CYBERTREND LEARNING CENTER
GRAMEDIA ACADEMY
MAKARA UI ACADEMY
Upgrade Yourself, Make Your Future

Pandas + Prefect

Course Overview

1 **Encounter 1**

- Python Pandas & MySQL
- Data to HDFS
- Reading with HIVE

2 **Encounter 2**

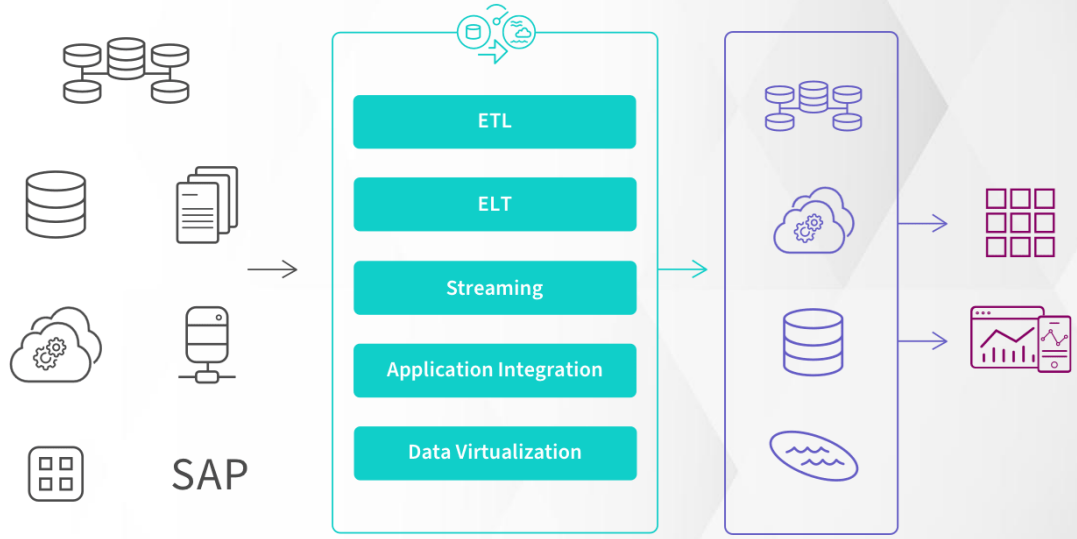
- Perfect workflows Basics
- Scheduling & Orchestrator
- Cloud Development

Overview of Data Integration

What is Data Integration?

Data integration refers to the process of bringing together data from multiple sources across an organization to provide a complete, accurate, and up-to-date dataset for BI, data analysis and other applications and business processes.

It includes data replication, ingestion and transformation to combine different types of data into standardized formats to be stored in a target repository such as a data warehouse, data lake or data lake house.



Benefits of data integration

Data Integration in Modern Business

- **Improved Data Quality**

Through data transformation and cleansing processes, data integration helps improve data quality by identifying and correcting errors, inconsistencies and redundancies. Accurate, reliable data instills confidence in decision makers.

- **Faster Decision-Making And Collaboration**

Integrating multiple data streams provides a comprehensive overview of your operations. This holistic perspective allows you to identify market shifts and capitalize on growth opportunities through data-driven decisions

- **Stronger Data Security**

A centralized data platform offers a higher security degree. With all data housed under one roof, it's easier to implement and manage strict security protocols. Detailed access controls and monitoring mechanisms can effectively prevent unauthorized access and data breaches. This integrated approach to security also makes regulatory compliance straightforward.

Benefits of data integration

Data Integration in Modern Business

- **Cost Savings**

Automating manual tasks with the help of data integration cuts down on employee expenses. The time that was once consumed by these everyday tasks can now be redistributed to make the most of your staff's efficiency. When you consolidate data into a unified system, it trims down the expenses linked to operating and maintaining multiple databases. This reduces costs across system licenses, infrastructure, and even staff training.

- **Increased Revenue Streams**

With a fully integrated data system, you are better positioned to uncover hidden opportunities. This could be in the form of new products, services, or even previously untapped markets. The kind of insights you gather through regular data analysis has the potential to open up new and inventive sources of revenue.

- **Data Driven Innovation**

Integrated data can uncover patterns, trends and opportunities that might not be apparent when enterprise data is scattered across disparate systems. This enables organizations to innovate and create new products or services.

Challenges of Data Integration

Data From
Legacy Systems

Data From Newer
Business Demands

External Data

Keeping Up

Integration Strategies for Business

1. Manual Data Integration

   It involves **manually collecting the data**, linking its many sources, and cleaning it. This approach is a manual one and highly laborious but delivers results. It's best suited to simple integrations as it can grow complex and time-consuming for more involved operations.

2. Application-based Integration

   The integration based on applications **method requires the use of software programs**. In this method, the programs handle every task in the data integration process. They identify, retrieve, clean, and combine data from many sources. Automation makes it simple to transfer data between sources.

   Because it is popular among businesses operating in hybrid cloud environments, this strategy is commonly referred to as enterprise application integration. These companies must work with numerous data sources, including those that are on premises and in the cloud.

Integration Strategies for Business

3. Uniform access integration

   Using this method, data is **accessed from different collections and presented uniformly in a single system**, yet the data is stored in its original location. This is **the best course of action for businesses** that must access several different systems. This method can produce insights without incurring the expense of making a backup or duplicating the data.

4. Common storage integration

   Common data storage, often referred to as **data warehousing**, is a data integration technique, where information from **various sources is collected, transformed, and stored in a unified, central location**.

   This data integration strategy ensures better data integrity, as all necessary information can be accessed from a single source. Consequently, you get a holistic view of business data and provide stakeholders with easy access to information for decision-making.

Integration Strategies for Business

Strategies	Weaknesses	Strengths	Best Suits
Manual Integration	• Resource-intensive approach, often requiring substantial time investment and advanced programming skills • Not suitable for large volumes of data or rapidly changing data sources due to its less scalable nature	• Cost efficient and allows for a quick start • Can be easily customized to meet your specific data integration requirements	• Smaller scale projects or scenarios where custom handling of data is paramount
Application-based	• Requires careful management to ensure data security during the exchange • Demands substantial technical expertise for the setup and management of the integrations	• Offers real-time data integration, ensuring all applications have access to the most recent data • Provides a comprehensive view of integrated data and easy information exchange	• Businesses with multiple software applications that need data sharing and synchronization • Scenarios requiring real-time or near-real-time data integration

Integration Strategies for Business

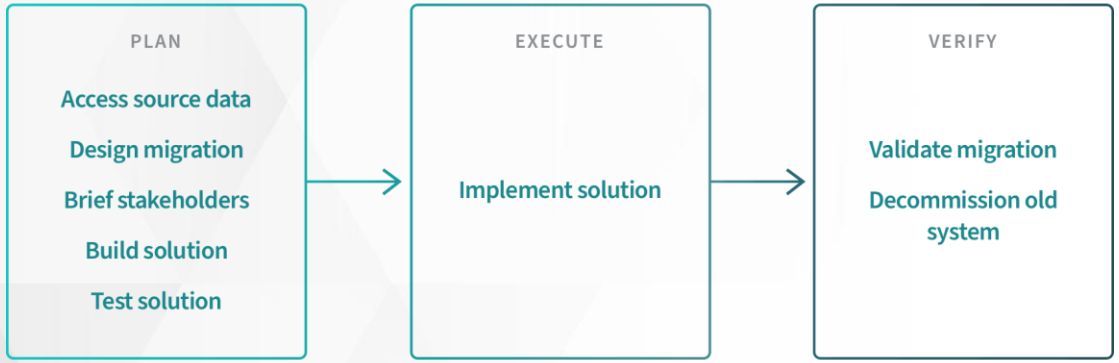
Strategies	Weaknesses	Strengths	Best Suits
Uniform access	• The performance of the integrated view depends heavily on the stability and speed of the underlying source systems • Accessing data from many disparate sources increases the potential for data integrity to be compromised if not managed carefully. • Setting up and managing the virtual integration layer can be challenging and may require technically-sound data analysts or managers. • This method is highly dependent on the stability and availability of all connected source systems.	• Data updates in source systems are propagated to the consolidated view almost instantly, as the data is accessed live from the source. • It avoids the need for a separate, expensive data warehouse or storage system, reducing costs and infrastructure requirements. • Applications interact with a consistent data layer, eliminating the need for developers to write code specific to each source system.	• Real-time or near real-time data access is crucial. • Businesses need to access data from multiple, highly disparate systems. • Minimizing data duplication and associated storage costs is a priority. • The volume of data requests is not so high that it would overwhelm the source systems' capacity.
Common storage	• Time and resource intensive approach, particularly when dealing with large volumes of data • High storage and maintenance costs • Demands strict data quality control for maintaining data reliability	• Provides a consistent view of data, simplifying data management and analysis • Ensures easy access to data for various decision-makers	• Businesses requiring a consolidated view of their data for easy access and analysis • Scenarios where data from multiple sources needs to be stored and accessed in a single location

Typically Use Case

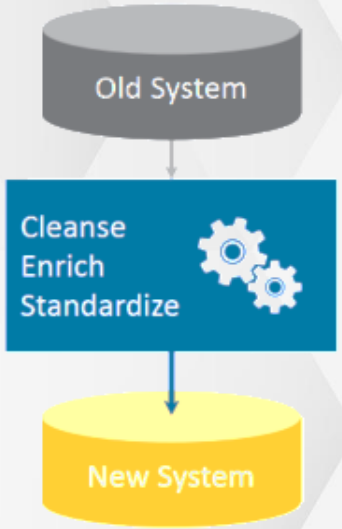
DATA MIGRATION

Data migration is the **process of moving data from one system to another**—for example, from an old database to a new one, from on-premise storage to the cloud, or from one application to another—**while keeping the data accurate, complete, and usable**.

The three key phases of the data migration process for most projects



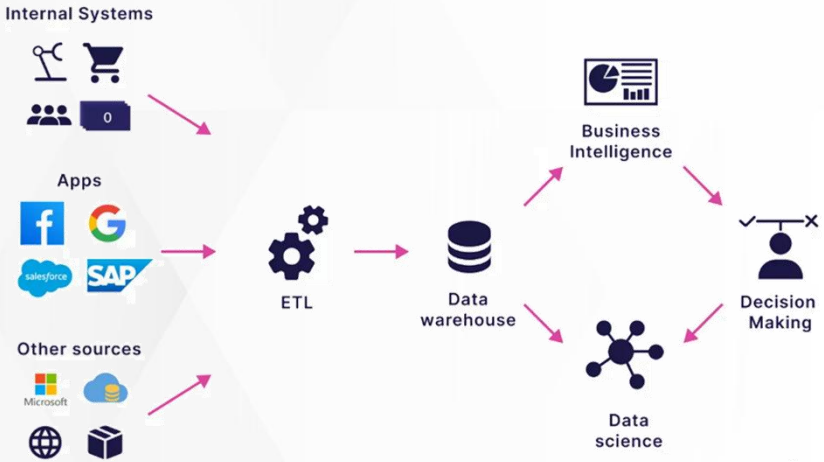
- Data Migration
- Data Warehousing
- Data Consolidation
- Data Synchronization



Typically Use Case

DATA WAREHOUSING

A **data warehouse** is a **centralized system used to store, integrate, and analyze data from multiple sources** so organizations can make better business decisions.



Typically Use Case

DATA CONSOLIDATION

As the term implies, **data consolidation** means bringing together data from various sources and assembling it within a single location. Data consolidation allows users to engage data from a single point of access and fosters the generation of data insights.

Data Consolidation Techniques

ETL

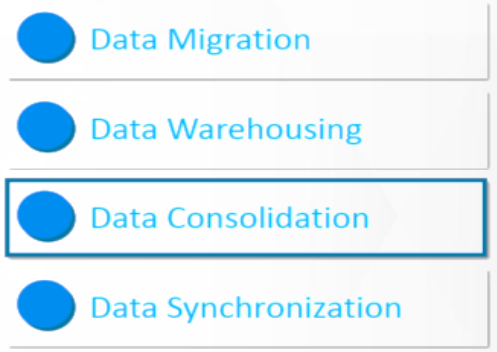
The most important data consolidation technique is known as Extract, Transform, Load. ETL processes begin with ETL tools extracting information from data sources. Then that data is transformed into a standard informational format. Lastly, the data is loaded into a selected destination.

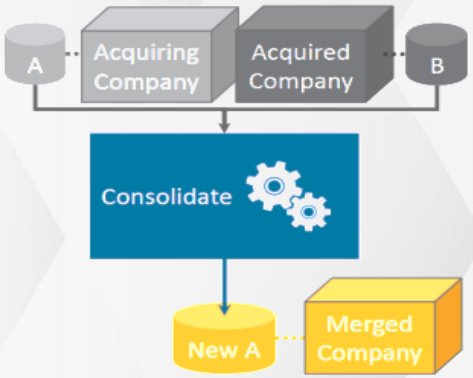
ELT

An emerging counterpart to ETL strategy is called Extract Transform, Load. The re-arrangement of ELT steps is crucial. In ELT, data is extracted, then loaded to a type of staging area. Data remains here as various entities within the organization study it from different angles, ultimately transforming the data.

Data Lake

Data warehousing is used in part to clean or process data. A data lake, on the other hand, is simply **a data repository that offers none of the data-processing capabilities**. A data lake is essentially a place to park data while it's still in its rawest form. Typically, this is where a company might deposit obscure data.



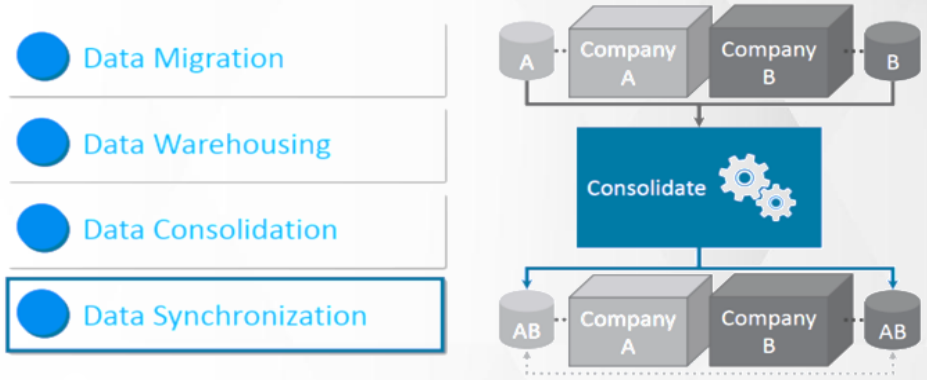


Typically Use Case

DATA SYNCHRONIZATION

Data synchronization, or data sync, is the continuous process of **keeping data records accurate** and **uniform across network systems and devices**.

Data synchronization prevents data conflicts, which can result in errors and low-quality, low-trust data. Synchronized, trustworthy data is essential for security, compliance, and a wide variety of operational functions.



Direction	Concept	Example
One-way Synchronization	Data moves **from A → B only**.	Transaction DB → Data Warehouse
Two-way Synchronization	Data flows **A ↔ B**.	CRM ↔ ERP Mobile app ↔ Web app
Multi-way Synchronization	Data is synchronized **across multiple systems**.	Inventory system ↔ Website ↔ Mobile app ↔ POS
Hybrid Synchronization	Combination of **one-way + two-way** sync.	OLTP DB ↔ CRM (two-way) CRM → Data Warehouse (one-way)

Python Pandas & MySQL

Do you know PYTHON?

Python is a **programming language** that's designed to be **easy to read, easy to write, and very powerful**.

What is python used for?

- Data Analysis & Data Science
- Machine Learning & Artificial Intelligence
- Web Development
- Automation & ETL
- Database & Backend Processing

```
print('Hello, world!')
```

What is Python used for?

Data Analysis & Data Science

What people do with Python:

- Read data from files (CSV, Excel, Parquet, JSON)
- Clean messy data (missing values, duplicates, wrong formats)
- Transform data (filter, group, aggregate)
- Analyze trends and patterns
- Visualize results in charts and graphs

Why Python is good for this:

- Short and readable code
- Powerful libraries like **pandas**, NumPy, matplotlib

Top Library Python

Pandas is an open-source software library built on Python for data analysis and data manipulation.

The pandas library provides data structures designed specifically to handle tabular datasets

What is Python used for?

Data Analysis & Data Science (Machine Learning & AI)

Read Data (.CSV)

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.to_string())
```

Aggregate Function

```
mean()   -> Compute the average of group values.
sum()    -> Compute the sum of group values
min()    -> Compute the minimum of group values
max()    -> Compute the maximum of group values
count()  -> Count the number of non-missing values
describe() -> Generate descriptive statistics values
```

Actual Duplicate Rows

```
duplicate_rows = df[df.duplicated(keep=False)]
print("Duplicate Rows:")
print(duplicate_rows)
```

Drop Duplicate Rows

```
df_unique = df.drop_duplicates()
```

To Import Library Python

```
import math
import pandas as pd
from sqlalchemy import create_engine
import sqlalchemy as db
import pyspark from pyspark.sql
import SparkSession
```

What is Python used for?

Web Development

Python is widely used for web development due to its fast processing, multipurpose frameworks, testing, and handling of the complete development process, and along with its large community support, Python provides a complete package solution for web development which is easy to maintain as well. Development of Python web applications focuses on scalability and great user interface.

Install Flask:

```
pip install Flask
```

Create a Flask app file called app.py

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def home():
    return "Hello World!"

if name == '__main__':
    app.run()
```

Run the app:

```
python app.py
```

Library Python for web Dev

What is Python used for?

ETL

extract.py

```python
import pandas as pd

def extract_data(file_path):
    try:
        data = pd.read_csv(file_path)
        print("Data extraction complete.")
        return data
    except FileNotFoundError:
        print(f"Error: {file_path} not found.") return None
```

"""Extracts data from a CSV file into a pandas DataFrame."""

transform.py

```python
import pandas as pd

def transform_data(df):
    if df is not None:
        df.rename(columns={'old_name': 'new_name'}, inplace=True)
        df['column_b'].fillna(value=0, inplace=True)
        print("Data transformation complete.")
        return df
    return None
```

"""Performs data cleaning and transformation."""

load.py

```python
from sqlalchemy
import create_engine

def load_data (df, db_connection_string, table_name):
    if df is not None:
        try:
            engine = create_engine(db_connection_string)
            df.to_sql(table_name, con=engine, if_exists='replace', index=False)
            print(f"Data loaded into table '{table_name}'.")
        except Exception as e:
            print(f"Error during data loading: {e}")
```

"""Loads a DataFrame into a target database table."""

Do you know MySQL?

MySQL is an **open-source**, **relational database management system (RDBMS)** that uses **Structured Query Language (SQL)** to **manage** and **manipulate** data. It is one of the most popular database systems used in web applications, known for its speed, reliability, and ease of use. MySQL is commonly used in conjunction with programming languages such as PHP, Java, and Python to build dynamic websites and applications.

```
from sqlalchemy import create_engine
engine = create_engine(
    "mysql+pymysql://username:password@host:3306/nama_database"
)
with engine.connect() as conn:
    result = conn.execute("SELECT * FROM table_name")
    print(result.fetchone())
```

Import Library

Connection DB

Execute Table

Data to HDFS

Hadoop Distributed File System (HDFS)

Hadoop Distributed File System (HDFS) is a file system that manages large data sets that can run on commodity hardware. This open source framework works by rapidly transferring data between nodes. It's often used by companies who need to handle and store big data. HDFS is a key component of many Hadoop systems, as it provides a means for managing big data, as well as supporting big data analytics.

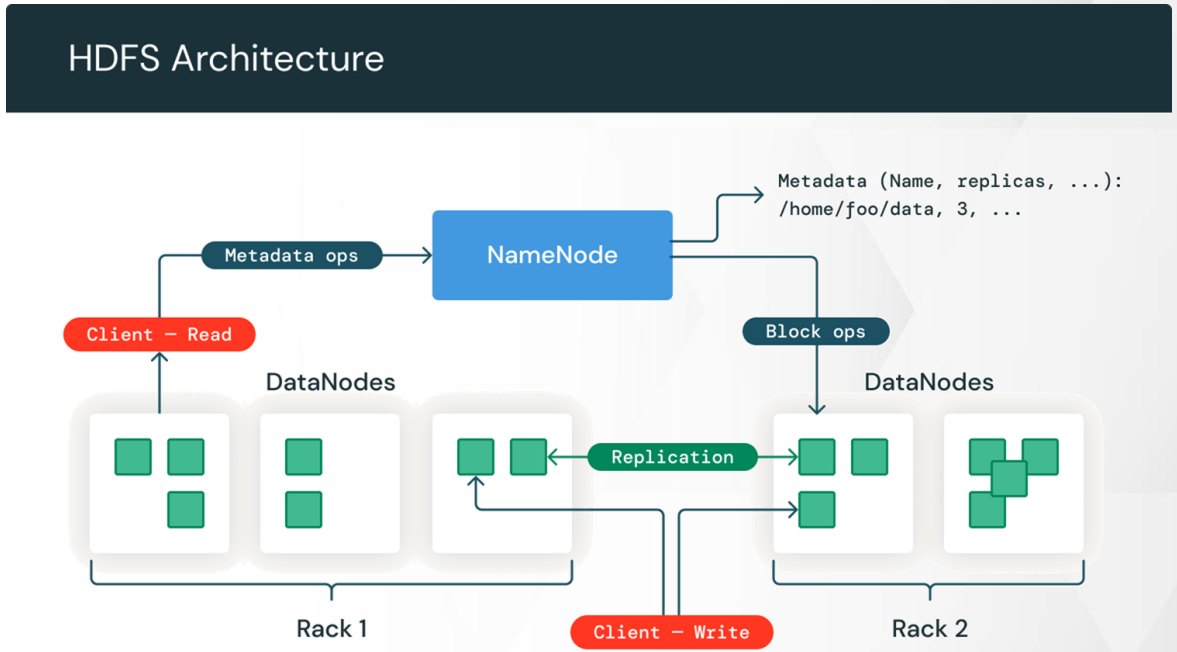
What Is Hadoop?

Apache Hadoop is an open source, Java-based software platform that manages data processing and storage for big data applications. The platform works by distributing Hadoop big data and analytics jobs across nodes in a computing cluster, breaking them down into smaller workloads that can be run in parallel. Some key benefits of Hadoop are scalability, resilience and flexibility.

HDFS Architecture

HDFS is designed to be highly scalable, reliable, and efficient, enabling the storage and processing of massive datasets. Its architecture consists of several key components:

1) NameNode
2) DataNode
3) Secondary NameNode
4) HDFS Client
5) Block Structure

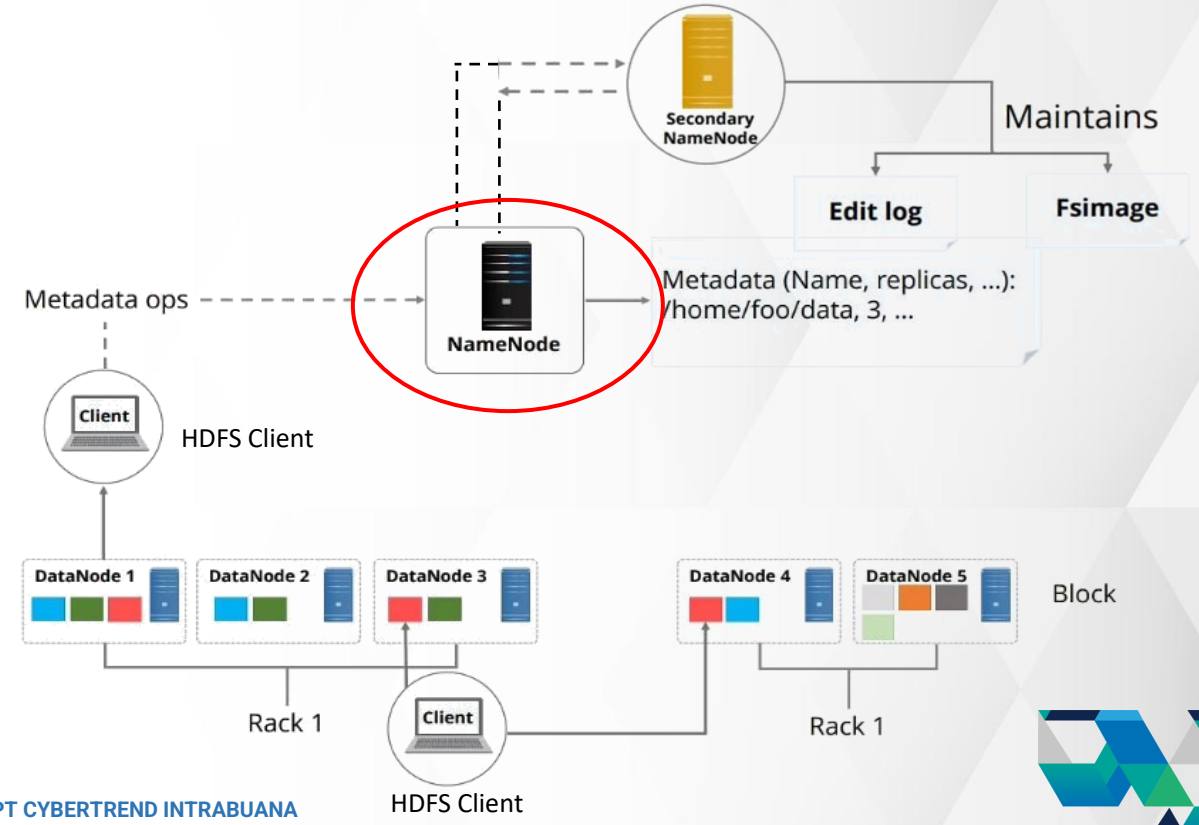


NameNode

The NameNode is the master server that manages the filesystem namespace and controls access to files by clients. It performs operations such as opening, closing, and renaming files and directories. Additionally, the NameNode maps file blocks to DataNodes, maintaining the metadata and the overall structure of the file system. This metadata is stored in memory for fast access and persisted on disk for reliability.

Key Responsibilities:

- Maintaining the filesystem tree and metadata.
- Managing the mapping of file blocks to DataNodes.
- Ensuring data integrity and coordinating replication of data blocks.

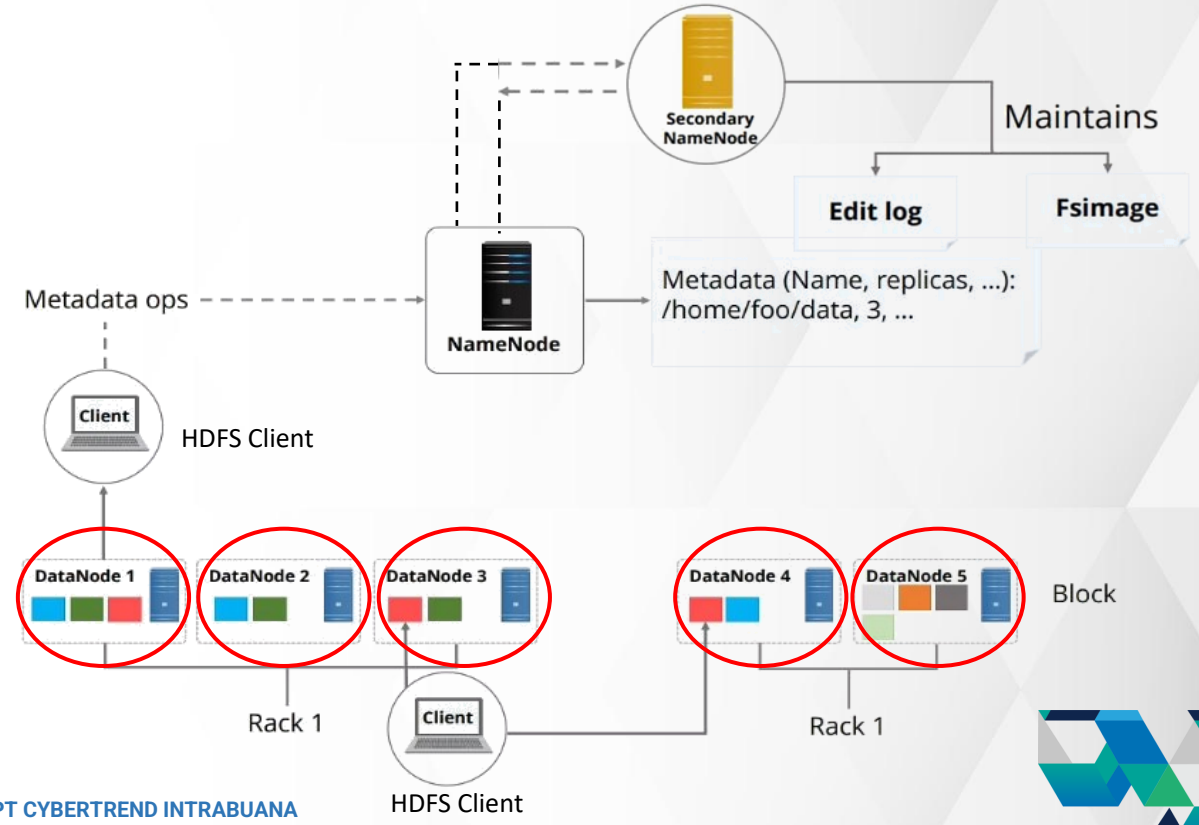


DataNode

DataNodes are the worker nodes in HDFS, responsible for storing and retrieving actual data blocks as instructed by the NameNode. Each DataNode manages the storage attached to it and periodically reports the list of blocks it stores to the NameNode.

Key Responsibilities:

- Storing data blocks and serving read/write requests from clients.
- Performing block creation, deletion, and replication upon instruction from the NameNode.
- Periodically sending block reports and heartbeats to the NameNode to confirm its status.

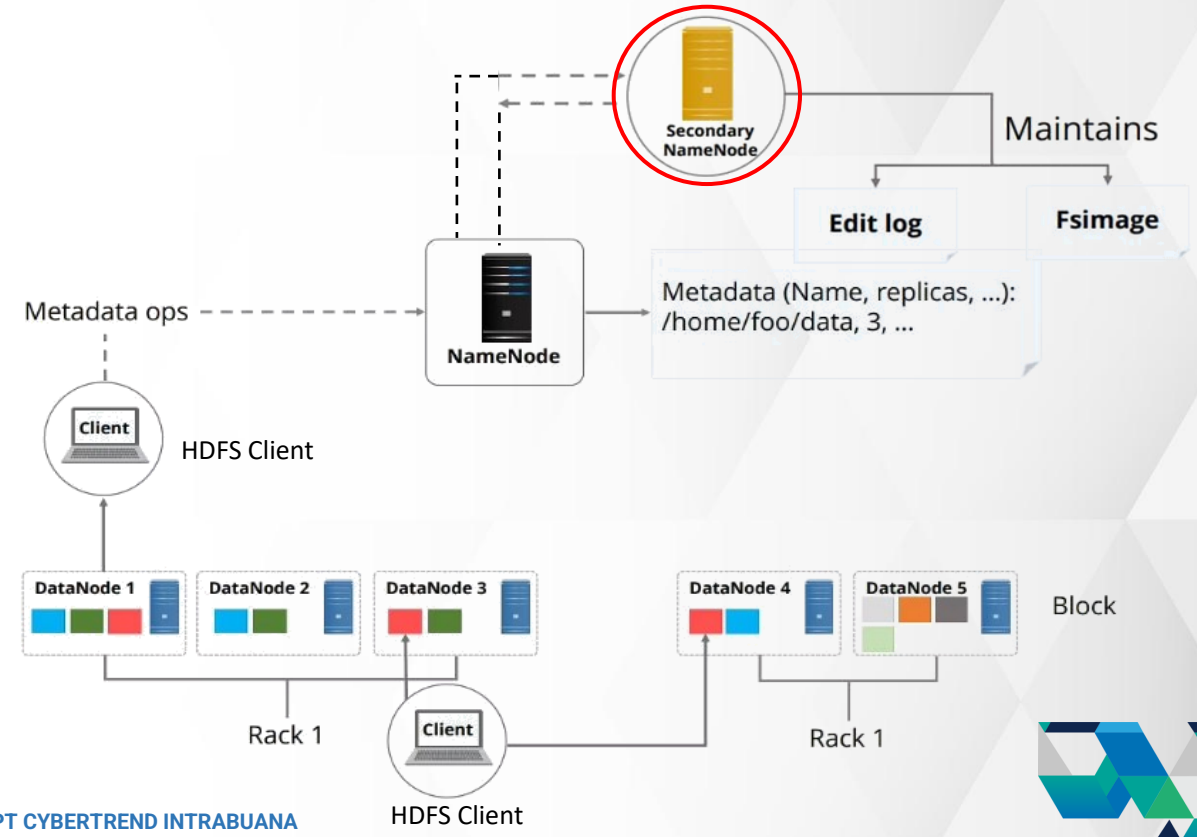


Secondary NameNode

The Secondary NameNode acts as a helper to the primary NameNode, primarily responsible for merging the EditLogs with the current filesystem image (FsImage) to reduce the potential load on the NameNode. It creates checkpoints of the namespace to ensure that the filesystem metadata is up-to-date and can be recovered in case of a NameNode failure.

Key Responsibilities:

- Merging EditLogs with FsImage to create a new checkpoint.
- Helping to manage the NameNode's namespace metadata.

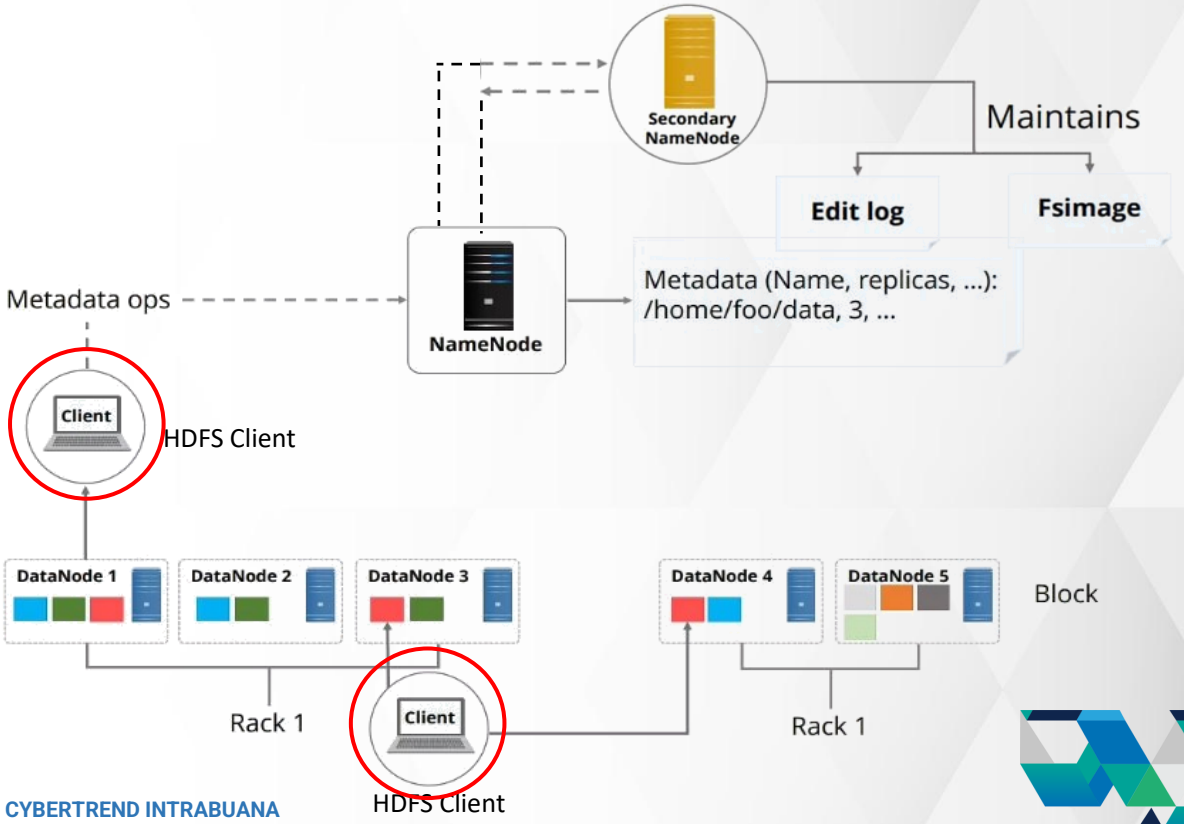


HDFS client

The HDFS client is the interface through which users and applications interact with the HDFS. It allows for file creation, deletion, reading, and writing operations. The client communicates with the NameNode to determine which DataNodes hold the blocks of a file and interacts directly with the DataNodes for actual data read/write operations.

Key Responsibilities:

- Facilitating interaction between the user/application and HDFS.
- Communicating with the NameNode for metadata and with DataNodes for data access.

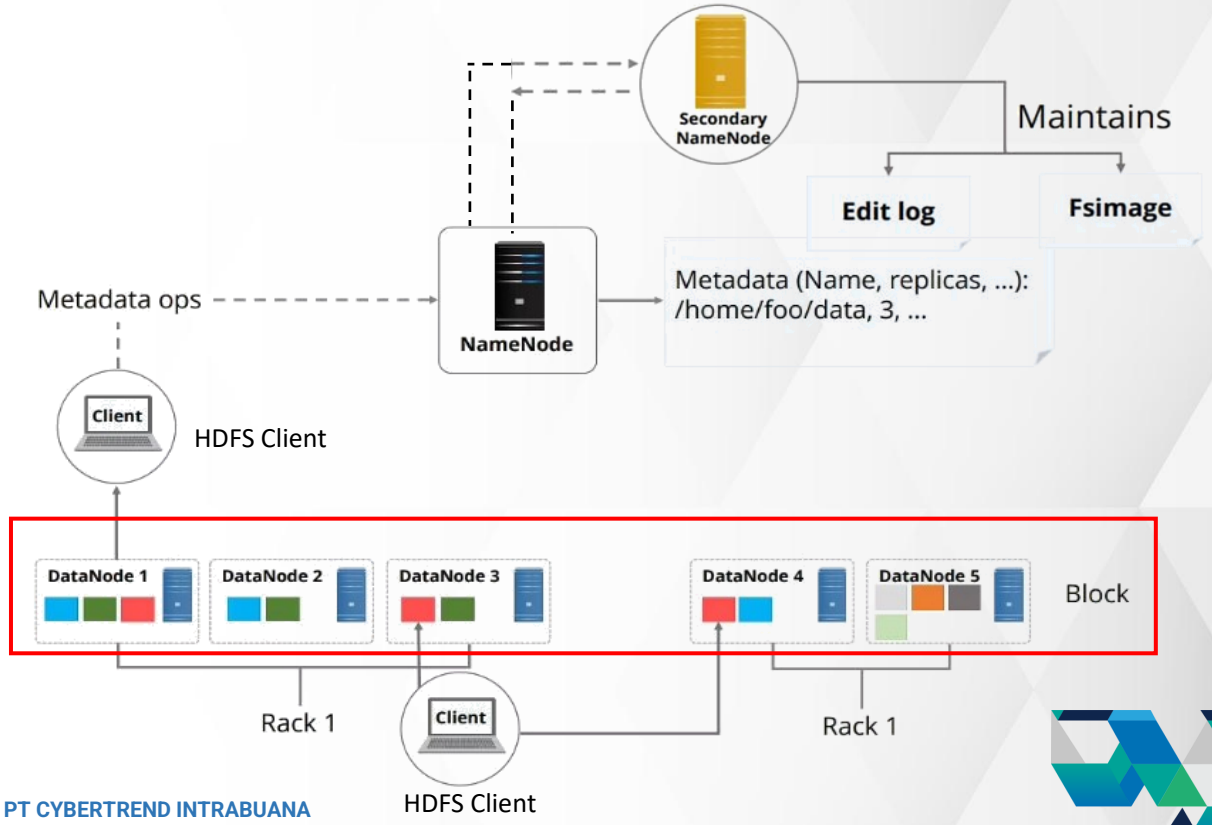


Block Structure

HDFS stores files by dividing them into large blocks, typically 128MB or 256MB in size. Each block is stored independently across multiple DataNodes, allowing for parallel processing and fault tolerance. The NameNode keeps track of the block locations and their replicas.

Key Features:

- Large block size reduces the overhead of managing a large number of blocks.
- Blocks are replicated across multiple DataNodes to ensure data availability and fault tolerance.



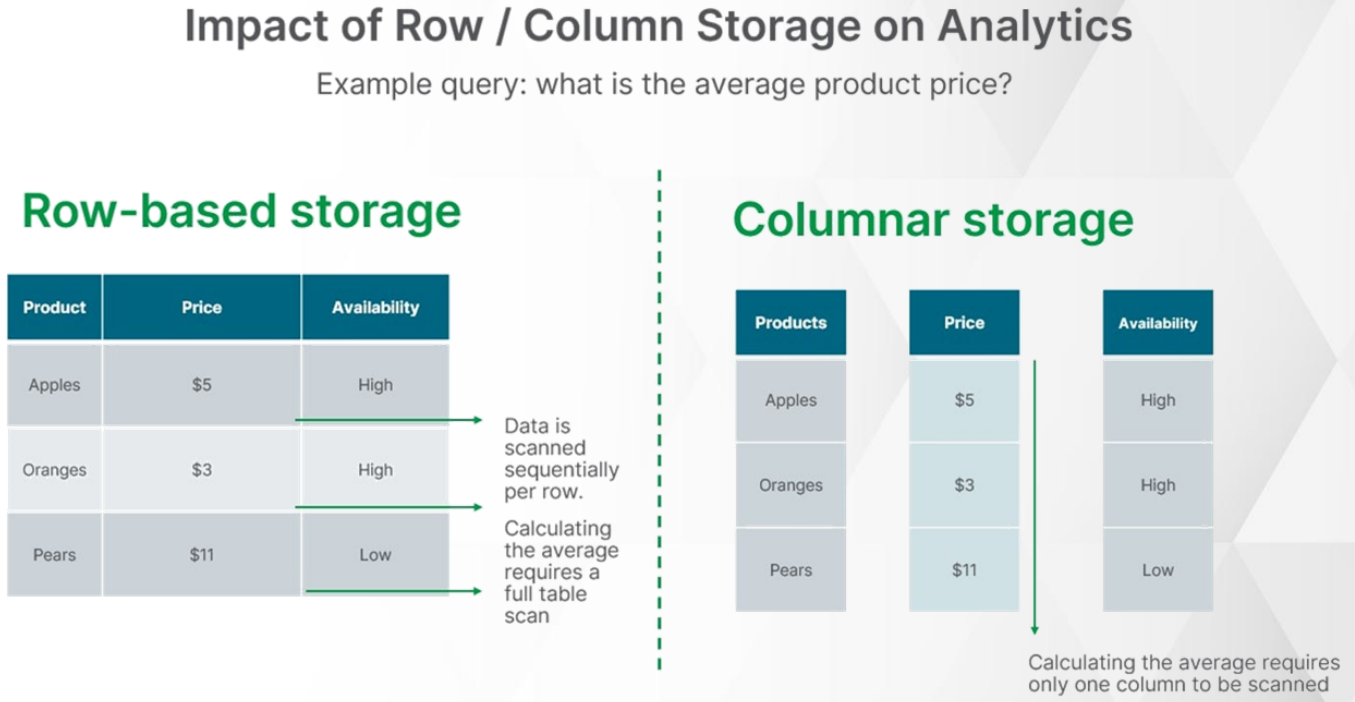
File Parquet vs File CSV

Apache Parquet is a data file format designed to support fast and efficient data processing and retrieval for complex data, with several notable characteristics:

1. Columnar: Unlike row-based formats such as CSV or Avro, Apache Parquet is column-oriented. Let's take a moment to explain what this means.

2. Open-source: Parquet is free to use and open sourced under the Apache Hadoop license, and is compatible with most Hadoop and other modern data processing frameworks.

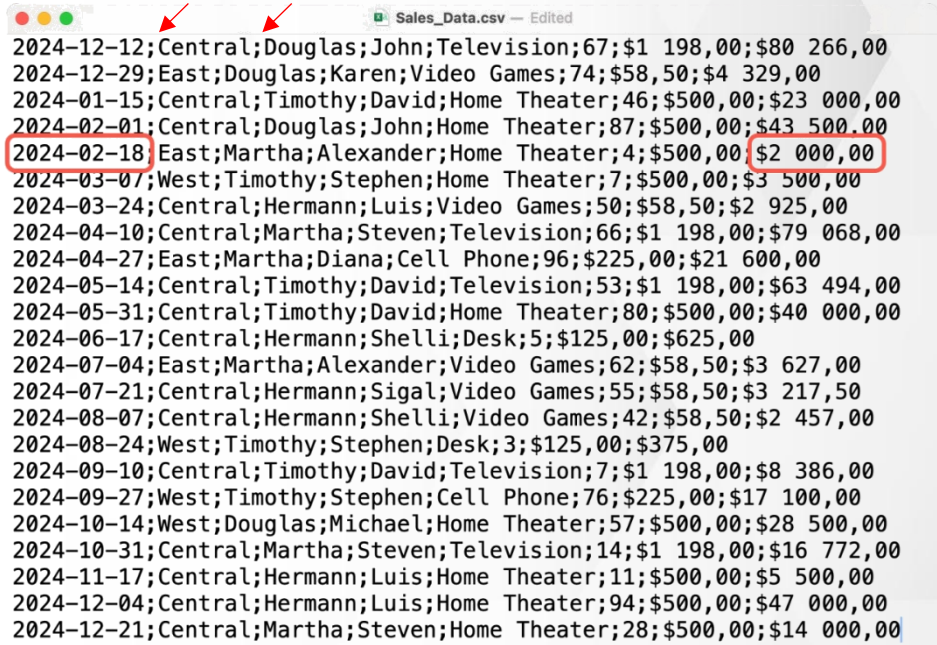
3. Self-describing: In addition to data, a Parquet file contains metadata including schema and structure. Each file stores both the data and the standards used for accessing each record – making it easier to decouple services that write, store, and read Parquet files.



File Parquet vs File CSV

A CSV file, short for comma-separated values, is a plain text format that simplifies data storage and transfer. A CSV file stores data in a tabular format, where each row represents a record, and each column is separated by a comma. The easily identifiable .csv file extension signifies this file type. The structure of the file allows for seamless data transfer between systems and ensures compatibility with a wide range of software tools. A CSV file is one of the most common file types for managing structured data.

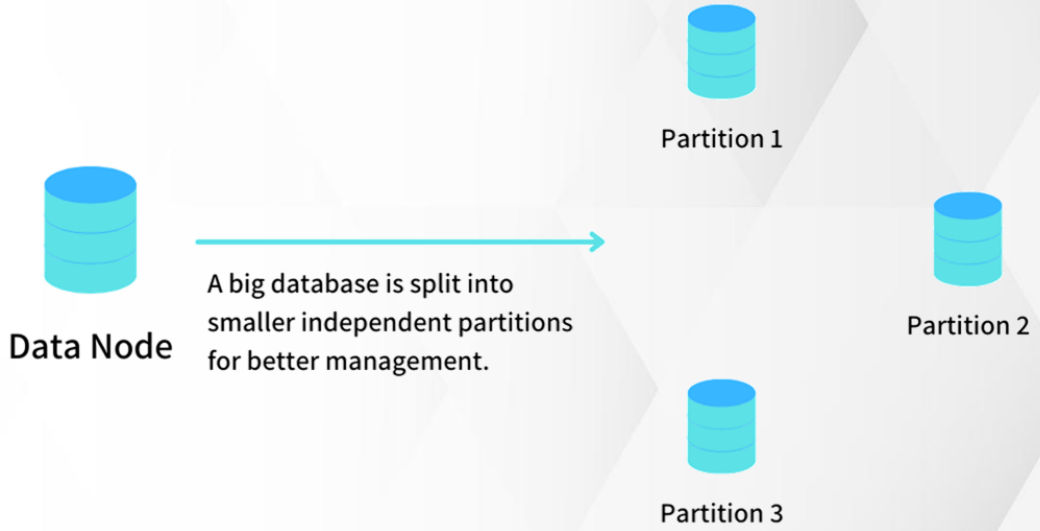
Generally, CSVs have small file sizes, which makes them great for large datasets or systems with limited storage. These sizes allow for faster file transfers and reduce overall storage costs.



Partitioning Strategy

Data partitioning is the process of splitting a dataset into more manageable, smaller pieces in order to improve efficiency, scalability, and performance.

- It can be accomplished by either **vertical partitioning**, which **separates data into columns**, or **horizontal partitioning**, which **divides data into rows according to particular criteria**.
- This method is especially helpful in databases, big data processing frameworks, and machine learning applications since it enables quicker query execution, simpler management of massive datasets, and better resource use.

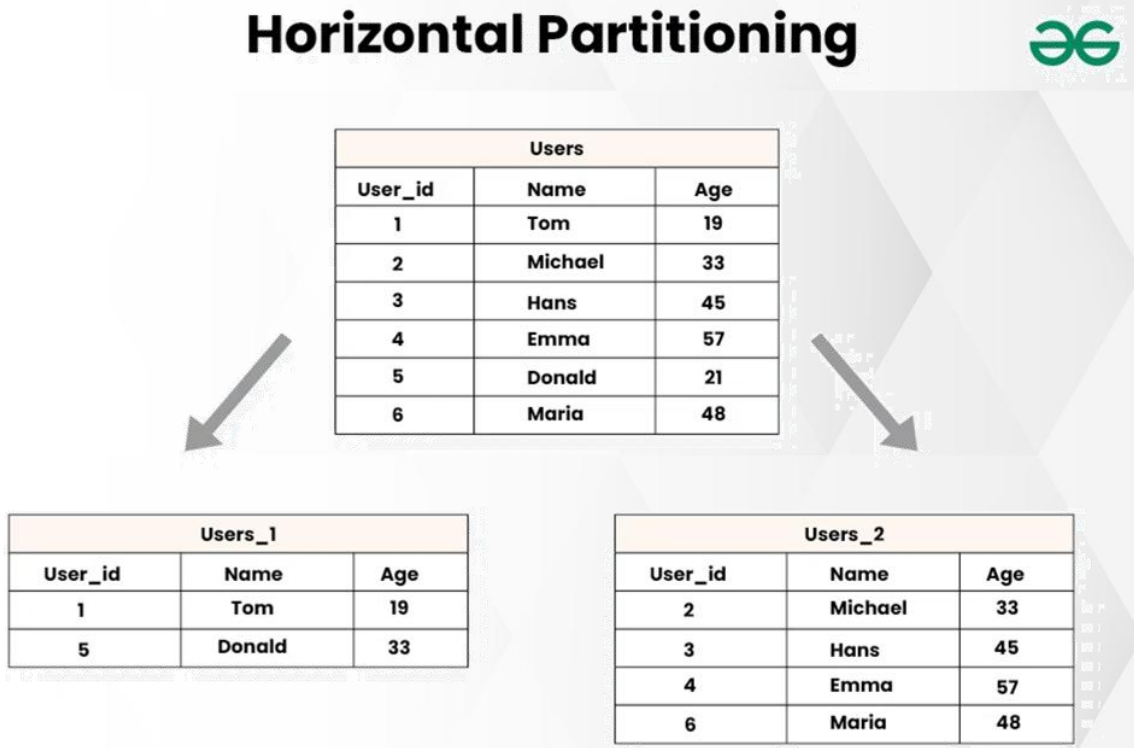


Partitioning Strategy

1. Horizontal Partitioning/Sharding

Horizontal Partitioning divides data by rows, but all partitions may still exist on the same server. When these horizontal partitions are placed across multiple servers, the approach is called Sharding.

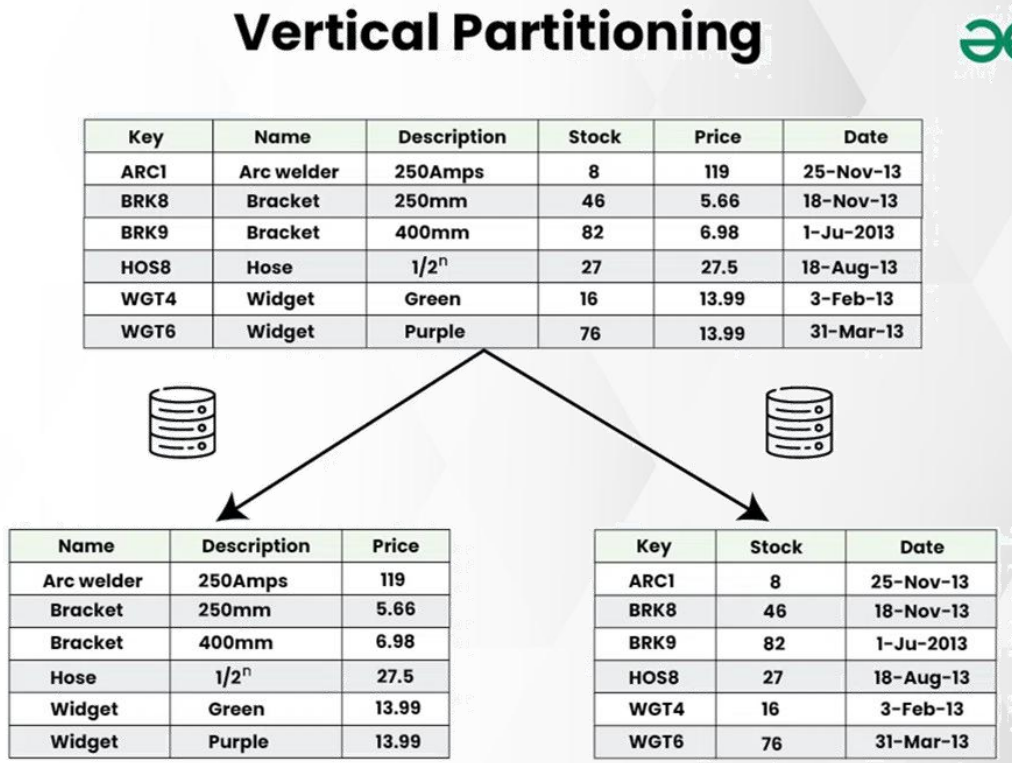
For example, user records might be divided into different shards based on geographic regions or user IDs. This approach enables efficient load distribution and improved query performance, as queries can be directed to specific shards rather than scanning the entire dataset.



Partitioning Strategy

2. Vertical Partitioning

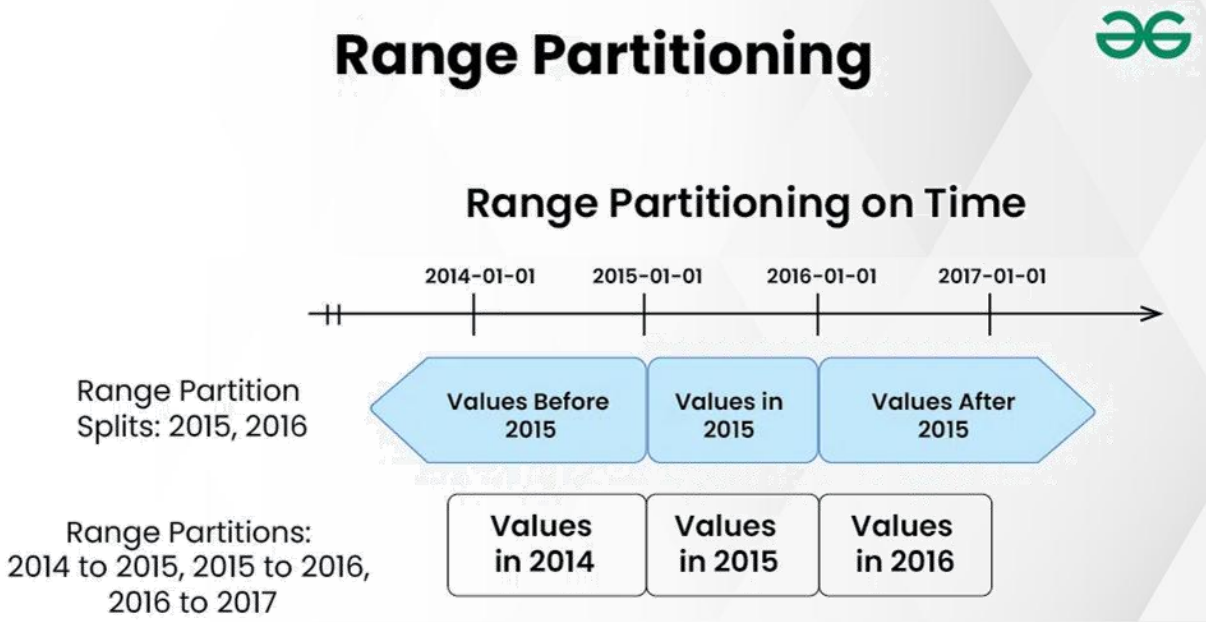
Vertical partitioning separates the dataset according to columns or attributes, in contrast to horizontal partitioning. Each partition in this method has a subset of columns for every row. When certain columns are visited more frequently than others or when different columns have different access patterns, vertical partitioning might be helpful.



Partitioning Strategy

3. Range Partitioning

This strategy organizes data into partitions based on specific ranges of values for a partitioning key. For example, a sales database might partition data by date ranges, allowing efficient queries for a specific period. Range partitioning is advantageous when dealing with time-series data or when queries often filter data by ranges.



Partitioning Strategy

4. Hase-based Partitioning

Hash partitioning involves applying a **hash function** to a specified key attribute to determine which partition will hold a given record. This method aims to evenly distribute data across partitions, minimizing the likelihood of hotspots where one partition receives a disproportionately high amount of traffic. Hash partitioning is beneficial for workloads with unpredictable access patterns.

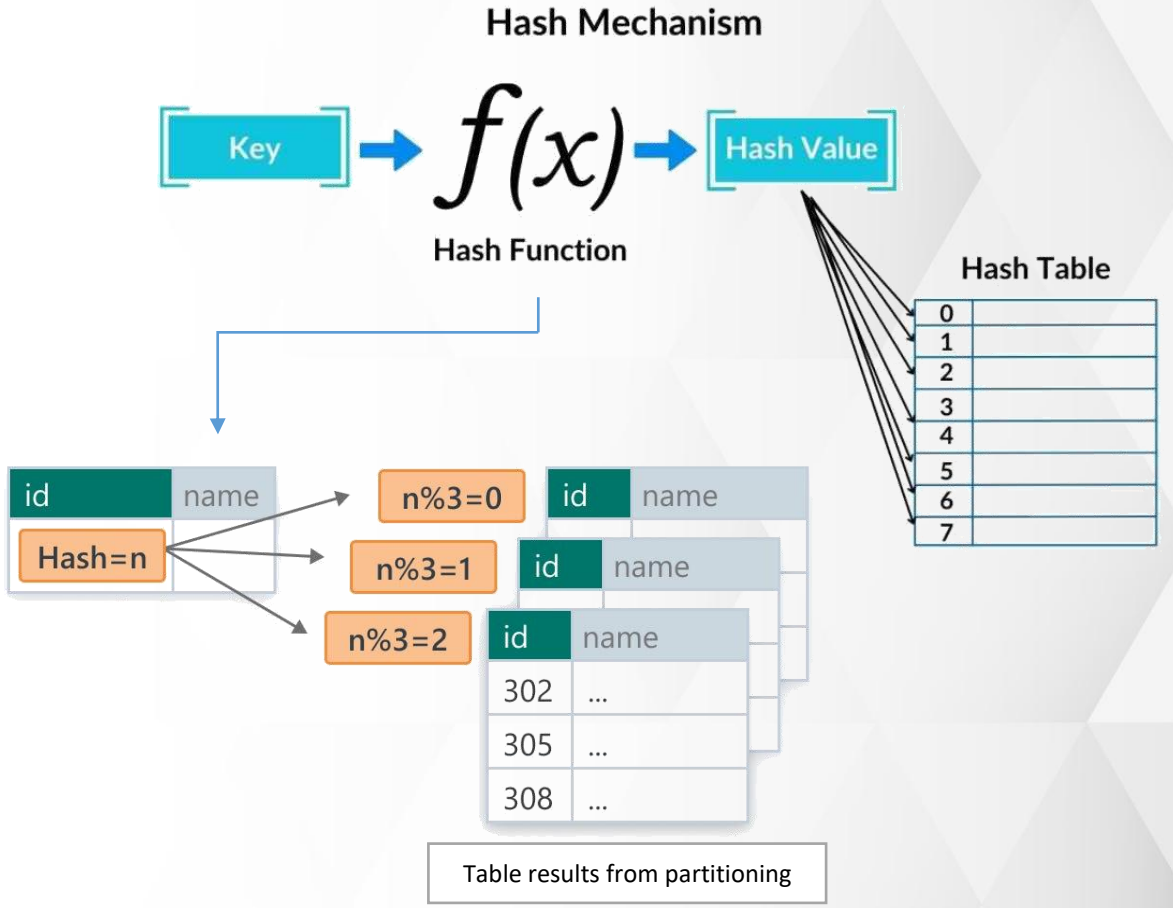


Table results from partitioning

Reading with HIVE

HIVE

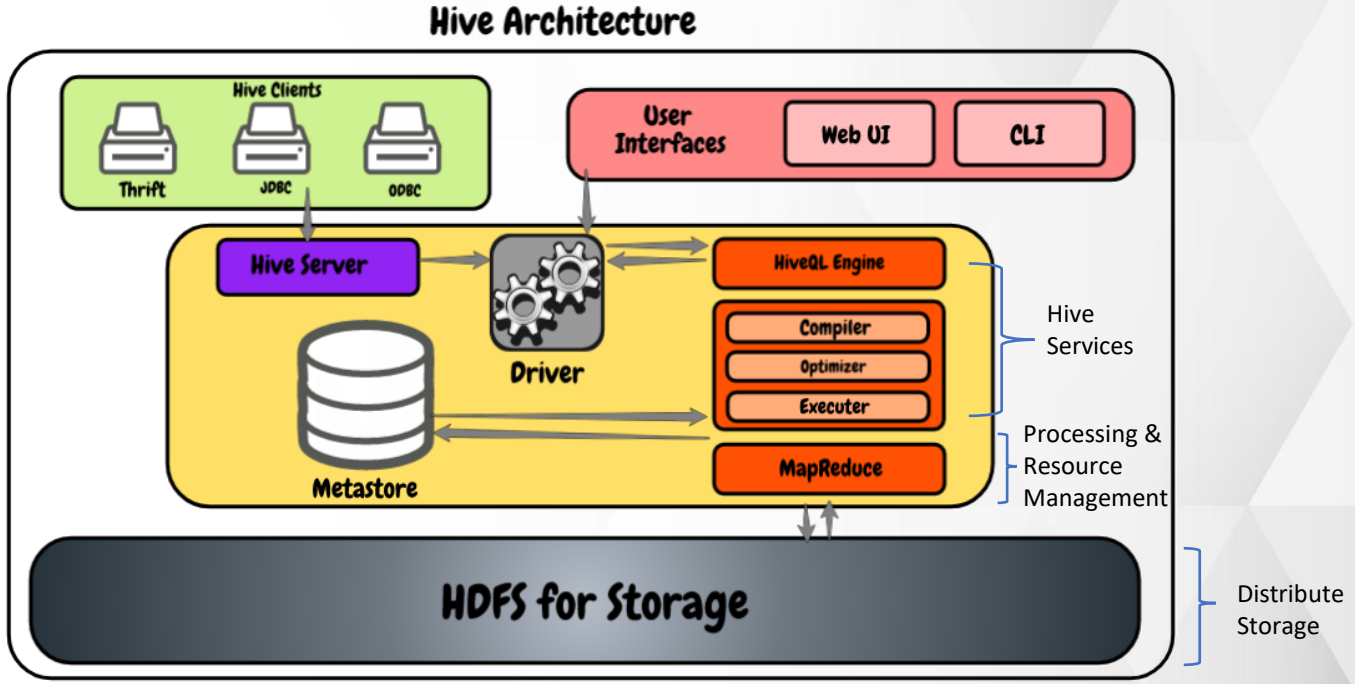
Apache Hive is a data warehouse system developed by Facebook to process a huge amount of structure data in Hadoop. We know that to process the data using Hadoop, we need to right complex map-reduce functions which is not an easy task for most of the developers. Hive makes this work very easy for us.

Hive programs are written in the Hive Query language, which is a declarative language similar to SQL. Hive translates hive queries into MapReduce programs

Hive queries can be used to replace complicated java MapReduce programs with structured and semi-structured data processing and analyses. A person who is knowledgeable about SQL statements can write the hive queries relatively easily.

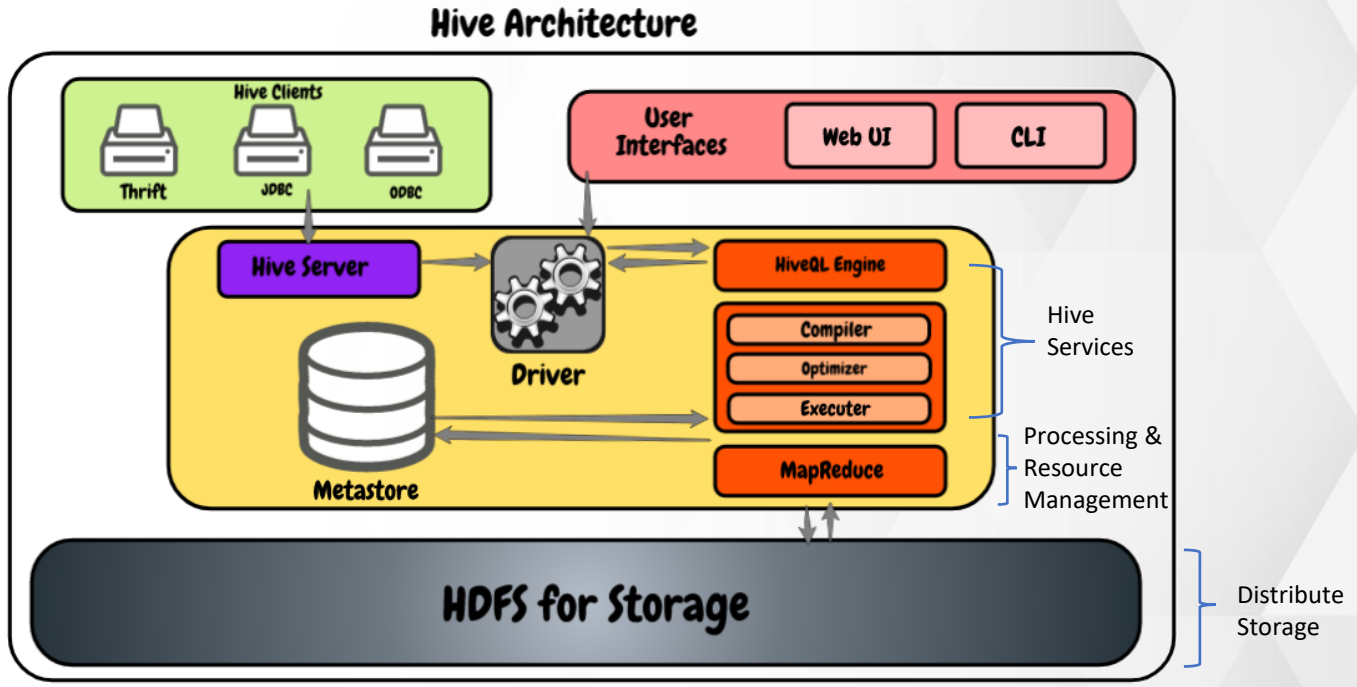
The Hive platform makes it simple to perform tasks like:

- Large scale data analysis
- Perform Ad-hoc queries
- Perform data encapsulation



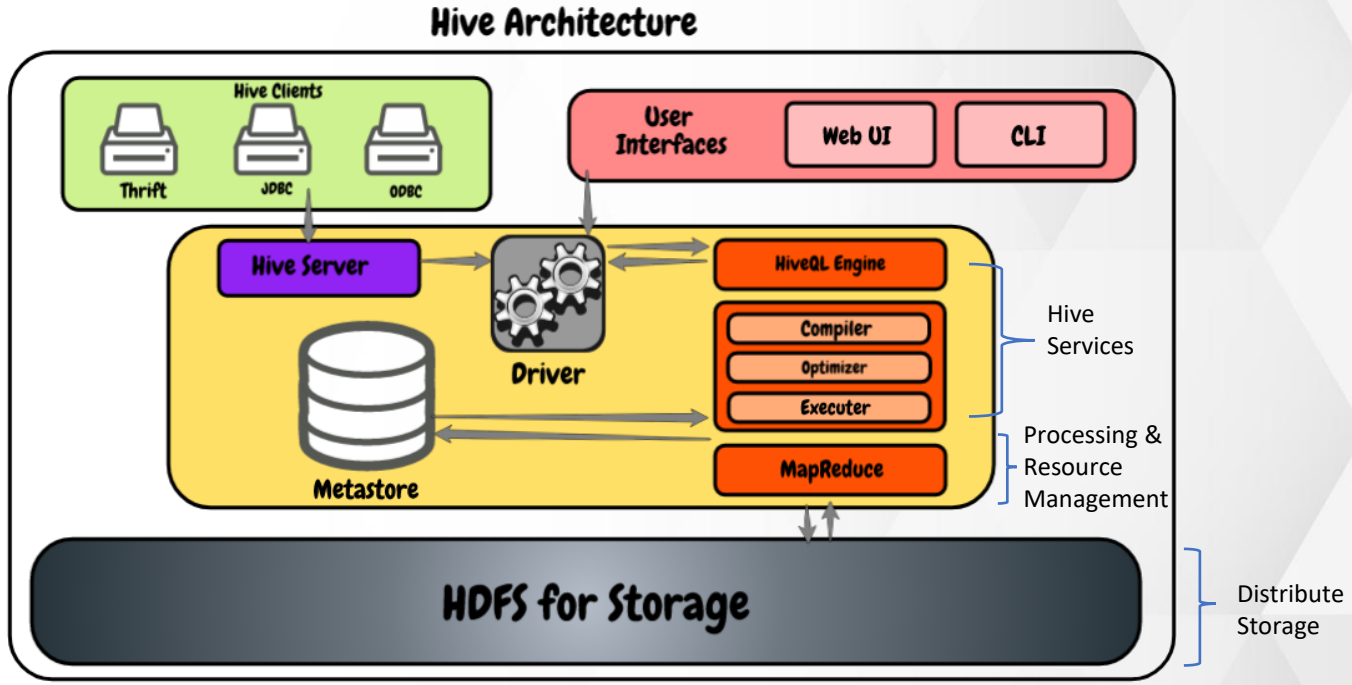
HIVE Architecture

- **Hive Clients:** It allows us to write hive applications using different types of clients such as thrift server, JDBC driver for Java, and Hive applications and also supports the applications that use ODBC protocol.
- **Hive Services:** As a developer, if we wish to process any data, we need to use the hive services such as hive CLI (Command Line Interface). In addition to that hive also provides a web-based interface to run the hive applications.



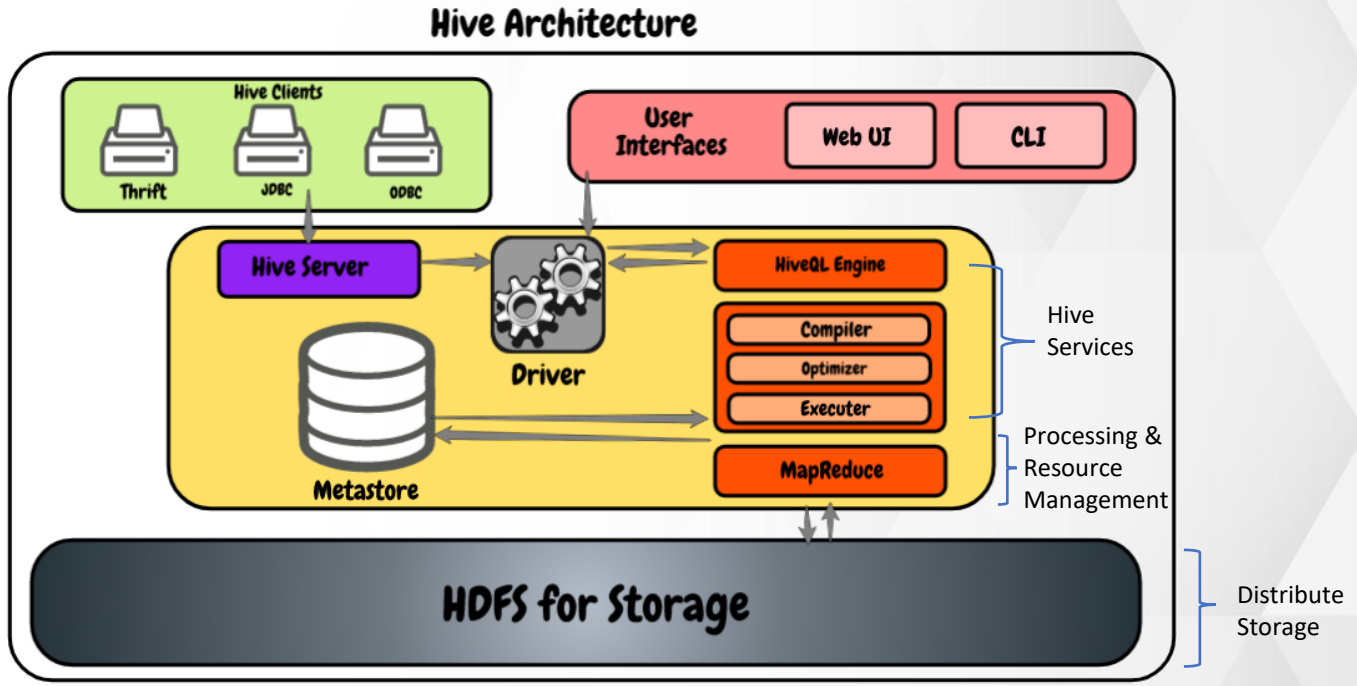
HIVE Architecture

- **Hive Driver:** It is capable of receiving queries from multiple resources like thrift, JDBC, and ODBS using the hive server and directly from hive CLI and web-based UI. After receiving the queries, it transfers it to the compiler.
- **HiveQL Engine:** It receives the query from the compiler and converts the SQL like query into the map-reduce jobs.

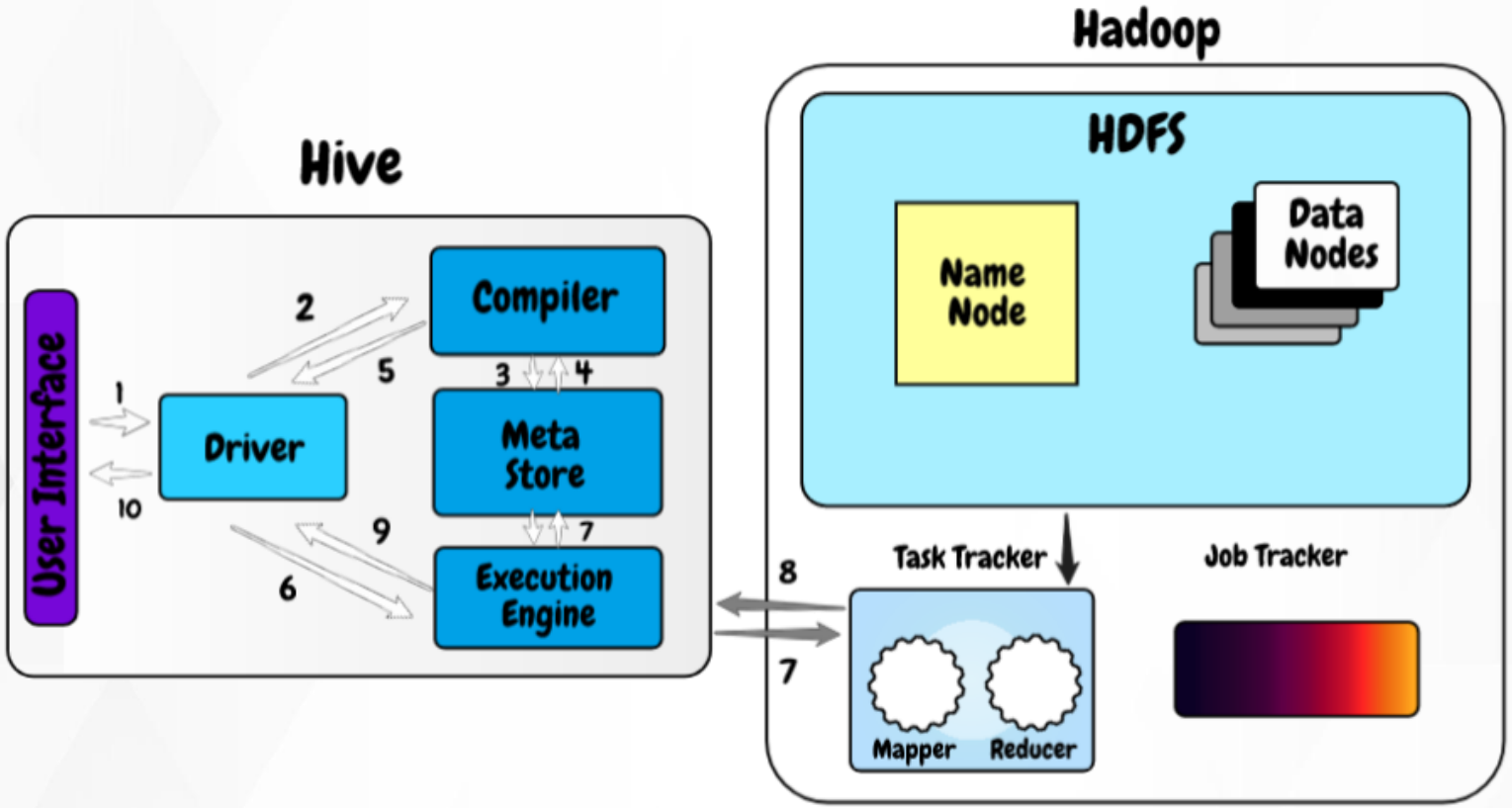


HIVE Architecture

- **Meta Store:** Here hive stores the meta-information about the databases like schema of the table, data types of the columns, location in the HDFS, etc
- **HDFS:** It is simply the Hadoop distributed file system used to store the data.



HIVE Architecture

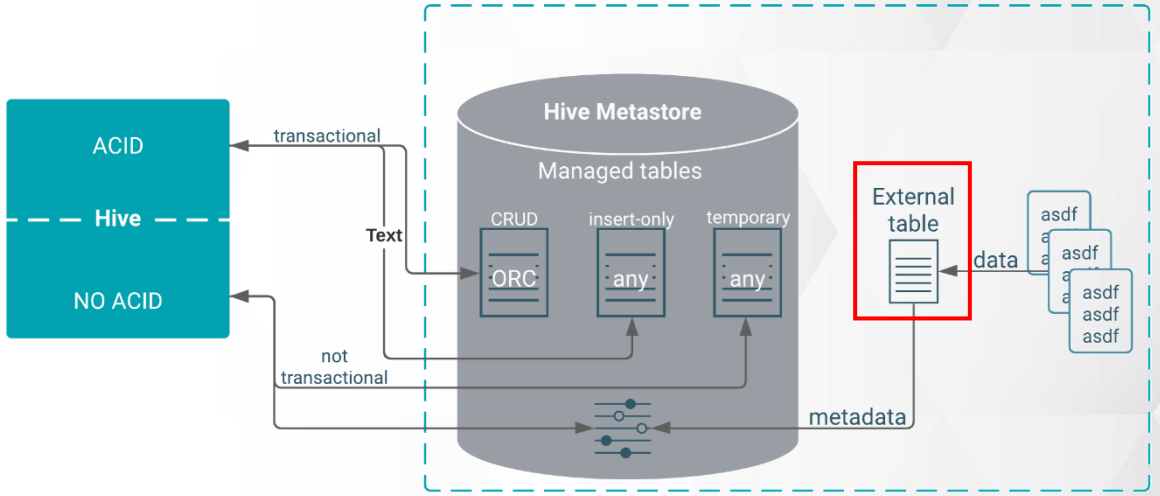


Hive can be used to manage structured data on the top of Hadoop. The data is stored in the form of a table inside a database. In Hive, the user is allowed to **create Internal** as well as **External tables** to manage and store data in a database.

External Table

External tables are an **excellent way to manage data** on the Hive **since Hive does not have ownership** of the data stored inside External tables. In case, if the user **drops** the External tables then only the metadata of tables will be removed and the data will be **safe**.

All the use cases where shareable data is available on HDFS so that Hive and other Hadoop components like Pig can also use the same data External tables are required. The metadata for External tables is managed by Hive but these tables take data from other locations on our HDFS.



Integration Patterns

Typically the data ingestion process involves following scenarios to add new set of data to the Hadoop layer.

Scenarios	Implementation Methods
Complete Load: In this method, entire table/partition is truncated and then added with new full set of data.	The primary method for a complete load in Hive is the LOAD DATA command using the **OVERWRITE** keyword.
Append Load: New data is appended to existing data for each batch of data refresh.	**Method 1: Using** INSERT INTO **Method 2: Using** LOAD DATA INPATH
Insert or Update Ingestion: In this method, data with new key is inserted to the table, whereas if the relevant key already exists in partition/table then record is updated with latest info.	The standard and most efficient pattern for "insert or update" ingestion in Hive is using **ACID transactional tables** and the **MERGE** SQL statement, or by implementing an **"intermediate table with deduplication"** approach.

References

https://cloud.google.com/learn/what-is-data-integration

https://www.ibm.com/think/topics/data-integration

https://www.domo.com/learn/article/data-integration-techniques

https://www.qlik.com/us/data-migration

https://www.ibm.com/think/topics/data-warehouse

https://www.ibm.com/think/topics/data-consolidation

https://www.ibm.com/think/topics/data-synchronization

https://www.databricks.com/glossary/hadoop

https://www.databricks.com/glossary/hadoop-distributed-file-system-hdfs

https://www.geeksforgeeks.org/system-design/data-partitioning-techniques/

https://www.adobe.com/acrobat/resources/document-files/what-is-a-csv-file.html

https://www.qlik.com/blog/what-is-the-parquet-file-format-use-cases-and-benefits

https://www.analyticsvidhya.com/blog/2020/10/getting-started-with-apache-hive/

Google Classroom

Access Jupyter Notebook

Command: ssh -L 9870:localhost:9870 -L 8888:localhost:8888 user1@103.93.236.91

Password: Cbi2026!@#

Token: mytoken

CYBERTREND LEARNING CENTER

GRAMEDIA ACADEMY

MAKARA UI ACADEMY
Upgrade Yourself, Make Your Future

Thank You.