# Lakehouse architecture:

Lakehouse architecture is a modern approach to data engineering that combines the advantages of data lakes and data warehouses. It is designed to provide a centralized, secure, and scalable platform for storing, managing, and analyzing large amounts of structured and unstructured data.

The Lakehouse architecture is composed of three layers: bronze, silver, and gold. Each layer serves a specific purpose and provides different levels of data quality, security, and performance.

### Bronze Layer

The bronze layer is the foundation of the Lakehouse architecture. It is responsible for storing raw data in its original format, without any modification or transformation. The data is stored in a data lake, which is a large, centralized repository that allows for the storage of structured, semi-structured, and unstructured data. This allows for the storage of large amounts of data in various formats, such as JSON, CSV, and Parquet, making it easy to handle different types of data.

#### Data loading:

**Batch Loading:** This method involves loading data in large batches, typically at specific intervals or on a schedule. Batch loading is suitable for loading large amounts of data, such as historical data.

**Real-time Streaming:** This method involves loading data in real-time as it is generated. Real-time streaming is suitable for loading data that is constantly changing, such as sensor data or social media data.

**Direct Loading:** This method involves loading data directly from the source, such as a database, into the data lake.

Once the data is loaded into Bronze Layer, it is stored in its original format, without any modification or transformation. This allows for the preservation of the data's integrity and the ability to track changes over time.

### Silver Layer

The silver layer, also known as the transformation layer, is the second layer of the Lakehouse architecture. It is responsible for cleaning and transforming the data from the bronze layer into a format that is suitable for analysis and reporting.

The silver layer includes several tasks, such as:

**Data Cleansing**: This process involves removing any inconsistencies or errors in the data, such as missing values, duplicate records, or incorrect data types. This ensures that the data is accurate and reliable.

**Data Transformation**: This process involves changing the data's format or structure to make it more suitable for analysis and reporting. This can include tasks such as normalizing data, converting data types, or aggregating data.

**Data Quality Checks**: This process involves verifying the data's accuracy and completeness. This can include tasks such as data validation, data reconciliation, and data profiling.

**Data Lineage**: This process involves tracking the origin and flow of the data from the raw data layer to the transformed data layer. This ensures that the data is traceable and can be audited.

**Data Catalog**: The silver layer also includes a data catalog, which is a centralized repository of metadata that describes the data and its lineage. The data catalog is used by data engineers and data scientists to discover and understand the data.

Overall, the silver layer plays a crucial role in the Lakehouse architecture by ensuring that the data is accurate, reliable, and suitable for analysis and reporting. The data cleansing, transformation, and quality-checking tasks, as well as the data lineage and data catalog, ensure that the data is of high quality and can be easily understood and used by business users and data analysts.

### Gold Layer

The gold layer, also known as the reporting and analysis layer, is the top layer of the Lakehouse architecture. It is responsible for storing the cleaned and transformed data from the silver layer in a data warehouse.

The gold layer is also used to create reports and dashboards, which can be used by business users to make data-driven decisions. These reports and dashboards can be created using business intelligence tools, such as Tableau, Power BI, and Looker.

The gold layer also includes data security and access controls, which ensures that the data is only accessible to authorized users. This is done through user authentication and authorization, and data encryption.

# Why are we using Spark?

Apache Spark is a powerful tool for data engineering because it offers several key features that make it well-suited for large-scale data processing and analysis. These include:

**Scalability**: Spark can handle very large datasets and process them quickly and efficiently, making it ideal for big data applications.

**In-memory processing**: Spark uses in-memory processing to speed up data processing and analysis, which is particularly useful for iterative algorithms and machine learning tasks.

**Flexibility**: Spark supports a wide range of data sources and formats, including structured and semi-structured data, making it easy to work with different types of data.

**Easy** **to** **use**: Spark has a simple and intuitive API that makes it easy for developers to write and execute data processing tasks, even for complex algorithms.

**Open-source**: Spark is an open-source project, which means that it is free to use and can be easily integrated with other tools and technologies.

**In-memory processing** refers to the use of computer memory (RAM) to process data, rather than traditional disk storage. This allows for faster data processing and retrieval, as data can be accessed more quickly from memory than from disk.

**Parallel processing** refers to the use of multiple processors or cores to perform tasks simultaneously. This allows for faster processing of large amounts of data, as multiple tasks can be completed at the same time.

**Advantages of in-memory processing and parallel processing include:**

**Faster processing times**: as data is accessed directly from memory, and tasks are completed simultaneously, data processing can be completed much faster than with traditional disk-based processing.

**Increased scalability**: as more processors or cores are added, the processing power can be increased, allowing for the processing of larger amounts of data.

**Improved** **performance**: by processing data directly in memory, the performance of applications can be improved, as the time to access data is greatly reduced.

**Reduced** **costs**: in-memory processing can reduce the need for expensive disk storage and I/O operations, which can be costly in terms of both hardware and maintenance.

**Better** **decision** **making**: by providing faster access to data, in-memory processing can enable faster and more accurate decision making in real-time applications.

# Difference between parquet and delta format

Parquet and Delta are both file formats used for storing large amounts of data in a columnar format. However, there are some key differences between the two:

**Compression**: Parquet uses a variety of compression algorithms to compress data, while Delta uses a more efficient encoding method that reduces the size of data even further.

**Schema** **Evolution**: Parquet requires a fixed schema for reading and writing data, while Delta supports schema evolution, allowing for the addition or removal of columns without the need for rewriting the entire dataset.

**Performance**: Delta is optimized for fast read and write operations, and is designed for use with big data processing tools such as Apache Spark and Apache Hive. Parquet is also fast, but not as optimized for big data processing.

**Data** **Management**: Delta provides built-in support for data management tasks such as data versioning and rollback, while Parquet does not.

Overall, Delta is a more advanced and efficient file format, while Parquet is a more widely-used and established format. Both formats have their own use cases and are suitable for different types of data processing tasks.

#### Delta tables are used for several reasons:

Data Management: Delta tables provide built-in support for data management tasks such as data versioning and rollback, making it easy to track changes to your data over time and revert to previous versions if necessary.

Performance: Delta tables are optimized for fast read and write operations, making them ideal for use with big data processing tools such as Apache Spark and Apache Hive. This allows for faster data processing and querying.

Schema Evolution: Delta tables support schema evolution, allowing for the addition or removal of columns without the need for rewriting the entire dataset. This makes it easier to adapt to changes in your data and allows for more flexible data processing.

Data Integrity: Delta tables provide a unique transaction ID for every change, making it easy to detect and correct errors in your data.

Data Governance: Delta tables offer a set of features for data governance such as time travel, auditing, and access control, making it easier to manage data access and compliance.

Overall, Delta tables are designed for big data processing and are suitable for use cases where data needs to be managed and processed at scale, and where performance, data integrity and governance are critical.

Delta log is a feature of Delta tables that allows for the tracking and management of changes to the data over time. The delta log is a separate file that is stored alongside the data files in the Delta table.

The delta log contains a record of all the changes made to the data, including insertions, updates, and deletions. Each change is recorded with a unique transaction ID and a timestamp, making it easy to track and revert to previous versions of the data.

#### The delta log is used for several purposes, such as:

Data versioning: The delta log allows for easy rollback to previous versions of the data. This can be useful for testing, auditing, or for recovering from errors.

Data recovery: The delta log can be used to recover data that was accidentally deleted or overwritten.

Data lineage: The delta log can be used to track the origin and history of the data, making it easier to understand how it was created and how it has changed over time.

Data Governance: The delta log can be used to track who made what change and when, making it easier to manage data access and compliance.

Overall, the delta log is a powerful tool for managing and tracking changes to data in a Delta table, and is critical for maintaining data integrity and governance.