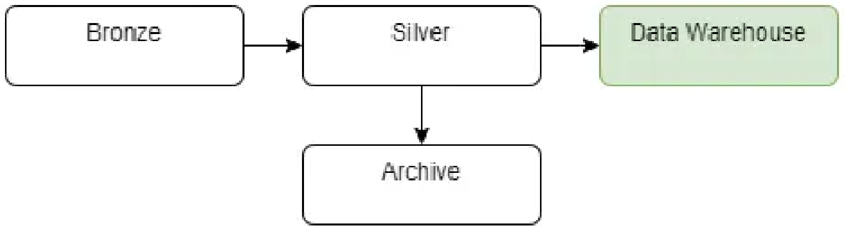
Gold/Curated zone This is the consumption layer, which is optimized for analytics rather than data ingestion or data processing. Data here can be aggregated or organized in a dimensional data model. You can use Spark to query this data or data virtualization. Performance could be an issue in this case. You can also import data to a BI tool to improve user experience with dashboards.

Evolution of Data Lake merges with Data Warehouse as a two-tier architecture

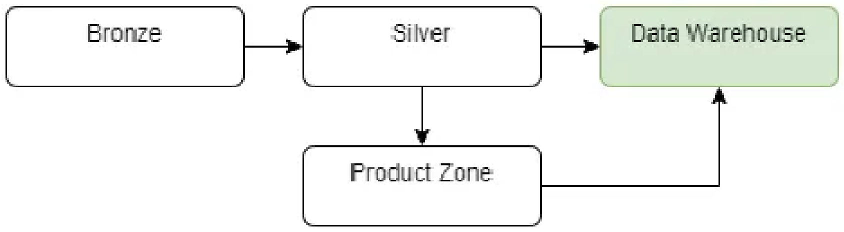
An alternative to the Medalion can be a two-tier data warehouse, which I mentioned above, and here I will explore it further. In this approach we have bronze and silver in the data lake and then we load data into the data warehouse for analytical purposes. To load data we can use Bulk loads commands or external tables. BigQuery, Redshift, Snowflake, and Synapse have this feature that will read data from data lake files and transform it by the ELT process. This way of working improves integration especially when we use parquet files that have a defined schema and it has a good performance to read.



We can also create an archive layer for instance in a separate bucket to reduce the cost of data storage(We can change the type of storage to reduce costs).



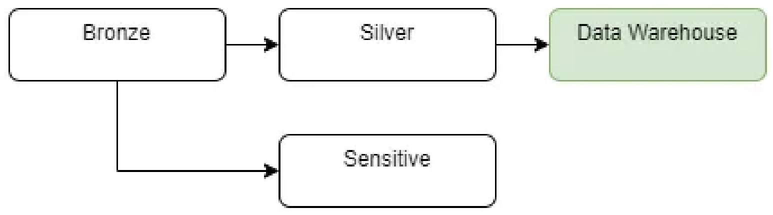
Additionally, we can create a product zone that will support data engineers and analysts to build their own structures and calculations for analytical and ML/AI purposes.



Archive Zone zone can be used to archive older data for future analyses to reduce the cost of storage. Cloud providers offer different types of storage which have different access times and costs. It's a good way to optimize the cost of data archivization.

Product/Labor zone This is the layer where exploration and experimentation occur. Here, data scientists, engineers, and analysts are free to prototype and innovate, mashing up their own data sets with production data sets. This zone is not a replacement for a test or dev environment.

Sensitive zone is another design for the data lake where we need to deal with sensitive data. Usually, only specific users have access to this layer and access is more restricted than to other parts of the data lake.



Folders organization in Data Lake

To prevent the occurrence of a data swamp, it is essential to establish a simple and self-descriptive folder structure within the data lake. A proper hierarchical structure should be implemented, ensuring that folders are human-readable, easily understandable, and self-explanatory. It is crucial to define naming conventions prior to initiating the data lake development process. These measures will facilitate the appropriate utilization of the data lake and aid in access management. Additionally, it is important to understand how to construct the folder structure to enhance the query engine's comprehension of the data. The recommended approach is to organize data in the manner presented within the bronze and silver layers.

```
-Source
  -Entity
    -year-month-date
      -files
```

Implementing this hierarchical structure will effectively manage partitioned data for the query engine. By organizing the data in a structured hierarchy, the query engine can efficiently navigate and access specific partitions, resulting in optimized performance and enhanced data retrieval capabilities.

```
-SAS
  -CTAS
    -2023-07-07
      ctas.parquet
```

```
-Source
  -Entity
    -files
```

When adopting a full load strategy for a specific table, it is possible to create a folder structure without partitioning. In such cases, where the entire table is loaded in its entirety, partitioning may not be necessary as the data is not segmented based on specific criteria. Instead, a straightforward folder structure can be established to organize and store the complete dataset.

```
-Salesforce
  -accounts
    accounts.parquet
```

When considering the gold layer, it is beneficial to establish a domain-oriented structure. This approach not only facilitates access management but also enhances the utilization of the data lake. By organizing the data within domain-specific categories, such as customer, product, or sales, users can easily navigate and leverage the relevant data for their specific business needs.

```
-Sales
  -accounts
    -accounts.parquet
-Finance
```

```
    -time_registraion
      -time.parquet
```

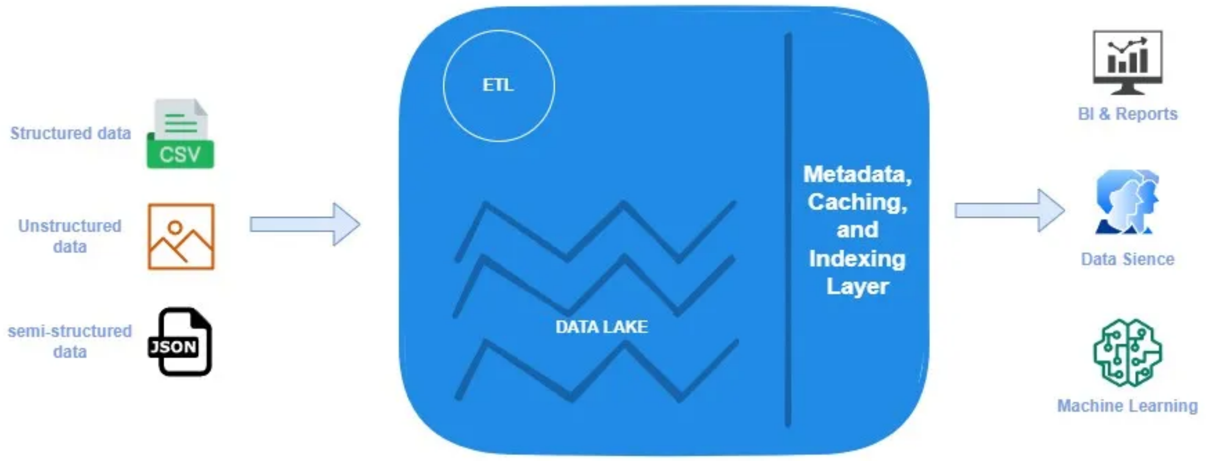
The efforts made in the direction of Lakehouses

When I initially encountered the term "Lakehouse", I found myself questioning the distinctions from the traditional data lake approach. At that time, AWS introduced features like Athena and Redshift external tables, and the Glue catalog facilitated seamless metadata sharing across various AWS services. This development enabled the creation of a data model within the gold layer, allowing querying capabilities akin to those found in a data warehouse. The availability of SQL engines such as Spark SQL, Presto, Hive, and External Tables implemented by major providers facilitated querying of the data lake. However, challenges persisted in areas such as data management, transaction ACID support, and achieving efficient performance through indexes.

Data Lakehouse

The **Data Lakehouse** architecture, implemented by Databricks, represents the **second generation** of data lakes and offers a unique approach to designing data platforms by combining the strengths of both Data Lake and Data Warehouse. It serves as a **single unified platform** for housing both a data warehouse and a data lake. The Data Lakehouse architecture not only supports ACID transactions like a traditional data warehouse but also provides the cost-effectiveness and scalability of storage found in a data lake. This architecture enables direct access to all layers of the data lake and includes **schema support** for effective data governance. Additionally, the Data Lakehouse introduces features such as **indexes, data caching, and time travel** to enhance performance and functionality. It continues to support both structured and unstructured data types, with data stored in file formats.

To implement the Lakehouse architecture, Databricks has developed a new type of file format called Delta (further details on file formats will be described later in this post). However, as alternatives to Delta, other file formats such as **Iceberg** and **Apache Hudi** can be utilized, offering similar features. These new file formats empower Spark to leverage new data manipulation commands that improve data loading strategies. In the Data Lakehouse, updates to rows in a table can be performed without the need to reload the entire dataset into the tables.



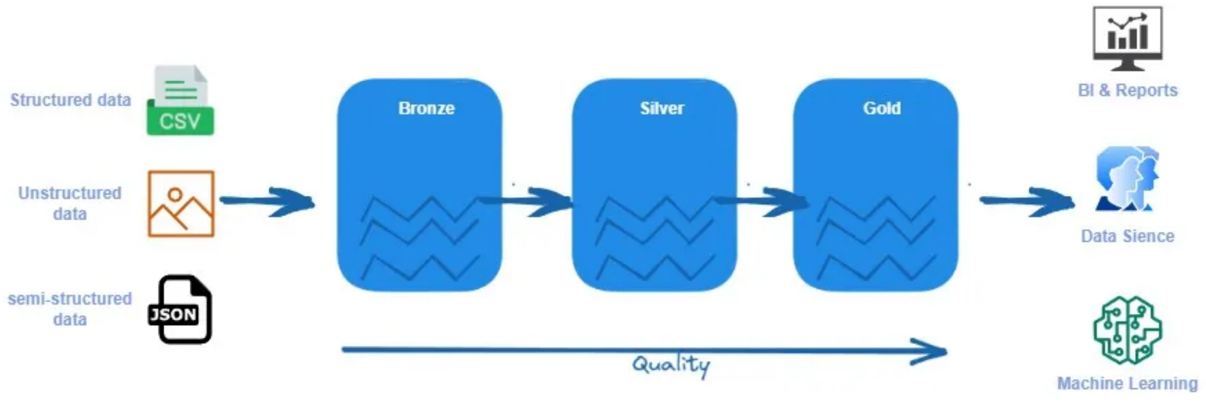
Data Lakehouse Architecture

Data Lakehouse organizes data mostly using the medallion architecture. As I mentioned we have three layers of data lakehouse Bronze, Silver, and Gold. The main difference in the case of data Lakehouse will be the file format. We use the Delta format to store data. Alternatively, we can use Iceberg or Hundi as the main data format if we work with other tools than Databricks. We can organize data in the lakehouse in a few ways that I'll present below.

The Bronze we use to ingest data from sources "as-is". Data is source-system aligned and organized in the storage. If we import data from sources we will save it in Delta format. Data can be ingested by ingestion tools and saved in its native format if it doesn't support Delta or when source systems dump data directly into storage.

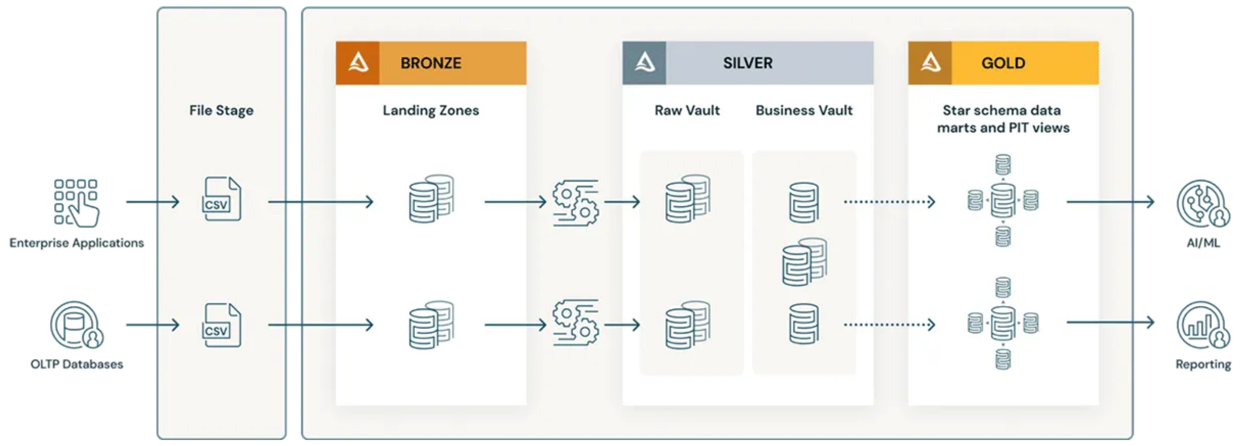
The Silver keeps data that were cleaned, unified, validated, and enriched with reference data or master data. The preferable format is Delta. Tables still correspond to source systems. This layer can be consumed by Data Sciences, ad-hoc reporting, advanced analytics, and ML.

The Gold layer hosts a data model that will be utilized by business users, BI tools, and reports. We use the Delta format. This layer keeps business logic that includes calculations and enrichment. We can have only a specific set of data from Silver or only aggregate data. The structure of tables is subject-oriented in this layer.



Data Vault with the Lakehouse

Another approach to organizing the Lakehouse is through the utilization of Data Vault methodology. This method is particularly suited for organizations dealing with vast amounts of data, multiple data sources, and frequent integration changes. The Data Vault pattern is similar, but it involves important changes in data modeling and organization.



The Bronze layer is responsible for ingesting data from source systems while maintaining the original structure 'as-is.' Additional metadata columns, such as load time and batch ID, can be added. This layer stores historical data, delta loads, and change data captures. When importing data from source systems, it is advisable to save it in the Delta format to enhance performance. In cases where data arrives in different formats, implementing a landing and stage zone or an additional Landing layer can facilitate data reception from source systems. In such scenarios, data is stored in its native format and then converted to Delta format in the Bronze/stage zone. The Databricks AutoLoader tool can be utilized to streamline the transformation process in the Bronze/stage zone.

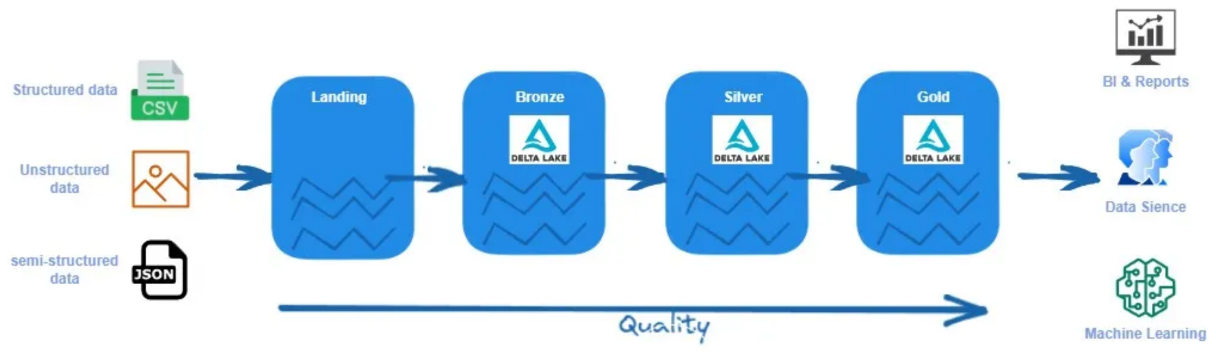
The Silver Layer often referred to as the Enterprise repository, houses cleaned, merged, and transformed data from the Bronze layer. This data can be readily consumed by users, supporting their reporting, advanced analytics, and ML processes. From a data modeling perspective, this layer adopts the 3rd-Normal Form and leverages the principles of the Data Vault methodology to accommodate agile development and adapt to fast-paced changes. The Data Vault model typically encompasses three types of entities:

- **Hubs** represent a core business entity, like customers, products, orders, etc. Analysts will use the natural/business keys to get information about a Hub. The primary key of Hub tables is usually derived by a combination of business concept ID, load date, and other metadata information that we can achieve using Hash, MD5, and SHA functions.

- **Links** represent the relationship between Hub entities. It has only the join keys. It is like a Factless Fact table in the dimensional model. Lack of attributes — only joins keys.
- **Satellite** tables have the attributes of the entities in the Hub or Links. They have descriptive information on core business entities. They are like a normalized version of a Dimension table. For example, a product hub can have many satellite tables like type attributes, costs, etc.

The tables are organized based on domains like Customer, Product, or Sales.

The Gold Layer serves as the foundation for data warehousing models or data marts. Following a dimensional modeling approach, such as the Kimball methodology, this layer focuses on reporting purposes. Users can leverage BI tools to consume data from this layer or perform ad-hoc reporting. Data marts can also be established within the Gold Layer to address specific data modeling needs for different departments. Organizing data by projects or use cases, such as C360, marketing, or sales, facilitates ease of data consumption using star schema structures.



Lakehouse with optional Landing layer.

Lakehouse File Formats

Delta is an open-source data storage format introduced by Databricks, aimed at providing reliable and high-performance data processing. It offers ACID transactional capabilities, versioning, schema evolution, and efficient data ingestion and query operations. Delta is commonly used in Apache Spark environments and supports both batch and streaming data workloads. It provides features such as data compaction, data skipping, and time travel queries, enabling efficient data management and analysis.

Iceberg is an open-source table format developed by Netflix, designed specifically for large-scale data lakes. Its primary focus is on providing ACID compliance, time travel capabilities, and schema evolution while ensuring excellent performance and scalability. Iceberg works well with popular query engines such as Apache Spark and Presto. It offers features like efficient data manipulation, column-level statistics, and metadata management. Iceberg allows users to perform fast, consistent, and reliable analytics on large datasets.

Apache Hudi is an open-sourced by Uber, Hudi was designed to support incremental updates over columnar data formats. It supports ingesting data from multiple sources, primarily Apache Spark and Apache Flink. It also provides a Spark-based utility to read from external sources such as Apache Kafka. Reading data from Apache Hive, Apache Impala, and PrestoDB is supported. There's also a dedicated tool to sync the Hudi table schema into Hive Metastore.

UniForm

The upcoming Delta Lake 3.0 aims to provide a ***Universal Format (UniForm) for all three** OTF*. The objective would be to have Delta Lake tables accessible as Hudi or Iceberg tables. UniForm is currently in Preview and there are limitations.

Data lake & Data Warehouse or Lakehouse

Is the Lakehouse the best approach to adopt? As is often the case, the answer is 'it depends.

There are various factors to consider when deciding on your architecture. For example, if your intention is to work with BigQuery, Snowflake, Synapse, or Redshift, employing a two-tier architecture remains a sound choice. This allows you to harness the advantages of a data lake while hosting your data model within a DBMS database. It is important to note that the Lakehouse concept is still relatively new and not as mature as other architectural paradigms. By utilizing both the data lake and the data warehouse, you can leverage the strengths of each platform and tap into different skill sets within your teams. For instance, your data engineering team may employ Python and Spark, while teams solely focused on the data warehouse can utilize SQL.

It is worth conducting a thorough analysis when selecting a Data Warehouse engine, as each option presents its own set of considerations. These may include factors such as concurrency, auto-scaling, separation of compute from storage, integration with file formats like Delta, Iceberg, and Hudi, limited compatibility with non-native cloud providers, zero-copy cloning, performance tuning, costs, GIS support, and maintenance efforts. Data warehouses offer robust features in terms of data consistency and governance. They are equipped with built-in mechanisms for data validation, data quality checks, and the enforcement of data governance policies. Such capabilities prove crucial for organizations with stringent data governance and compliance requirements.

A two-tier architecture is a suitable choice when dealing with complex ETL processes and when massive compute power for data presentation is not required. In such cases, leveraging the data lake alongside Spark for data processing, combined with an open-source database like PostgreSQL to serve the data, can be an effective approach.

If you have any questions or would like to further discuss data platform architecture, feel free to reach out to me on LinkedIn.

Search

Write

Written by Mariusz Kujawski

Follow

529 Followers

Cloud Data Architect | Data Engineer https://www.linkedin.com/in/mariusz-kujawski-812bb1103/

More from Mariusz Kujawski