# Unit # 4

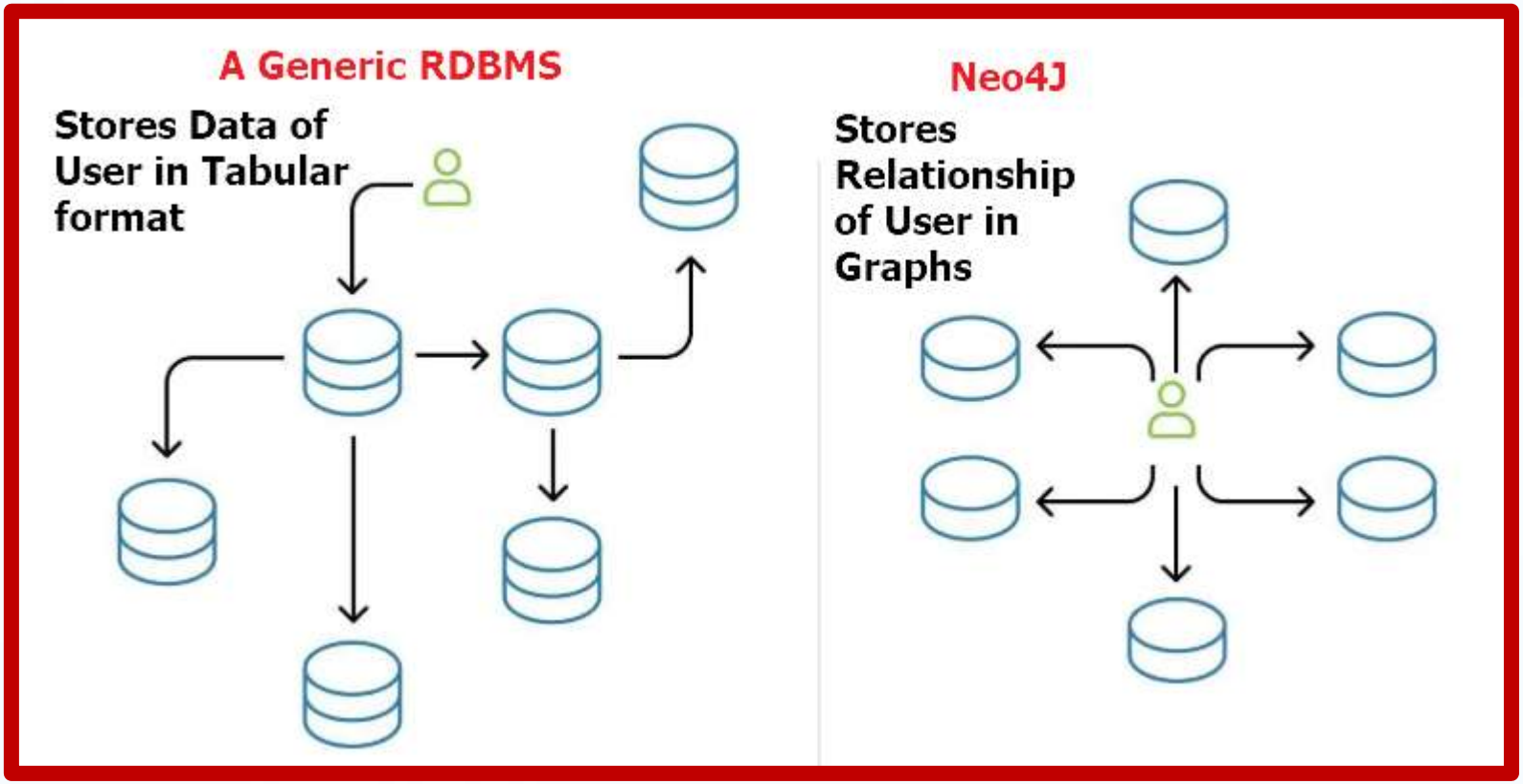
- Neo4J,JENA
- NewSQL & NuoSQL,
- Pig , Pig Data Model & Grunt
- Data types and file formats ( Read from ppt 5 slide# 8-10 )
- HiveQL data definition, HiveQL data manipulation , HiveQL queries

# Short Notes – Unit:4

In the context of Big Data Analytics, managing, querying, and analyzing massive and complex datasets requires specialized databases and frameworks. Neo4j, Jena, NeoSQL, and NewSQL cater to different aspects of Big Data, each offering unique capabilities to tackle specific challenges in analytics.

## 1. Neo4J

Neo4j is commercially avaialble, fully supported ,open sourced database that was created in 2007. It is a No-SQL database that's Java based, schema optional and massively scalable. It's commonly viewed as the best enterprise-ready graph database available today.
A  Graph Databases store data in the form of graphs. A graph is a mathematical concept that classifies elements in terms of vertices (nodes) and edges (relationships). A graph databases is a relatively new class of database that focus more heavily on the relationships that are often hidden among the many elements within masses of data. This comes from the fact that we live in a world that's is highly connected. As a result, our data is also is more connected. In many instances thus the value of the relationships inside the data is more valuable than the data itself.
The graph databases tries to use this relationships in data and has this unique ability to surface new interconnected knowledge, natively and at scale, as analytical insights that have material impacts on businesses and other organizations.
This whole concept is visually representable in the diagram on next slide.




**Key Features :**

*Graph Queries:*

Neo4J is optimized for handling billions of nodes and relationships. It provides special Cypher Query Language for exploring relationships, such as shortest paths, connected components, and community detection. In SQL if we try to reference a table or a column that doesn't exist in the schema it throws an error However with Neo4j Cypher such is not the case as Neo4j Cypher tries to match patterns & if the pattern doesn't exist, then nothing is returned.

## *Schema-Free and Flexible Data Model :*

Neo4j is schema-optional, meaning we can store data without defining a strict schema beforehand. This flexibility allows for adding new nodes, relationships, and properties dynamically without altering existing data. Managing heterogeneous data, where different nodes and relationships can have varying structures.

## *High Performance for Relationship Queries :*

Neo4J is optimised for graph traversal . Relationships are stored as direct pointers between nodes, enabling fast traversal even for deeply nested graphs. Traversals are performed in constant time (O(1)), as Neo4j does not rely on relational joins. When a query starts from a node, Neo4j smart algorithms immediately accesses connected relationships and nodes without scanning unrelated data. The performance is not disrupted even in scenario when the dataset grows and traversal performance remains consistent.

## *ACID Compliance*

Neo4j guarantees **atomicity**, **consistency**, **isolation**, and **durability (ACID)** for all transactions. Multi-step operations are executed as a single transaction, ensuring data consistency even in case of failures. If a failure occurs during the update, Neo4j ensures the entire transaction is rolled back, preventing partial updates.

## *Scalability & High Availability*

Neo4j clusters distribute the graph across multiple nodes, ensuring performance , fault tolerance & availability.

# 2. JENA

Apache Jena is an open-source framework for building Semantic Web and Linked Data applications. It provides tools and APIs for managing RDF (Resource Description Framework) data, OWL ontologies . Jena is designed to handle large-scale data integration, representation, and reasoning, making it a critical component for building knowledge graphs and linked data systems.

**Key Features :**

*RDF (Resource Description Framework) Support :*

RDF is the foundational data model for representing information in the Semantic Web. It encodes data as triplet viz

1.Subject  the entity being described
2. Predicate the property or relationship
3. Object  the value or related entity.

Jena can store and manipulate such large numbers of RDF triples. It provides APIs for creating, reading, updating, and deleting RDF triples programmatically.

*Ontology Management :*

Jena supports OWL (Web Ontology Language) and RDFS (RDF Schema), which are used to define vocabularies and schemas for RDF data. It provides tools for managing and querying ontologies, including :

1.Defining Classes  for representing concepts or entities
2. Properties  for defining relationships between entities
3. Reasoning  which means inferring new knowledge based on rules and relationships.

*Inference and Reasoning :*

Jena includes something known as ***reasoners*** to infer new knowledge from existing RDF data based on ontologies or custom rules. For eg , If an entity is a Manager and manages an Employee, then that entity also works For the same organization.

*Data Integration and Linked Data :*

Jena enables linking data across different sources using shared identifiers (URIs). It facilitates seamless integration with external RDF datasets also. It is super powerful for managing, combining, and linking structured, semi-structured, and unstructured data across distributed systems. Sources may have different formats (e.g., JSON, XML, CSV, relational databases) , misaligned concepts or terminology used in different datasets. But JENA can integrates all such large datasets and in a distributed environment.

*Integration with Big Data Frameworks :*

Jena can process RDF data at scale by integrating with Apache Hadoop and Spark. Particularly , the integration of Apache Jena with Hadoop allows organizations to process and analyze large-scale RDF datasets in a distributed environment. RDF datasets in Big Data applications often grow to billions of triplets. Hadoop provides the scalability required for storing and processing such datasets across distributed clusters. The JENA-Hadoop integration leverages a distributed computing and storage capabilities of Hadoop with Jena's Semantic Web features, such as RDF data management, reasoning, and SPARQL querying.

# 3. New SQL

NewSQL is a modern category of relational database systems designed to overcome the limitations of traditional RDBMS while retaining the strengths of SQL for managing structured data. It combines the scalability and performance of NoSQL systems with the ACID properties and familiar SQL interface of relational databases, making it an ideal choice for modern applications requiring high throughput, low latency, and strong consistency. NewSQL databases are built to address challenges faced by traditional RDBMS systems in handling Big Data, such as horizontal scalability, distributed architectures, and the need for real-time analytics.

**Key Features :**

*Horizontal Scalability :*

Unlike traditional RDBMS systems that rely on vertical scaling (adding resources like CPU, RAM, or storage to a single server), NewSQL databases achieve horizontal scaling by distributing the database across multiple nodes in a cluster. The characteristics of a big data is that data is partitioned (sharded) across nodes and hence this requires Queries and Transactions to work across distributed nodes and should process data in parallel. New SQL enable such system to handle massive data volumes and high throughput workloads.

*ACID Compliance :*

NewSQL databases are fully ACID-compliant, ensuring atomicity, consistency, isolation, and durability for transactions even in distributed environments.

## *SQL Interface :*

NewSQL databases retain the familiar SQL interface for querying and managing data, providing compatibility with existing tools, applications, and developer skills. Its support for standard SQL queries, joins, transactions, and stored procedures, ensuring developers do not need to learn new query languages or paradigms. SQL commands such as SELECT, INSERT, UPDATE, DELETE, and DDL statements like CREATE TABLE and DROP TABLE are also supported in NEWSQL.

## *Distributed Architecture :*

One of the critical features of NewSQL databases is its distributed architecture, which enables horizontal scalability, fault tolerance, and high availability while maintaining the strengths of traditional relational databases, such as ACID compliance and SQL support. Unlike traditional RDBMS systems that rely on centralized architectures, NewSQL databases distribute data and query processing across multiple nodes in a cluster. In this , sharding divides the database into smaller, manageable pieces called **shards**, with each shard containing a subset of the data. Data is replicated across nodes for fault tolerance and availability and Queries are broken into smaller tasks and executed in parallel on the relevant nodes.

## *Real-Time Analytics*

NewSQL databases also support real-time analytical queries on live transactional data, eliminating the need for separate systems. Thus NoSQL accelerates decision-making even in real time mode which is its very attractive feature.

# 4. NUO SQL

NuoSQL, also known as NuoDB, is a distributed SQL database designed to handle the challenges of modern cloud-native applications. It combines the scalability and elasticity of NoSQL databases with the ACID compliance and familiarity of SQL found in traditional relational databases. NuoDB is particularly well-suited for applications requiring continuous availability, distributed operations, and seamless scaling across cloud environments.

**Key Features :**

*Distributed Architecture :*

NuoSQL operates as a distributed database system, ensuring scalability, fault tolerance, and high availability. It decouples storage and compute layers, allowing for independent scaling and efficient resource utilization. Its components are

*Transaction Engines (TE):* Handle SQL query execution and transaction processing. It is stateless and can be added or removed dynamically to adjust compute capacity.

*Storage Managers (SM):* Manage the physical storage of data. It is responsible for ensuring data durability and fault tolerance.

TEs and SMs work together to create a shared, distributed system. Queries and transactions are distributed across nodes, ensuring balanced workload and fast processing.

*Elastic Scalability :*

NuoSQL is designed to scale out seamlessly by adding or removing nodes in the cluster without service interruption. TEs can be added smoothly to increase the Query

rate processing and without hindering the normal operation . Similarly , adding a new Storage Manager (SM) increases storage capacity and fault tolerance. It thus offers a real-time scaling to handle workload spikes. During peak hours, an e-commerce platform can add TEs to handle increased transactional loads.

## *SQL Compliance :*

Just like New SQL , NuoSQL also provides full SQL support, enabling developers to use familiar SQL syntax for querying and managing data. It offers full support to all SQL features like :

Data Manipulation: SELECT, INSERT, UPDATE, DELETE

Data Definition: CREATE TABLE, ALTER TABLE, DROP TABLE

## *Cloud-Native Design :*

The cloud-native design of NuoSQL (NuoDB) is one of its standout features, enabling it to seamlessly support modern cloud-based applications. Built to leverage the scalability, elasticity, and flexibility of cloud environments, NuoSQL aligns with the principles of microservices architectures and containerized deployments, making it ideal for dynamic and distributed workloads. This is possible since NuoSQL separates its architecture into compute and storage layers, allowing each to scale independently. The Transaction Engines (TE), nodes handle SQL queries and transaction processing while Storage Managers (SM) nodes manage the durable storage of data. As explained above TEs can be dynamically added or removed to scale compute capacity as needed handling spikes in query load. Adding SMs cater to increased storage capacity or higher data redundancy.

# Comparing NewSQL & NuoSQL

NewSQL and NuoSQL (NuoDB) both aim to solve scalability, performance, and availability challenges of traditional relational databases while retaining SQL and ACID compliance. However, NuoSQL (a specific implementation of NewSQL) has unique architectural and operational features designed for cloud-native and hybrid environments, which distinguish it from general NewSQL databases. Following is the key differences between the them :

| NewSQL | NuoSQL |
|---|---|
| Primarily designed for distributed environments, with some systems supporting cloud-native features. | Fully cloud-native, designed for dynamic cloud environments with multi-cloud and hybrid support. |
| General distributed architecture with shared-nothing or shared-storage designs. | Decouples compute (Transaction Engines) and storage (Storage Managers) layers for independent scaling. |
| Scaling can require reconfiguration in some implementations. | True elastic scaling, allowing addition removal of compute or storage nodes real time without downtime based on workload. |
| CockroachDB, Google Spanner, TiDB. | NuODB (exclusive) |

# Apache PIG and GRUNT

Apache **Pig** and **Grunt** are integral components of the Hadoop ecosystem that simplify the process of managing and analyzing massive datasets stored in the Hadoop Distributed File System (**HDFS**). **Pig** provides a high-level abstraction for processing data, while **Grunt** serves as its interactive command-line interface (CLI). Together, they enable developers to efficiently create, execute, and debug data workflows.

## 1. Apache PIG :

Pig is a high-level platform or tool which is used to process the large datasets. It provides a high-level scripting language, known as *Pig Latin* which is used for processing the MapReduce. One limitation of MapReduce is that the development cycle is very long. Writing the reducer and mapper, compiling packaging the code, submitting the job and retrieving the output is a time-consuming task. Apache Pig reduces the time of development using the multi-query approach. Also, Pig is beneficial for programmers who are not from Java background. 200 lines of Java code can be written in only 10 lines using the Pig Latin language.

**Key Features**

- Easy to learn, read and write. Especially for SQL-programmer, Apache Pig is a boon.
- Provides rich sets of operators like the filtering, joining, sorting, aggregation etc.
- Integrates with other components of the Apache Hadoop ecosystem, such as Apache Hive, Apache Spark, and Apache ZooKeeper, Apache Pig enables users to take advantage of these components' capabilities while transforming data.
- Pig can handle the analysis of both structured and unstructured data.

# Apache PIG and GRUNT (contd)

Data Models in PIG:

## 1. Atom

An Atom represents a single data value. It is the simplest data type in Pig's data model. Atoms can hold values of primitive types such as int, long, float, double, chararray (string), and bytearray. Examples

string: "John“ , integer: 42 , float: 3.14

## 2. Tuple

A Tuple is a collection of fields, each of which can hold an Atom or another complex data structure.It is similar to a row in a table where each field has a specific value and type. Tuples are enclosed in parentheses: (field1, field2, field3, ...). Example

(John, 42, 3.14)

## 3. Bag

A Bag is a collection of tuples, allowing duplicate tuples.Bags are unordered and are the equivalent of a table in Pig. Bags are enclosed in curly braces: {(tuple1), (tuple2), ...}. Example

{ (John, 25, 50000.0), (Alice, 30, 60000.0), (Bob, 35, 70000.0) }

## 4. Map

A Map is a collection of key-value pairs, where the key is a chararray (string) and the value can be an Atom or a complex data type (e.g., a Tuple or Bag).Maps are similar to dictionaries or hash tables in other programming languages. Maps are enclosed in square brackets: [key#value, key#value, ...]. Example

[name#John, age#25, salary#50000.0]

# Apache PIG and GRUNT(contd)

## 2. GRUNT :

Grunt is the interactive command-line shell for Apache Pig, designed to simplify working with Pig scripts and commands. It allows users to execute Pig Latin statements interactively, debug data workflows, and manage Hadoop data stored in HDFS. Grunt acts as a bridge between users and the Apache Pig engine, offering an intuitive environment for creating, testing, and refining data transformation pipelines. Since Pig translates Pig Latin scripts into MapReduce jobs, Grunt indirectly interacts with the Hadoop MapReduce framework.

**Key Features**

*Integration with Pig Latin:*

Grunt is built to interpret Pig Latin commands, the scripting language of Apache Pig.It allows users to execute Pig Latin statements interactively, making it a natural companion to Pig. It specifically facilitate **testing**, **debugging**, and **execution** of Pig Latin scripts, which are core to Apache Pig's data processing workflows.

*Pig-Specific Debugging Tools:*

Commands like EXPLAIN and ILLUSTRATE in Grunt are tailored to analyze and debug Pig scripts by showing their logical, physical, and MapReduce execution plans. These commands are specific to how Pig processes data.

*HDFS Command Extensions:*

Grunt provides shortcuts for HDFS commands like fs, ls, and rm, but only within the context of Apache Pig.

# HIVE

Apache Hive is a data warehouse infrastructure built on top of Apache Hadoop. It facilitates querying, summarization, and analysis of large datasets stored in the Hadoop Distributed File System (HDFS) using a SQL-like language called HiveQL. Hive simplifies data analytics for users familiar with SQL, enabling them to leverage Hadoop's scalability without writing low-level MapReduce programs.

Hive uses a language called HiveQL, which is similar to SQL, to allow users to express data queries, transformations, and analyses in a familiar syntax. HiveQL statements are compiled into MapReduce jobs, which are then executed on the Hadoop cluster to process the data.

**Hive Architecture & Working :**

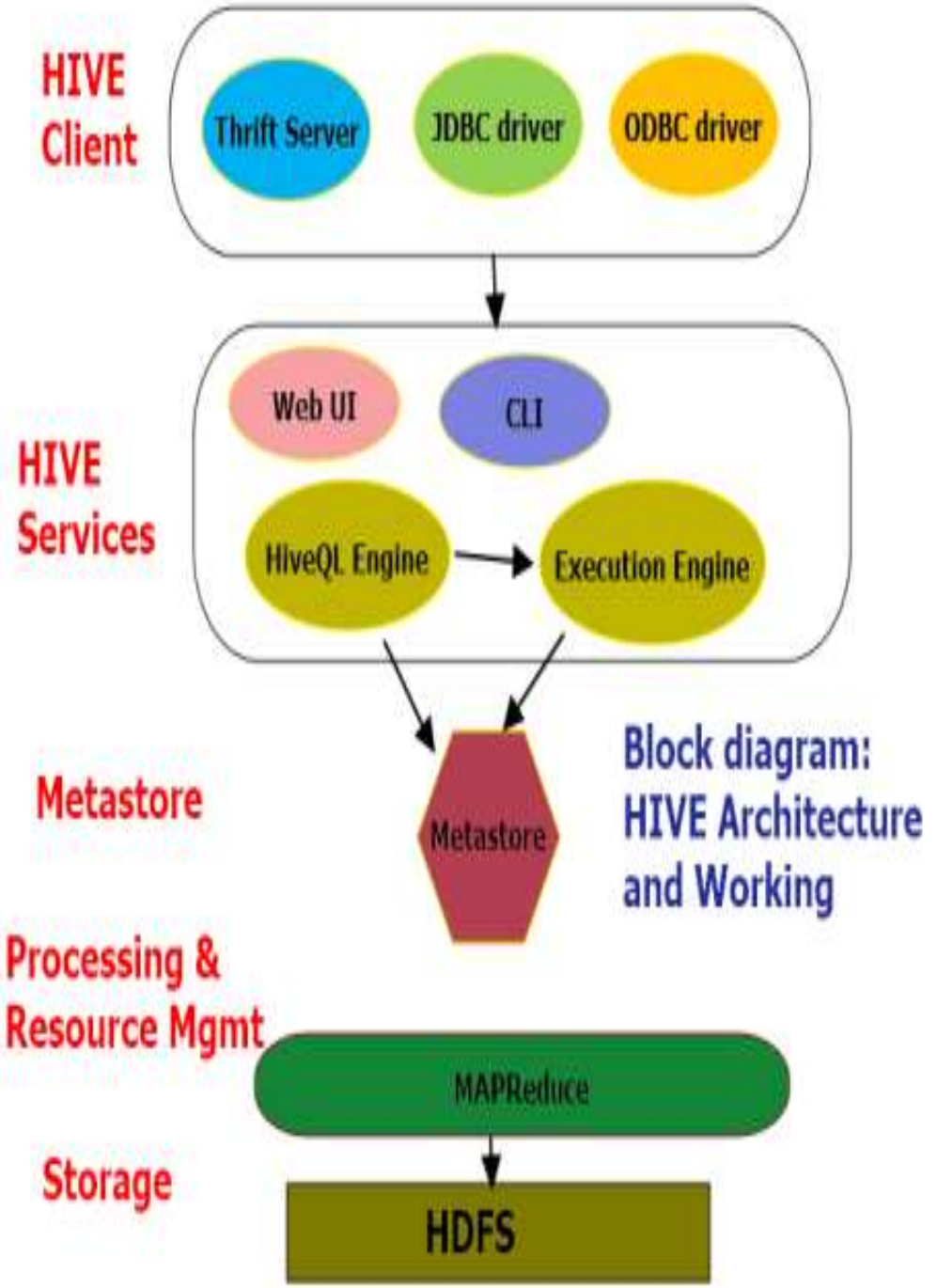
The major components of Apache Hive are:

1. Hive Client
2. Hive Services
3. Metastore
4. Processing and Resource Management
5. Distributed Storage

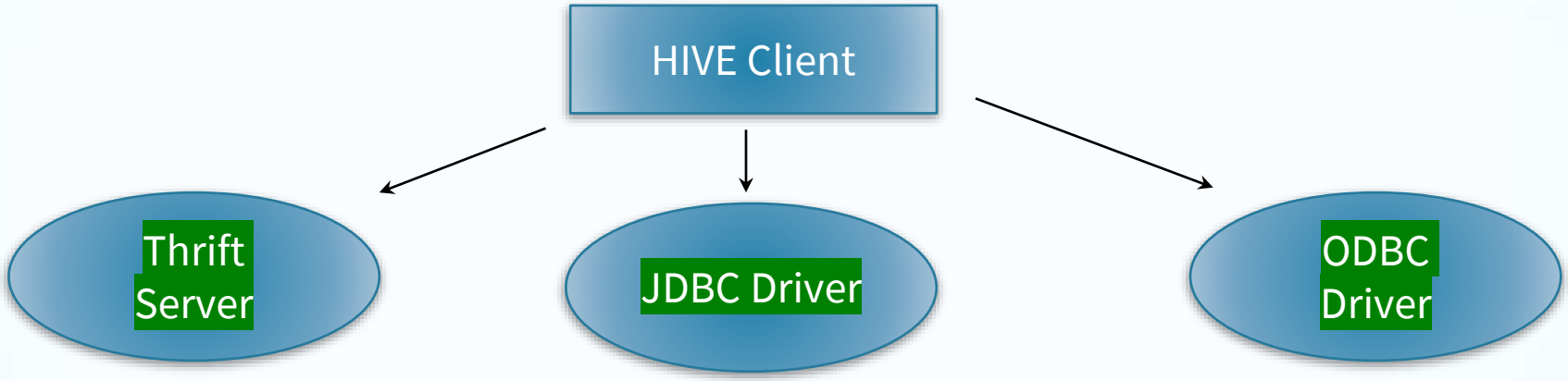
## *1. Hive Client*

Hive allows writing applications in various languages like Java, Python, and C .

Hive Client are categorised in to three types as shown & explained on next slide :



# HIVE (contd)




Thrift server is a cross-language service provider platform that serves the request from all those programming languages that supports Thrift.

It is used to establish a connection between hive and Java applications. JDBC driver uses Thrift to communicate with the Hive Server.

Similar to the JDBC driver, the ODBC driver uses Thrift to communicate with the Hive Server. It allows the applications that support the ODBC protocol to connect to Hive.

*2. Hive services : Web UI & CLI*

Apache Hive provides two primary interfaces for interacting with its data warehouse capabilities: the Web User Interface (Web UI) and the Command Line Interface (CLI). These interfaces allow users to execute HiveQL queries, manage metadata, and interact with the underlying data.

# HIVE (contd)

**Hive Web UI**

- Users can write and execute HiveQL queries directly through the browser.
- Results are displayed in the web interface.
- All the queries executed by the user are maintained as history in the UI. Users can review, re-execute, or debug previously run queries.
- User can monitor the status of Hive jobs. UI displays details like job progress, MapReduce tasks, and logs.
- However , the Hive Web UI has been deprecated in newer versions of Hive. It is replaced by monitoring and querying tools like **Hue**, **Ambari**, or **Beeline**.
- UI does not require much technical expertise to execute queries. It helps in quick exploration of Hive capabilities without needing to know command-line syntax . The dowside is it is useful for simple tasks only but lacks the full flexibility and control provided by the CLI or programmatic interfaces.

**Hive Web CLI**

- Hive CLI is a terminal-based tool that allows users to run queries, manage schemas, and execute scripts. It can run HiveQL queries interactively or execute pre-written scripts stored in .sql or .hql files.
- It provides full functionality unlike Web UI. It offers access to schema management, query execution, and advanced configurations.
- It allows direct interaction with the Hadoop ecosystem, such as viewing logs or monitoring MapReduce jobs.

# HIVE (contd)

## *2. Metastore*

The Metastore is a centralized metadata repository that stores and manages metadata about the Hive data warehouse. It enables Hive to interact with structured data store d in Hadoop Distributed File System (HDFS) or other storage systems. The Metastore keeps track of the structure, schema, and location of Hive tables and partitions, facilitating the execution of HiveQL queries without requiring the user to manually reference the underlying files or directories.

## *3. QL Processing Engine*

The HiveQL Processing Engine is responsible for interpreting and processing HiveQL queries, transforming them into an executable plan, and handing over the plan to the Hive Execution Engine for execution. It validates the query by checking schema information from the Metastore & Ensures the query is syntactically correct. It Verifies that the tables, columns, and functions used in the query exist and are accessible. It generates a logical plan and cost optimises it to a Physical Execution Plan.

## *4. Execution Engine*

The Hive Execution Engine is responsible for executing the physical plan generated by the HiveQL Processing Engine. It determines how the tasks are carried out on the Hadoop cluster. It ensures proper communication and data transfer between tasks, handling intermediate data like shuffle and sort operations. It coordinate with YARN for resource management for the query execution. Handles task failures , retries etc.

## *5. Distributed Storage*

Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

# HiveQL Data Definition & Manipulation Query Commands

**HiveQL Data Definition Commands :**

They are used to define and manage schemas, databases, tables, and other metadata in Hive. Few of these commands are listed and briefly described below

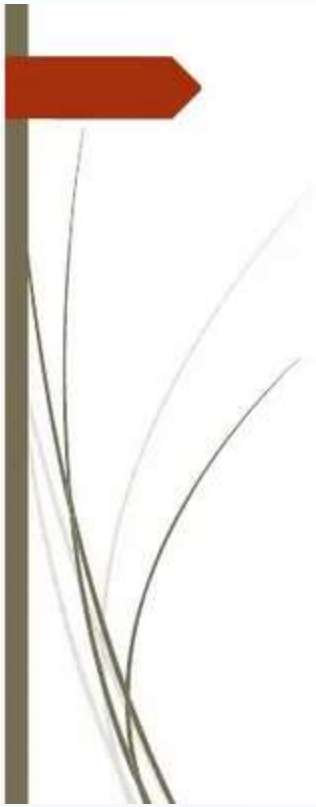
CREATE: Create databases, tables, or views.
DROP : Delete databases, tables, or views
ALTER : Modify table structures or properties
SHOW : List databases, tables, or partitions.
DESCRIBE: Display schema details for a table.

**HiveQL Data Manipulation Commands :**

These are used to manipulate the data stored in Hive tables, such as inserting, loading, querying, or modifying data.

LOAD DATA : Load data into a Hive table from HDFS or local file system.
INSERT : Insert data into a table or partition.
SELECT : Retrieve data from Hive tables.
UPDATE : Update specific rows in a transactional table.
DELETE : Delete rows from a transactional table.
EXPORT/IMPORT : Move data and metadata between Hive instances.

This list highlights the main HiveQL commands for defining schemas (DDL) and manipulating data (DML) in Hive.



# Thanks