HIVE

BY
PROF. PRAKASH PATEL
ASST.PROF.-IT DEPT.
GIT

What is HIVE?

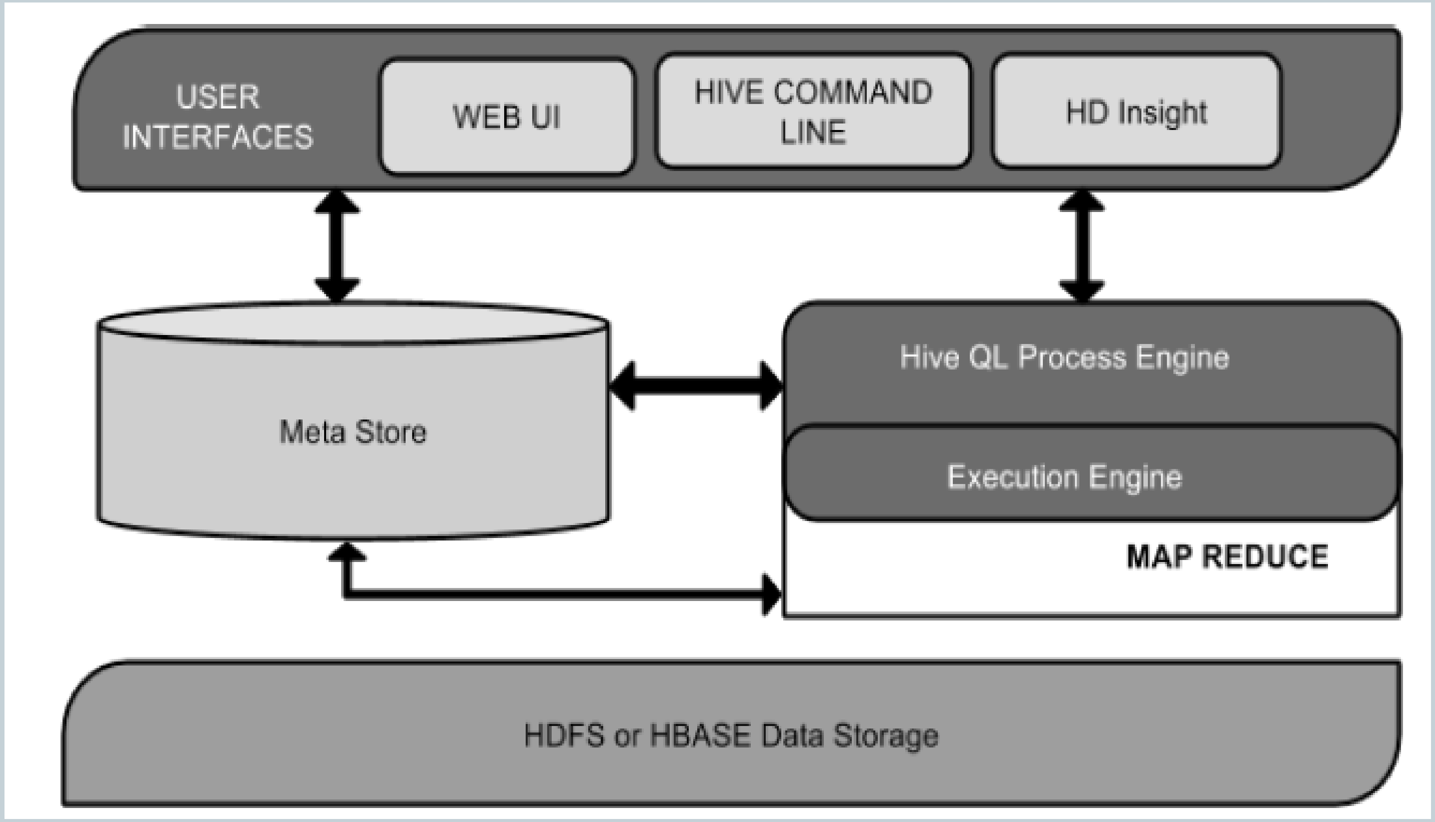
- Hive is a data warehouse infrastructure tool to process structured data in Hadoop.
- It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.
- Initially It was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source.
- It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

Features of HIVE

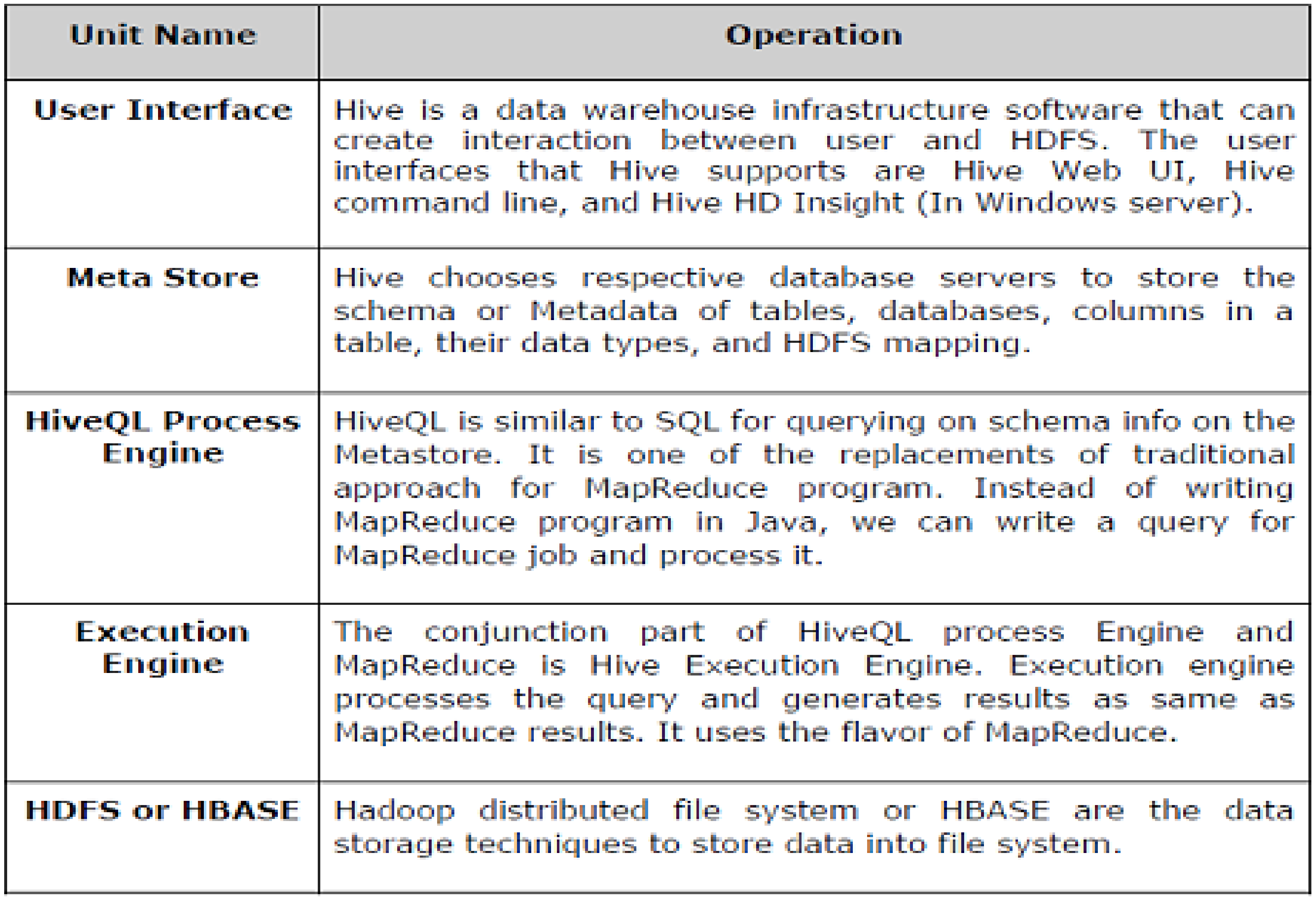
- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

Architecture of Hive

- The following component diagram depicts the architecture of Hive:



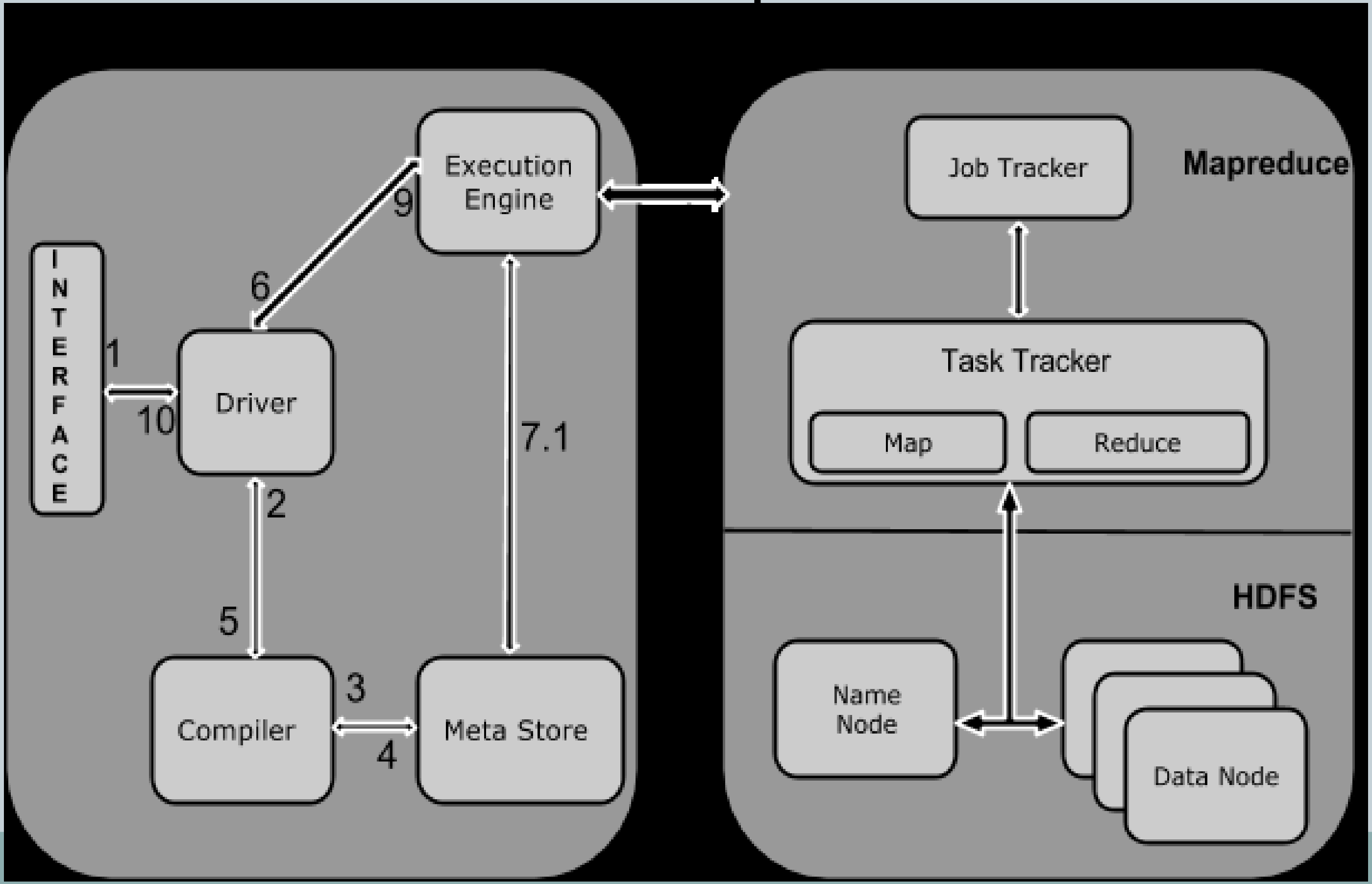
Components of HIVE



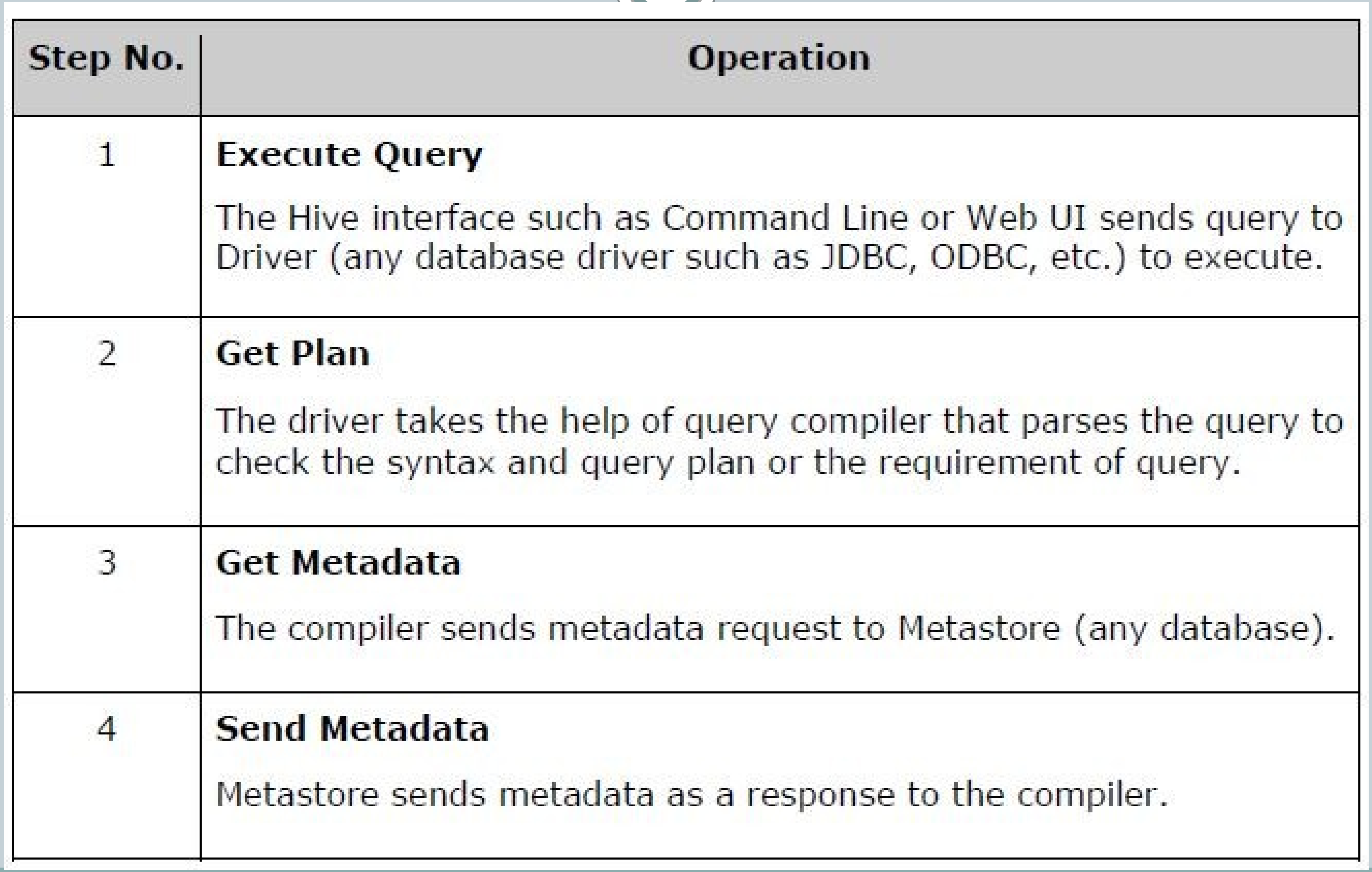
Unit Name	Operation
User Interface	Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).
Meta Store	Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.
HiveQL Process Engine	HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.
Execution Engine	The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.
HDFS or HBASE	Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

Working of Hive

- The following diagram depicts the workflow between Hive and Hadoop.

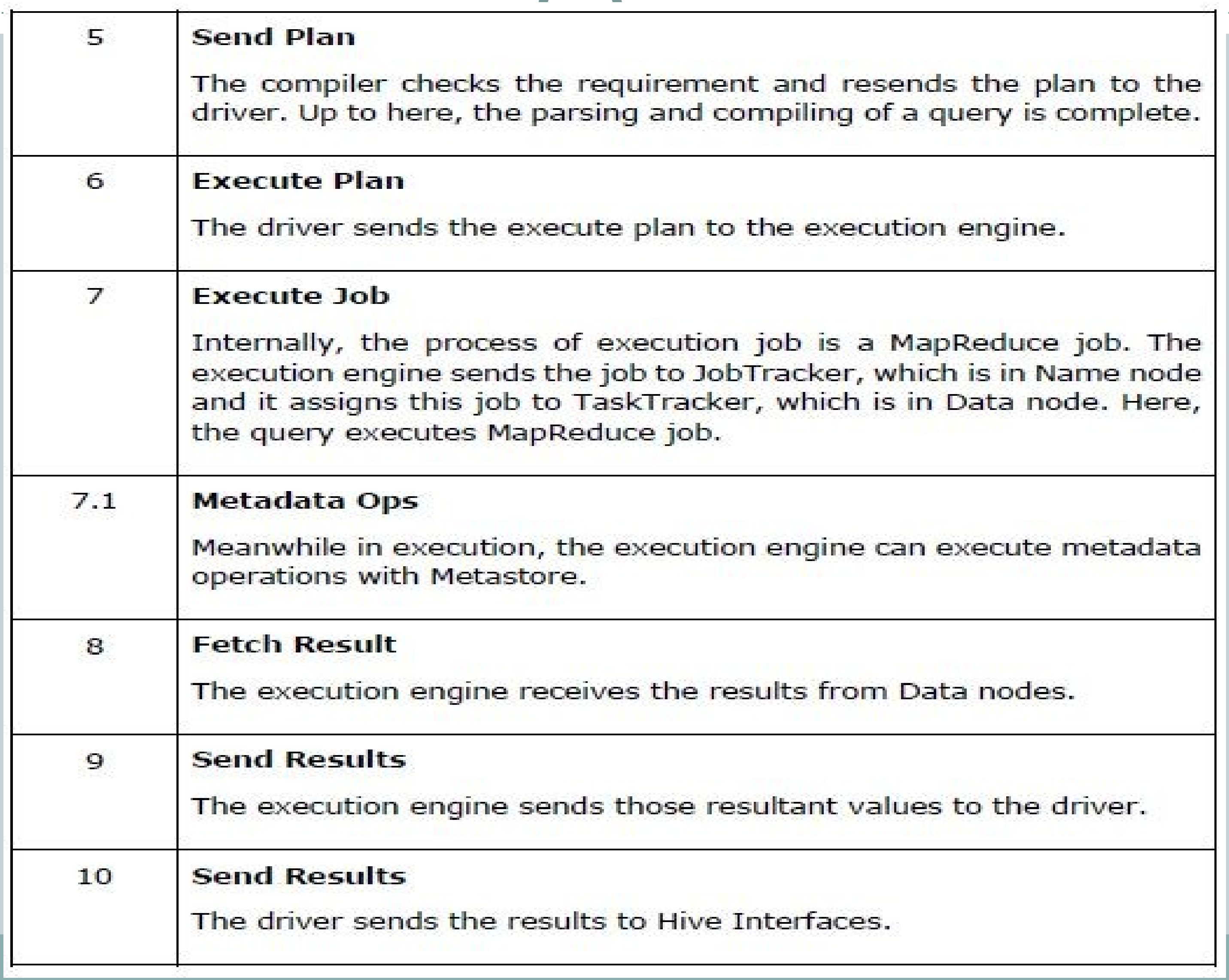


The following table defines how Hive interacts with Hadoop framework:



Step No.	Operation
1	**Execute Query** The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute.
2	**Get Plan** The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query.
3	**Get Metadata** The compiler sends metadata request to Metastore (any database).
4	**Send Metadata** Metastore sends metadata as a response to the compiler.

continue



5	**Send Plan** The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete.
6	**Execute Plan** The driver sends the execute plan to the execution engine.
7	**Execute Job** Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job.
7.1	**Metadata Ops** Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
8	**Fetch Result** The execution engine receives the results from Data nodes.
9	**Send Results** The execution engine sends those resultant values to the driver.
10	**Send Results** The driver sends the results to Hive Interfaces.

HIVE Data Types

- All the data types in Hive are classified into four types, given as follows:
- 1. Column Types
- 2. Literals
- 3. Null Values
- 4. Complex Types

Column Types

- **Integral Types**
  - Integer type data can be specified using integral data types, INT. When the data range exceeds the range of INT, you need to use BIGINT and if the data range is smaller than the INT, you use SMALLINT. TINYINT is smaller than SMALLINT.
- **String Types**
  - String type data types can be specified using single quotes (' ') or double quotes (" "). It contains two data types: VARCHAR and CHAR. Hive follows C-types escape characters.

Data types

- **Timestamp**
  - It supports traditional UNIX timestamp with optional nanosecond precision. It supports java.sql.Timestamp format “YYYY-MM-DD HH:MM:SS.fffffffff” and format “yyyy-mm-dd hh:mm:ss.ffffffffff”.
- **Dates**
  - DATE values are described in year/month/day format in the form {{YYYY-MM-DD}}.
- **Decimals**
  - The DECIMAL type in Hive is as same as Big Decimal format of Java. It is used for representing immutable arbitrary precision.
- **Union Types**
  - Union is a collection of heterogeneous data types. You can create an instance using **create union**.

Literals

- **Floating Point Types**
  - Floating point types are nothing but numbers with decimal points. Generally, this type of data is composed of DOUBLE data type.
- **Decimal Type**
  - Decimal type data is nothing but floating point value with higher range than DOUBLE data type. The range of decimal type is approximately -10^{-308} to 10^{308}.

Null Value

- Missing values are represented by the special value NULL.

Complex Types

- **Arrays**
- Arrays in Hive are used the same way they are used in Java.
- Syntax: ARRAY<data_type>
- **Maps**
- Maps in Hive are similar to Java Maps.
- Syntax: MAP<primitive_type, data_type>
- **Structs**
- Structs in Hive is similar to using complex data with comment.
- Syntax: STRUCT<col_name : data_type [COMMENT col_comment], ...>

Create Database

- **Create Database Statement:**
- Create Database is a statement used to create a database in Hive.
- A database in Hive is a **namespace** or a collection of tables.
- The **syntax** for this statement is as follows:
- CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>;

continue

- The following query is executed to create a database named **userdb**:
- hive> CREATE DATABASE [IF NOT EXISTS] userdb;

or

- hive> CREATE SCHEMA userdb;
- The following query is used to verify a databases list:
- hive> SHOW DATABASES;
- default
- userdb

Drop Database

- Drop Database is a statement that drops all the tables and deletes the database. Its syntax is as follows:
- DROP DATABASE Statement
  - DROP (DATABASE|SCHEMA) [IF EXISTS] database_name [RESTRICT|CASCADE];
- The following queries are used to drop a database. Let us assume that the database name is **userdb**.
- hive> DROP DATABASE IF EXISTS userdb;

- The following query drops the database using **CASCADE**. It means dropping respective tables before dropping the database.
- hive> DROP DATABASE IF EXISTS userdb CASCADE;
- The following query drops the database using **SCHEMA**.
- hive> DROP SCHEMA userdb;

Create Table Statement

- **Syntax**
- CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.] table_name
- [(col_name data_type [COMMENT col_comment], ...)]
- [COMMENT table_comment]
- [ROW FORMAT row_format]
- [STORED AS file_format]

- hive> CREATE TABLE IF NOT EXISTS employee (eid int, name String,
- > salary String, destination String)
- > COMMENT ‘Employee details’
- > ROW FORMAT DELIMITED
- > FIELDS TERMINATED BY ‘\t’
- > LINES TERMINATED BY ‘\n’
- > STORED AS TEXTFILE;

Load Data Statement

- There are two ways to load data: one is from local file system and second is from Hadoop file system.
- **Syntax**
- The syntax for load data is as follows:
- LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)]
- LOCAL is identifier to specify the local path. It is optional.
- OVERWRITE is optional to overwrite the data in the table.
- PARTITION is optional.

- The following query loads the given text into the table from sample.txt file.
- hive> LOAD DATA LOCAL INPATH '/home/user/sample.txt'
- > OVERWRITE INTO TABLE employee;

Alter Table Statement

- **Syntax :**

- ALTER TABLE name RENAME TO new_name
- ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])
- ALTER TABLE name DROP [COLUMN] column_name
- ALTER TABLE name CHANGE column_name new_name new_type
- ALTER TABLE name REPLACE COLUMNS (col_spec[, col_spec ...])

- The following query renames the table from **employee** to **emp**.
- hive> ALTER TABLE employee RENAME TO emp;

- The following query adds a column named **dept** to the **employee** table.
- hive> ALTER TABLE employee ADD COLUMNS (> dept STRING COMMENT 'Department name');

- The following query deletes all the columns from the **employee** table and replaces it with **emp** and **name** columns:
- hive> ALTER TABLE employee REPLACE COLUMNS (> eid INT empid Int, > ename STRING name String);

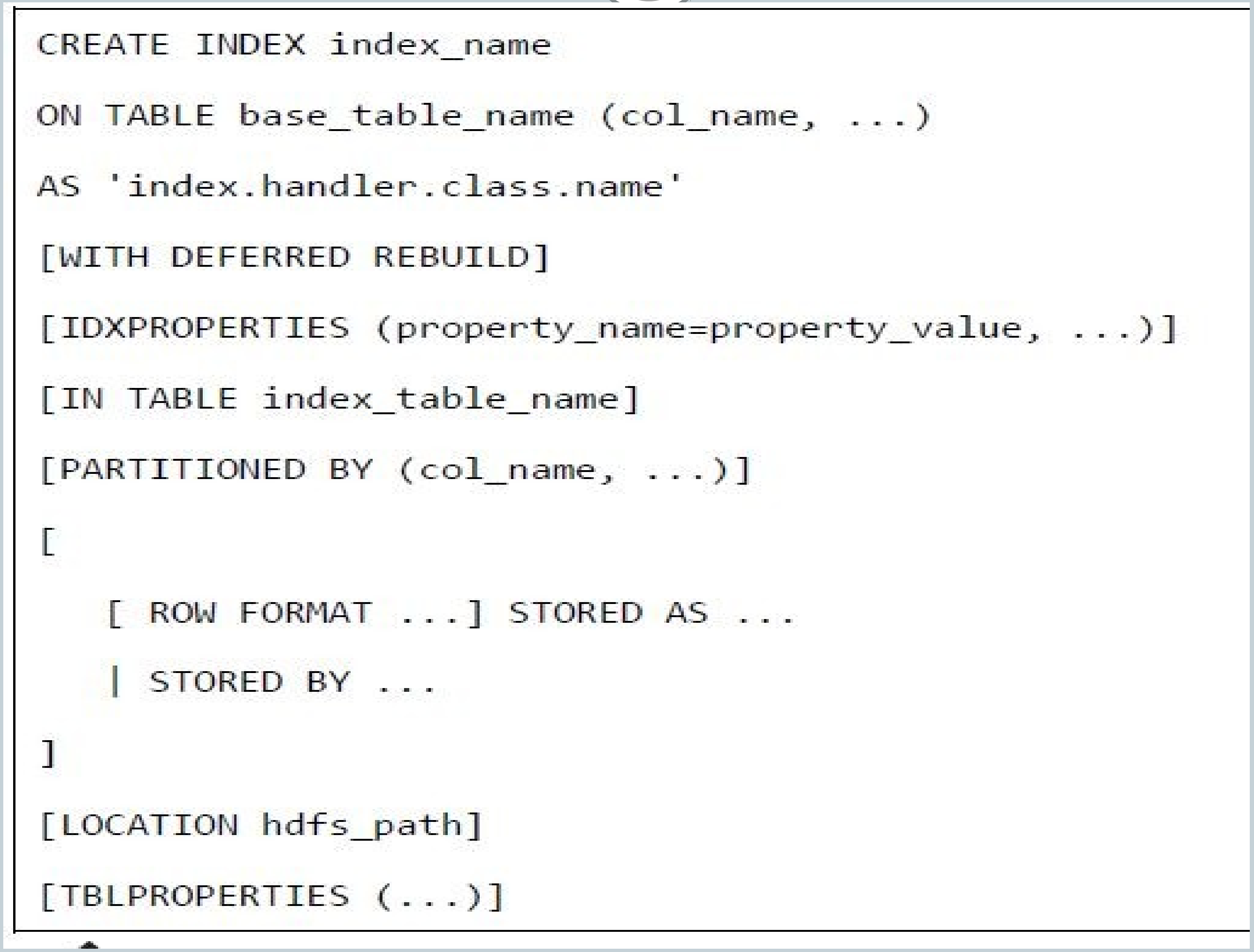
Drop Table Statement

- The syntax is as follows:
- DROP TABLE [IF EXISTS] table_name;
- The following query drops a table named **employee:**
- hive> DROP TABLE IF EXISTS employee;

- **Creating a View**
- You can create a view at the time of executing a SELECT statement. The syntax is as follows:
- CREATE VIEW [IF NOT EXISTS] view_name [(column_name [COMMENT column_comment], ...)]
- [COMMENT table_comment]
- AS SELECT ...

- The following query retrieves the employee details using the above scenario:
- hive> CREATE VIEW emp_30000 AS
- > SELECT * FROM employee
- > WHERE salary>30000;

Creating an Index

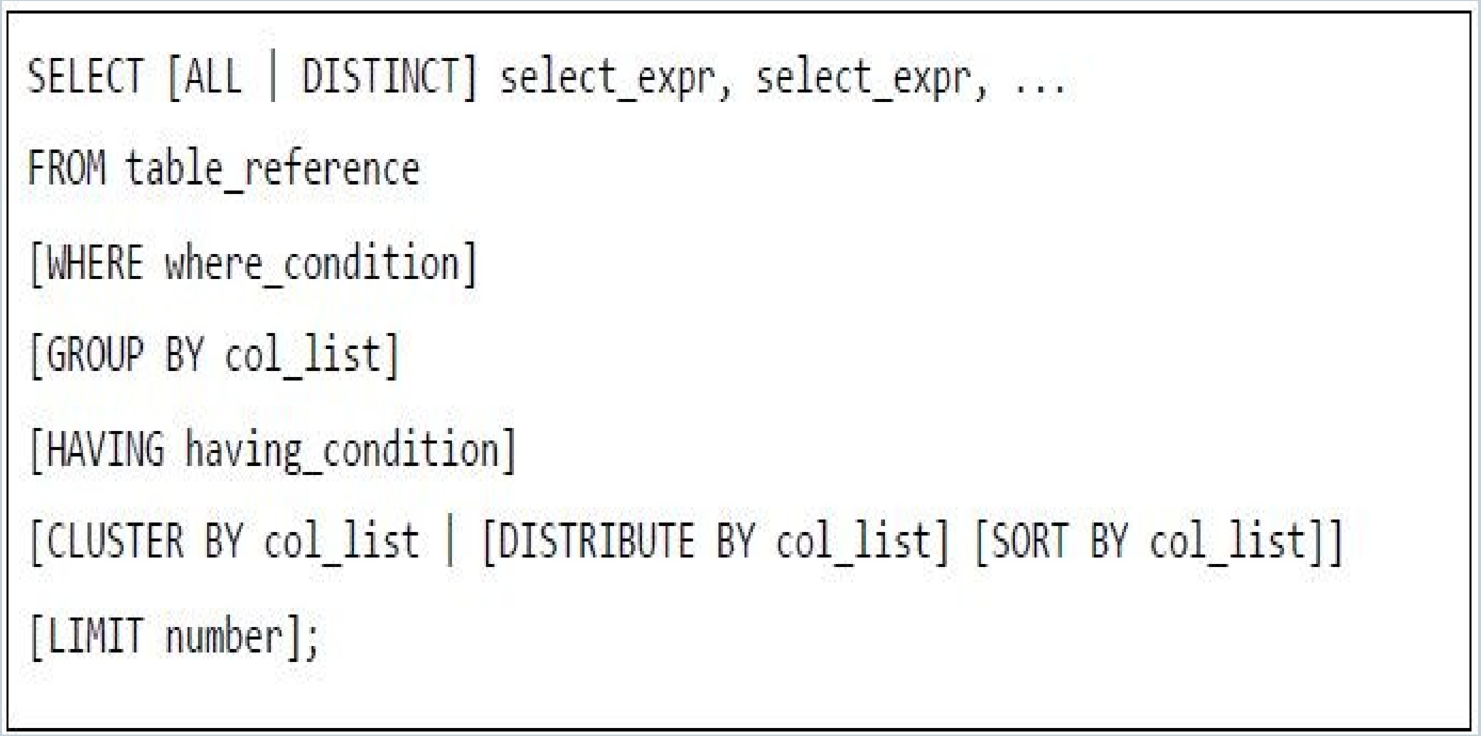


```
CREATE INDEX index_name

ON TABLE base_table_name (col_name, ...)

AS 'index.handler.class.name'

[WITH DEFERRED REBUILD]

[IDXPROPERTIES (property_name=property_value, ...)]

[IN TABLE index_table_name]

[PARTITIONED BY (col_name, ...)]

[

   [ ROW FORMAT ...] STORED AS ...

   | STORED BY ...

]

[LOCATION hdfs_path]

[TBLPROPERTIES (...)]
```

- hive> CREATE INDEX inedx_salary ON TABLE employee(salary)
- > AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler';

HIVEQL SELECT…WHERE

- The Hive Query Language (HiveQL) is a query language for Hive to process and analyze structured data in a Metastore.



```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[HAVING having_condition]
[CLUSTER BY col_list | [DISTRIBUTE BY col_list] [SORT BY col_list]]
[LIMIT number];
```

- hive> SELECT * FROM employee WHERE salary>30000;

HIVEQL JOINS

- JOINS is a clause that is used for combining specific fields from two tables by using values common to each one.
- It is used to combine records from two or more tables in the database.
- It is more or less similar to SQL JOINS.

- There are different types of joins given as follows:
- JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

Syntax

- The following query executes JOIN on the CUSTOMER and ORDER tables, and retrieves the records:

- hive> SELECT c.ID, c.NAME, c.AGE, o.AMOUNT > FROM CUSTOMERS c JOIN ORDERS o > ON (c.ID = o.CUSTOMER_ID);

Outer Join

- Left Outer
- hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE > FROM CUSTOMERS c > LEFT OUTER JOIN ORDERS o > ON (c.ID = o.CUSTOMER_ID);
- Right Outer
- hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE > FROM CUSTOMERS c > RIGHT OUTER JOIN ORDERS o > ON (c.ID = o.CUSTOMER_ID);

- Full outer:
- hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE > FROM CUSTOMERS c > FULL OUTER JOIN ORDERS o > ON (c.ID = o.CUSTOMER_ID);