UNIT-V (Applying Structure to Hadoop Data with Hive)

Introduction to Hive: **Apache Hive** is a data warehouse software project built on top of Apache Hadoop for providing data summarization, query and analysis. Hive gives an SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop.

Most of the Data Scientists use SQL queries in order to explore the data and get valuable insights from them. Now, as the volume of data is growing at such a high pace, we need new dedicated tools to deal with big volumes of data.

Initially, Hadoop came up and became one of the most popular tools to process and store big data. But developers were required to write complex map-reduce codes to work with Hadoop. This is Facebook's Apache Hive came to rescue. It is another tool designed to work with Hadoop. We can write SQL like queries in the hive and in the backend it converts them into the map-reduce jobs.

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.

Introduction

Most of the Data Scientists use SQL queries in order to explore the data and get valuable insights from them. Now, as the volume of data is growing at such a high pace, we need new dedicated tools to deal with big volumes of data.

Initially, Hadoop came up and became one of the most popular tools to process and store big data. But developers were required to write complex map-reduce codes to work with Hadoop. This is Facebook's Apache Hive came to rescue. It is another tool designed to work with Hadoop. We can write SQL like queries in the hive and in the backend it converts them into the map-reduce jobs.

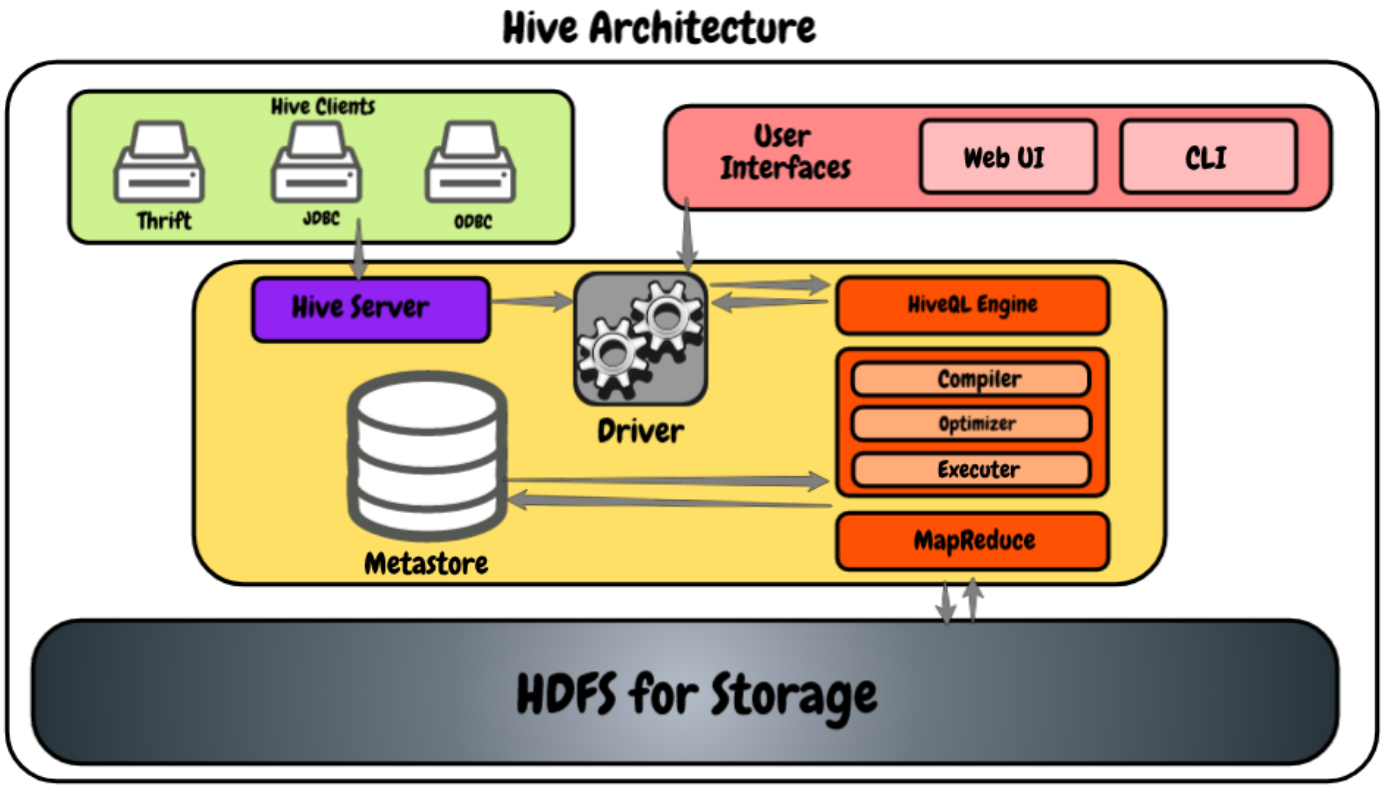
What is Apache Hive?

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.

It uses a scripting language called HiveQL which is almost similar to the SQL. So now, we just have to write SQL-like commands and at the backend of Hive will automatically convert them into the map-reduce jobs.

Apache Hive Architecture:

Let's have a look at the following diagram, which shows the architecture.

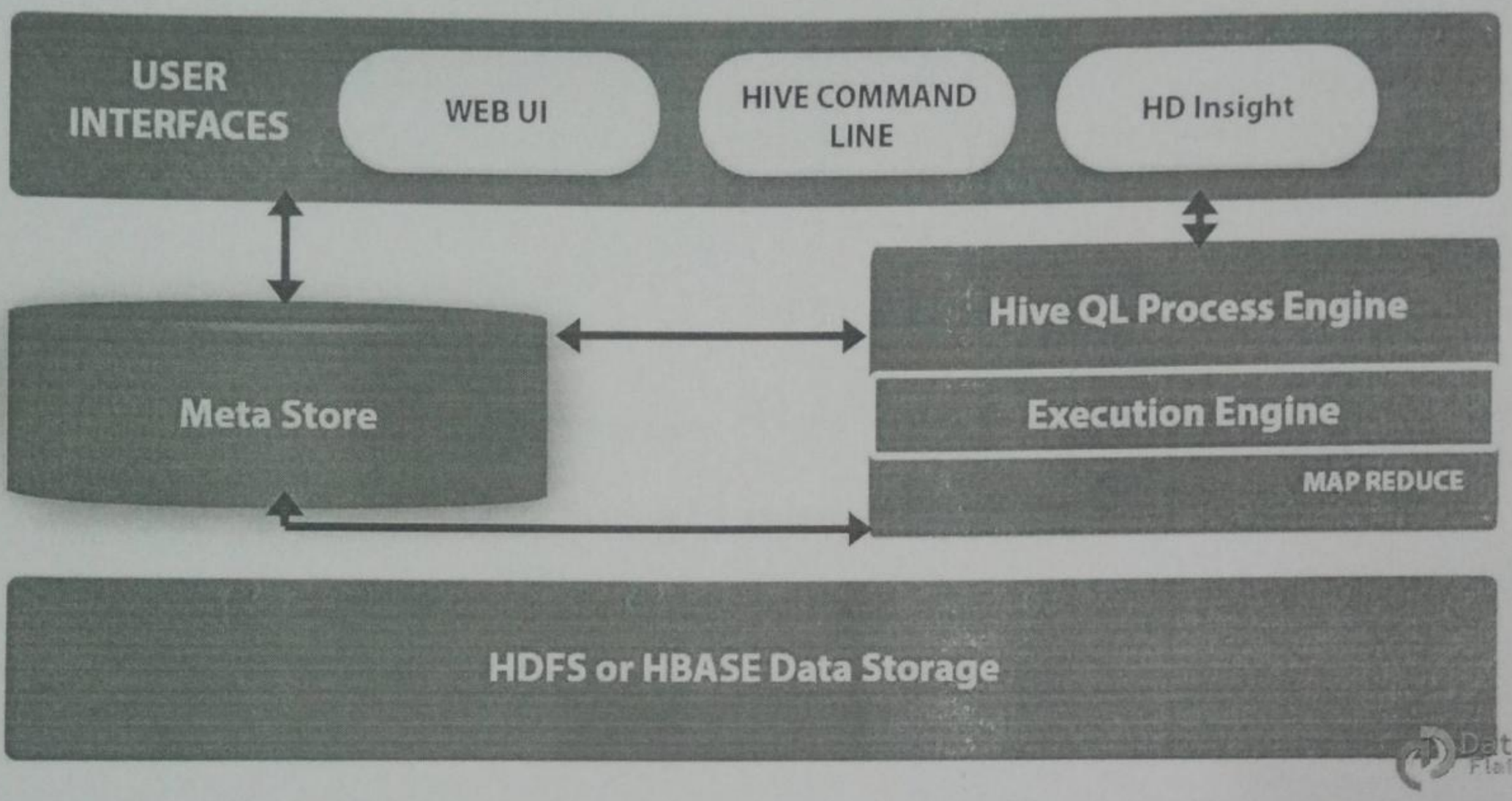


- **Hive Clients:** It allows us to write hive applications using different types of clients such as thrift server, JDBC driver for Java, and Hive applications and also supports the applications that use ODBC protocol.
- **Hive Services:** As a developer, if we wish to process any data, we need to use 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 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.
- **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.

Swami Keshvanand Institute of Technology, Management &Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017, INDIA
Approved by AICTE, Ministry of HRD, Government of India
Recognized by UGC under Section 2(f) of the UGC Act, 1956
Tel. : +91-0141- 5160400Fax: +91-0141-2759555
E-mail: info@skit.ac.in Web: www.skit.ac.in

Hive Architecture: The following diagram shows the hive architecture introduce by HIVE.



There are several different units in this component diagram. Now, let's describes each unit:

a. User Interface

As we know it is a data warehouse infrastructure software. It can create interaction between user and HDFS. Moreover, there are various user interfaces that Hive supports. They are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).

b. Meta Store

Basically, to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping, it chooses respective database servers.

c. HiveQL Process Engine

Also, we can say HiveQL is same as SQL Especially, for querying on schema info on the Metastore. In addition, for MapReduce program, it is one of the replacements of the traditional approach. Moreover, we can write a query for MapReduce job and process it, instead of writing MapReduce program in Java.

d. Execution Engine

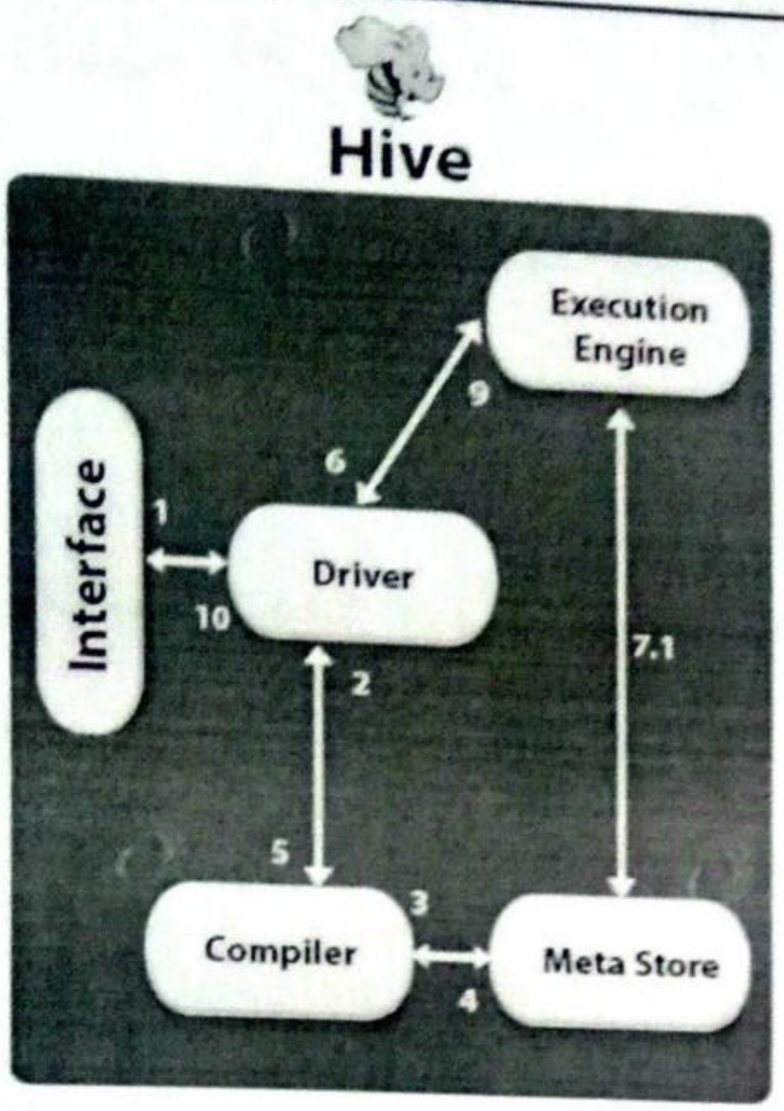
Although, Hive Execution Engine is the conjunction part of HiveQL process Engine and MapReduce. Execution engine processes the query and generates results as same as MapReduce results. Also, it uses the flavour of MapReduce.

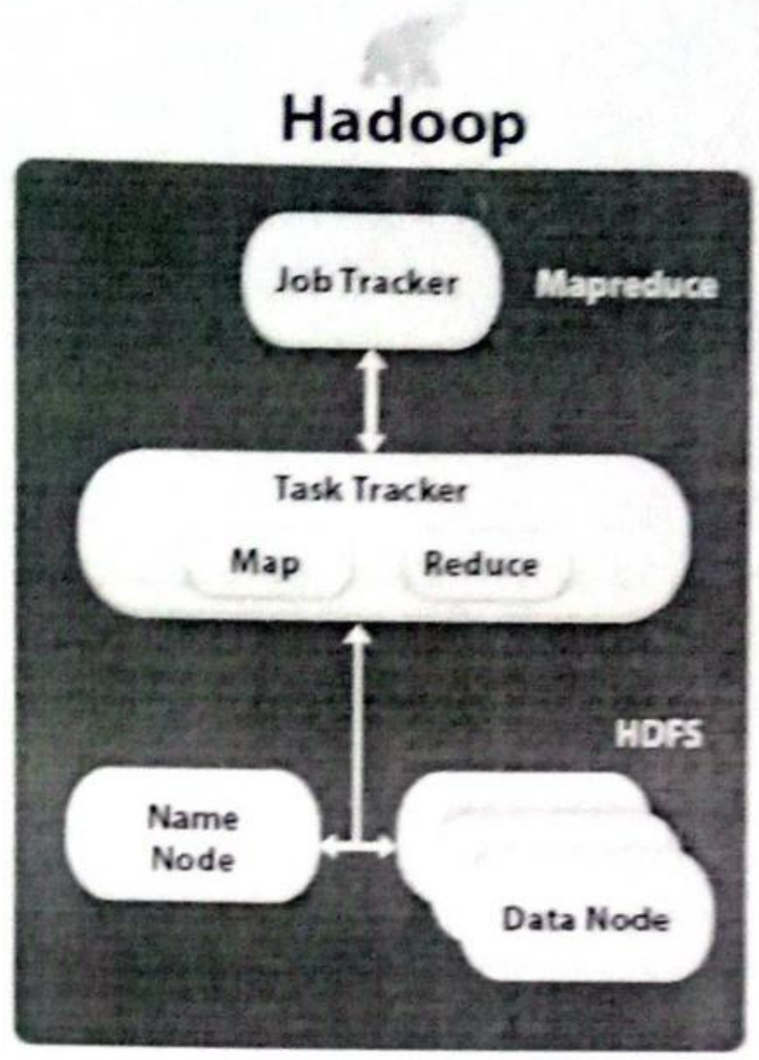
e. HDFS or HBase

Basically, to store data into file system Hadoop distributed file system

or HBase is the data storage techniques.

Swami Keshvanand Institute of Technology, Management &Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017, INDIA
Approved by AICTE, Ministry of HRD, Government of India
Recognized by UGC under Section 2(f) of the UGC Act, 1956
Tel. : +91-0141- 5160400Fax: +91-0141-2759555
E-mail: info@skit.ac.in Web: www.skit.ac.in





The following table defines how Hive interacts with Hadoop framework.

Step-1 Execute Query

At very first, the Hive interface (Command Line or Web UI) sends the query to Driver (any database driver such as JDBC, ODBC etc.) to execute.

Step-2 Get Plan

Afterwards, the driver takes the help of query compiler, which parses the query to check the syntax and query plan or the requirement of the query.

Step-3 Get Metadata

Further, the compiler sends metadata request to Metastore (any database)

Step-4 Send Metadata

After that Metastore sends metadata as a response to the compiler

Step-5 Send Plan

Then the compiler checks the requirement and resends the plan to the driver

However, the parsing and compiling of a query are complete, Up to here

Step-6 Execute Plan

Further, the driver sends the execution plan to the execution engine

Step-7 Execute Job

Then, the process of execution job is a MapReduce job, internally Also, 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. Moreover, the query executes MapReduce job, here.

- Metadata Ops

During the execution, the execution engine can execute metadata operations with Metastore.

Step-8 Fetch Result

While execution is over, the execution engine receives the results from Data nodes

Step-9 Send Results

After fetching results, execution engine sends those resultant values to the driver

Step-10 Send Results

At last, the driver sends the results to Hive interfaces

Features of Apache Hive

There are so many features of Apache Hive. Let's discuss them one by one-

- Hive provides data summarization, query, and analysis in much easier manner.
- Hive supports external tables, which make it possible to process data without actually storing in HDFS.
- Apache Hive fits the low-level interface requirement of Hadoop perfectly
- It also supports partitioning of data at the level of tables to improve performance.
- Hive has a rule-based optimizer for optimizing logical plans.
- It is scalable, familiar, and extensible.
- Using HiveQL does not require any knowledge of programming language, Knowledge of basic SQL query is enough.

- We can easily process structured data in Hadoop using Hive
- Querying in Hive is very simple as it is similar to SQL
- We can also run Ad-hoc queries for the data analysis using Hive.

Limitation of Apache Hive

Hive has the following limitations-

- Apache does not offer real-time queries and row level updates.
- Hive also provides acceptable latency for interactive data browsing
- It is not good for online transaction processing.
- Latency for Apache Hive queries is generally very high

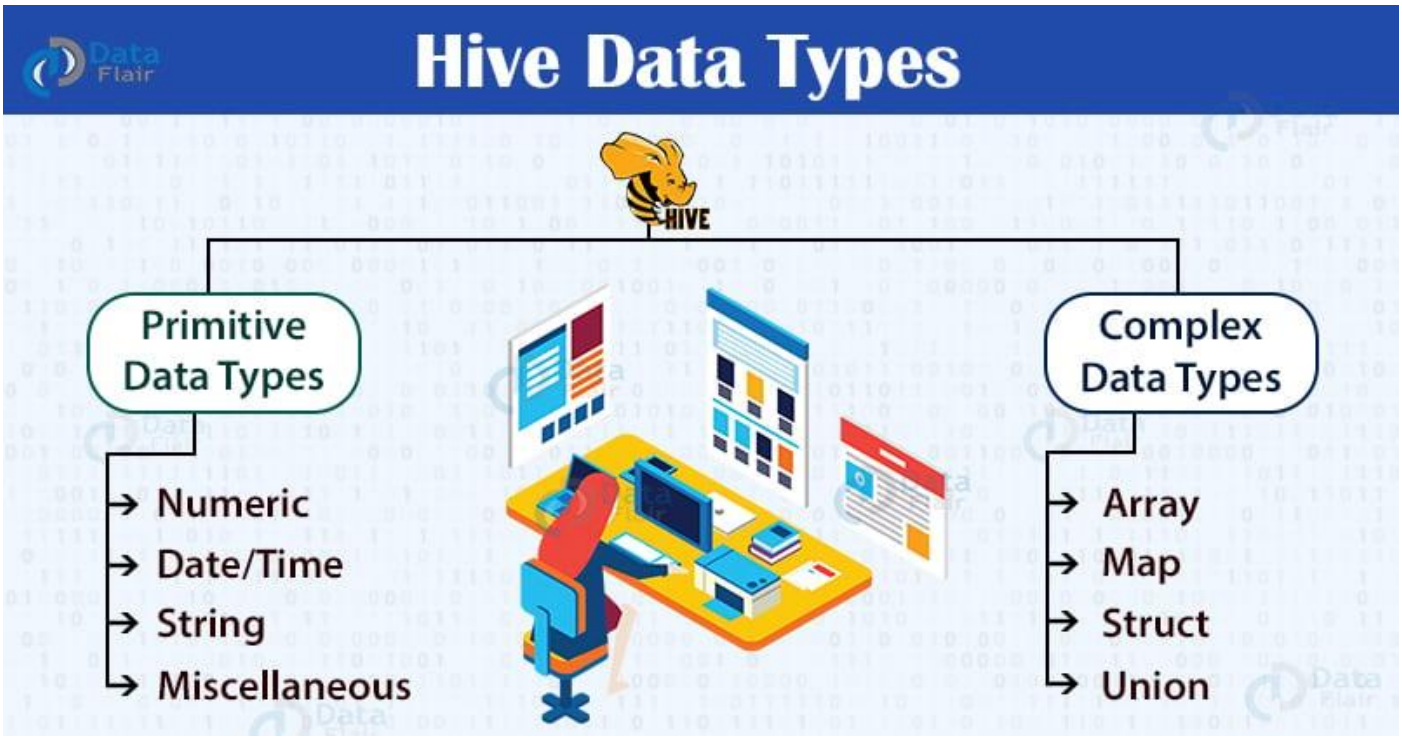
Hive Clients:

- Command Line
- JDBC
  - JDBC Client Sample Code
  - Running the JDBC Command Line
  - JDBC
  - Python
  - PHP
  - Thrift Java Client
  - ODBC
  - Thrift C++ Client

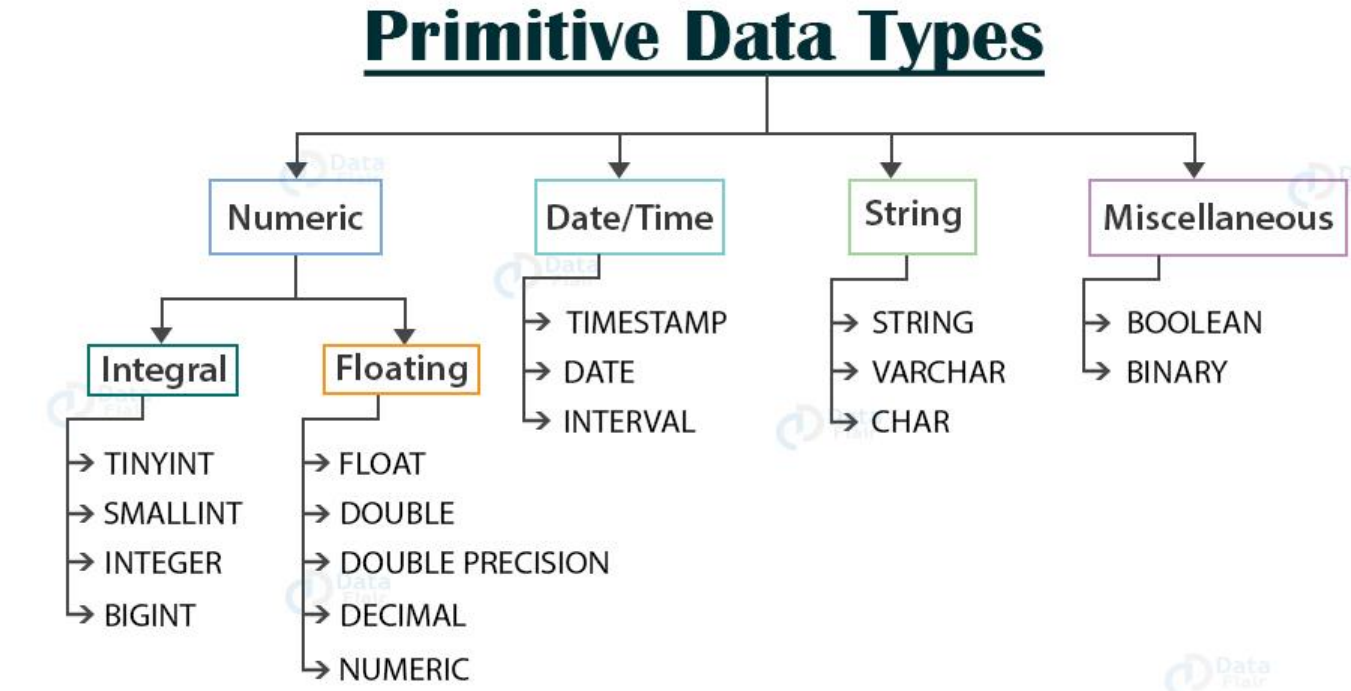
This page describes the different clients supported by Hive. The command line client currently only supports an embedded server. The JDBC and thrift-java clients support both embedded and standalone servers. Clients in other languages only support standalone server.

Swami Keshvanand Institute of Technology, Management &Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017, INDIA
Approved by AICTE, Ministry of HRD, Government of India
Recognized by UGC under Section 2(f) of the UGC Act, 1956
Tel. : +91-0141- 5160400Fax: +91-0141-2759555
E-mail: info@skit.ac.in Web: www.skit.ac.in

Working With Hive Data Type: Data Types in Hive specifies the column type in Hive tables, which describes the two categories of Hive Data types that are **primitive data type** and **complex data type.**



Primitive Data Types:



Integral data type

a. TINYINT – (1-byte signed integer ranging from -128 to 127)

b. SMALLINT – (2-byte signed integer ranging from -32, 768 to 32, 767)

c. INTEGER – (4-byte signed integer ranging from -2, 147, 483, 648 to 2, 147, 483, 647)

d. BIGINT – (8-byte signed integer ranging from -9, 223, 372, 036, 854, 775, 808 to 9, 223, 372, 036, 854, 775, 807)

1.2 Floating data type

a. FLOAT

It is a 4-byte single-precision floating-point number.

b. DOUBLE

It is an 8-byte double-precision floating-point number.

c. DOUBLE PRECISION

It is an alias for DOUBLE. It is only available starting with Hive 2.2.0

d. DECIMAL

It was introduced in Hive 0.11.0. It is based on Java's BigDecimal. DECIMAL types support both scientific and non-scientific notations.

e. NUMERIC

It started with Hive 3.0.0. The NUMERIC data type is the same as the DECIMAL type.

[ps2id id='Date-time' target="/]2. Date/Time data type:

a. TIMESTAMP

Timestamps were introduced in Hive 0.8.0. It supports traditional UNIX timestamp with the optional nanosecond precision.

The supported Timestamps format is yyyy-mm-dd hh:mm:ss[.f...] in the text files.

a. STRING

In Hive, String literals are represented either with the single quotes(' ') or with double-quotes(" ").

Hive uses C-style escaping.

b. VARCHAR

In Hive, VARCHAR data types are of different lengths, but we have to specify the maximum number of characters allowed in the character string.

If the string value assigned to the varchar is less than the maximum length, then the remaining space will be freed out.

Also, if the string value assigned is more than the maximum length, then the string is silently truncated.

The length of the varchar is between(1 to 65535).

c. CHAR

CHAR data types are fixed-length.

The values shorter than the specified length are padded with the spaces.

Unlike VARCHAR, trailing spaces are not significant in CHAR types during comparisons.

The maximum length of CHAR is fixed at 255.

[ps2id id='Miscellaneous' target="/]4. Miscellaneous data type

a. BOOLEAN

Boolean types in Hive store either true or false.

b. BINARY

BINARY type in Hive is an array of bytes.

This is all about Hive Primitive Data Types. Let us now study Hive Complex Data Types.

Complex Data Types:

1. Array:

   Example: array('Data','Flair'). The second element is accessed as array[1].

2. Map: **Example:** 'first' -> 'John', 'last' -> 'Deo', represented as map('first', 'John', 'last', 'Deo'). Now 'John' can be accessed with map['first'].

Struct: STRUCT <col_name : data_type [COMMENT col_comment], ...>

Example: For a column c3 of type STRUCT {c1 INTEGER; c2 INTEGER}, the c1 field is accessed by the expression c3.c1.

Swami Keshvanand Institute of Technology, Management &Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017, INDIA
Approved by AICTE, Ministry of HRD, Government of India
Recognized by UGC under Section 2(f) of the UGC Act, 1956
Tel. : +91-0141- 5160400Fax: +91-0141-2759555
E-mail: info@skit.ac.in Web: www.skit.ac.in

Creating and Managing Databases and Tables: The table in the hive is consists of multiple columns and records. The table we create in any database will be stored in the sub-directory of that database. The default location where the database is stored on HDFS is /user/hive/warehouse. The way of creating tables in the hive is very much similar to the way we create tables in SQL. We can perform the various operations with these tables like Joins, Filtering, etc.

Creating Table in Hive: Firstly create a database so that we can create tables inside it. The command for creating a database is shown below.

Syntax To Make Database:

CREATE DATABASE <database-name>;

Command:

```
CREATE DATABASE student_detail;

# this will create database student_detail

SHOW DATABASES;

# list down all the available databases

USE student_detail;
```

Syntax To Create Table in Hive:

```
CREATE TABLE [IF NOT EXISTS] <table-name> (
<column-name>    <data-type>,
<column-name>    <data-type> COMMENT 'Your Comment',
<column-name>    <data-type>,
.
.
.
<column-name>    <data-type>
)
COMMENT 'Add if you want'
LOCATION 'Location On HDFS'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

Note:

1. We can add a comment to the table as well as to each individual column.

2. ROW FORMAT DELIMITED shows that whenever a new line is encountered the new record entry will start.

3. FIELDS TERMINATED BY ',' shows that we are using ',' delimiter to separate each column.

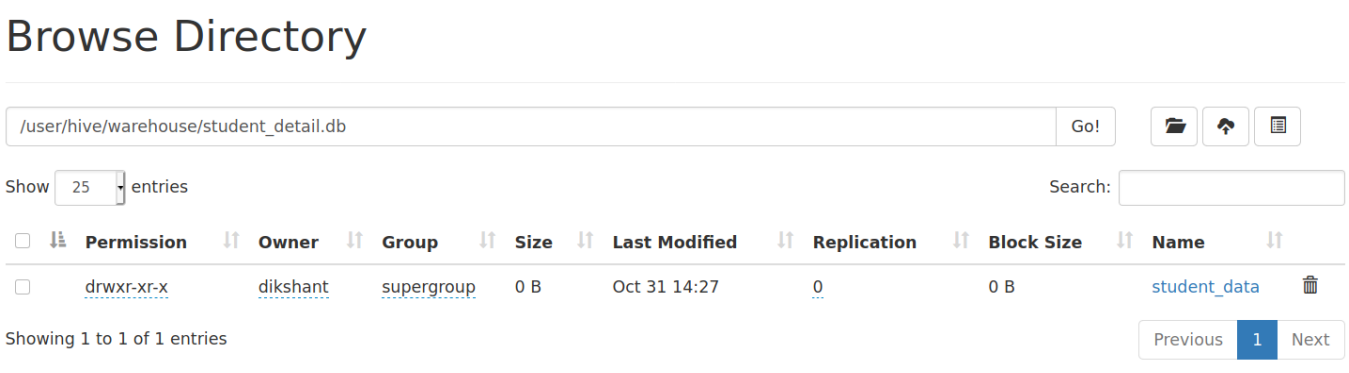
4. We can also override the default database location with the LOCATION option.

So let's create the table student_data in our student_detail database with the help of the command shown below.

```
CREATE TABLE IF NOT EXISTS student_data(
Student_Name STRING COMMENT 'This col. Store the name of student',
Student_Rollno INT COMMENT 'This col. Stores the rollno of student',
Student_Marks FLOAT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

SHOW TABLES IN student_detail;

Finally, let's check the location on HDFS where our student_detail database and student_data table is made. Move to localhost:50070/ for Hadoop 2 and to localhost:9870/ for Hadoop 3. Then Utilities -> Browse the file system and go to /user/hive/warehouse which is a default location where hive databases are created.



Swami Keshvanand Institute of Technology, Management &Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017, INDIA
Approved by AICTE, Ministry of HRD, Government of India
Recognized by UGC under Section 2(f) of the UGC Act, 1956
Tel. : +91-0141- 5160400Fax: +91-0141-2759555
E-mail: info@skit.ac.in Web: www.skit.ac.in

Types of Tables in Hive: There are two types of Table supported by the Hive.

Managed Tables

In a managed table, both the table data and the table schema are managed by Hive. The data will be located in a folder named after the table within the Hive data warehouse, which is essentially just a file location in HDFS.

The location is user-configurable when Hive is installed. By managed or controlled we mean that if you drop (delete) a managed table, then Hive will delete both the Schema (the description of the table) and the data files associated with the table. Default location is **/user/hive/warehouse).**

Creation of Managed Table:

```
CREATE TABLE IF NOT EXISTS stocks (exchange STRING,
symbol STRING,
```

```
price_open FLOAT,
price_high FLOAT,
price_low FLOAT,
price_adj_close FLOAT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;
```

External Tables:

An external table is one where only the table schema is controlled by Hive. In most cases, the user will set up the folder location within HDFS and copy the data file(s) there. This location is included as part of the table definition statement.

When an external table is deleted, Hive will only delete the schema associated with the table. The data files are not affected.

Syntax:

```
CREATE EXTERNAL TABLE IF NOT EXISTS stocks (exchange STRING,
symbol STRING,
price_open FLOAT,
price_high FLOAT,
price_low FLOAT,
price_adj_close FLOAT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/data/stocks';
```

Managed vs. External Table:

Managed Table	External Table
Hive assumes that it owns the data for managed tables.	For external tables, Hive assumes that it does not manage the data.
If a managed table or partition is dropped, the data and metadata associated with that table or partition are deleted.	Dropping the table does not delete the data, although the metadata for the table will be deleted.
For **Managed tables**, **Hive** stores data into its warehouse directory	For **External Tables**, **Hive** stores the data in the LOCATION specified during creation of

	the **table**(generally not in warehouse directory)
Managed table provides ACID/transnational action support.	External Table does not provide ACID/transactional action support.
Statements: ARCHIVE, UNARCHIVE, TRUNCATE, MERGE, CONCATENATE supported	Not supported.
Query Result Caching supported(saves the results of an executed Hive query for reuse).	Not Supported

Swami Keshvanand Institute of Technology, Management &Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017, INDIA
Approved by AICTE, Ministry of HRD, Government of India
Recognized by UGC under Section 2(f) of the UGC Act, 1956
Tel. : +91-0141- 5160400Fax: +91-0141-2759555
E-mail: info@skit.ac.in Web: www.skit.ac.in

DML Commands in Hive: Hive Data Manipulation Language commands are used for inserting, retrieving, modifying, deleting, and updating data in the Hive table.

There are many Hive DML commands like LOAD, INSERT, UPDATE, etc.

1. LOAD Command

The LOAD statement in Hive is used to move data files into the locations corresponding to Hive tables.

- If a LOCAL keyword is specified, then the LOAD command will look for the file path in the local filesystem.
- If the LOCAL keyword is not specified, then the Hive will need the absolute URI of the file.

- In case the keyword OVERWRITE is specified, then the contents of the target table/partition will be deleted and replaced by the files referred by filepath.
- If the OVERWRITE keyword is not specified, then the files referred by filepath will be appended to the table.

```
LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename
[PARTITION (partcol1=val1, partcol2=val2 ...)];
```

Example:

```
dataflair@admin1-All-Series: ~/apache-hive-3.1.2-bin
File  Edit  View  Search  Terminal  Help
0: jdbc:hive2://localhost:10000> LOAD DATA LOCAL INPATH '/home/dataflair/dab' INTO TABLE emp_data;
INFO  : Compiling command(queryId=dataflair_20200207151436_f132ee00-9293-44d3-ac40-e5929be8ed87): LOAD DATA
 LOCAL INPATH '/home/dataflair/dab' INTO TABLE emp_data
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO  : Completed compiling command(queryId=dataflair_20200207151436_f132ee00-9293-44d3-ac40-e5929be8ed87);
 Time taken: 0.02 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=dataflair_20200207151436_f132ee00-9293-44d3-ac40-e5929be8ed87): LOAD DATA
 LOCAL INPATH '/home/dataflair/dab' INTO TABLE emp_data
INFO  : Starting task [Stage-0:MOVE] in serial mode
INFO  : Loading data to table default.emp_data from file:/home/dataflair/dab
INFO  : Starting task [Stage-1:STATS] in serial mode
INFO  : Completed executing command(queryId=dataflair_20200207151436_f132ee00-9293-44d3-ac40-e5929be8ed87);
 Time taken: 0.153 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
No rows affected (0.181 seconds)
0: jdbc:hive2://localhost:10000>
```

Insert into:

INSERT INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select_statement1 FROM from statement;

```
dataflair@admin1-All-Series: ~/apache-hive-3.1.2-bin
File  Edit  View  Search  Terminal  Help
0: jdbc:hive2://localhost:10000> CREATE TABLE IF NOT EXISTS example(id STRING, name S
TRING, dep STRING, state STRING, salary STRING, year STRING);
INFO  : Compiling command(queryId=dataflair_20200224135418_bb6fcff6-48bc-4761-b1f8-a6
484a7a74e5): CREATE TABLE IF NOT EXISTS example(id STRING, name STRING, dep STRING, s
tate STRING, salary STRING, year STRING)
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO  : Completed compiling command(queryId=dataflair_20200224135418_bb6fcff6-48bc-47
61-b1f8-a6484a7a74e5); Time taken: 0.016 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=dataflair_20200224135418_bb6fcff6-48bc-4761-b1f8-a6
484a7a74e5): CREATE TABLE IF NOT EXISTS example(id STRING, name STRING, dep STRING, s
tate STRING, salary STRING, year STRING)
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=dataflair_20200224135418_bb6fcff6-48bc-47
61-b1f8-a6484a7a74e5); Time taken: 0.069 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
No rows affected (0.094 seconds)
0: jdbc:hive2://localhost:10000>
```

INSERT statement to load data into table "example".

```
dataflair@admin1-All-Series: ~/apache-hive-3.1.2-bin
File  Edit  View  Search  Terminal  Help
0: jdbc:hive2://localhost:10000> INSERT INTO TABLE example SELECT emp.emp_id,emp.emp_name,emp.emp_dep,emp.s
tate,emp.salary,emp.year_of_joining FROM emp_data emp;
```

Insert Overwrite:

INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, ..) [IF NOT EXISTS]] select_statement FROM from_statement;

```
dataflair@admin1-All-Series: ~/apache-hive-3.1.2-bin
File  Edit  View  Search  Terminal  Help
0: jdbc:hive2://localhost:10000> INSERT OVERWRITE TABLE example SELECT dmy.enroll,dmy.name,dmy.department,d
my.state,dmy.salary,dmy.year FROM dummy dmy;
```

By using the SELECT statement, we can verify whether the existing data of the table 'example' is overwritten by the data of table 'dummy' or not.

Insert Value Statement:

```
dataflair@admin1-All-Series: ~/apache-hive-3.1.2-bin
File  Edit  View  Search  Terminal  Help
0: jdbc:hive2://localhost:10000> INSERT INTO TABLE student VALUES (101,'Callen','IT','7.8'), (103,'Joseph',
'CS','8.2'), (105,'Alex','IT','7.9');
```

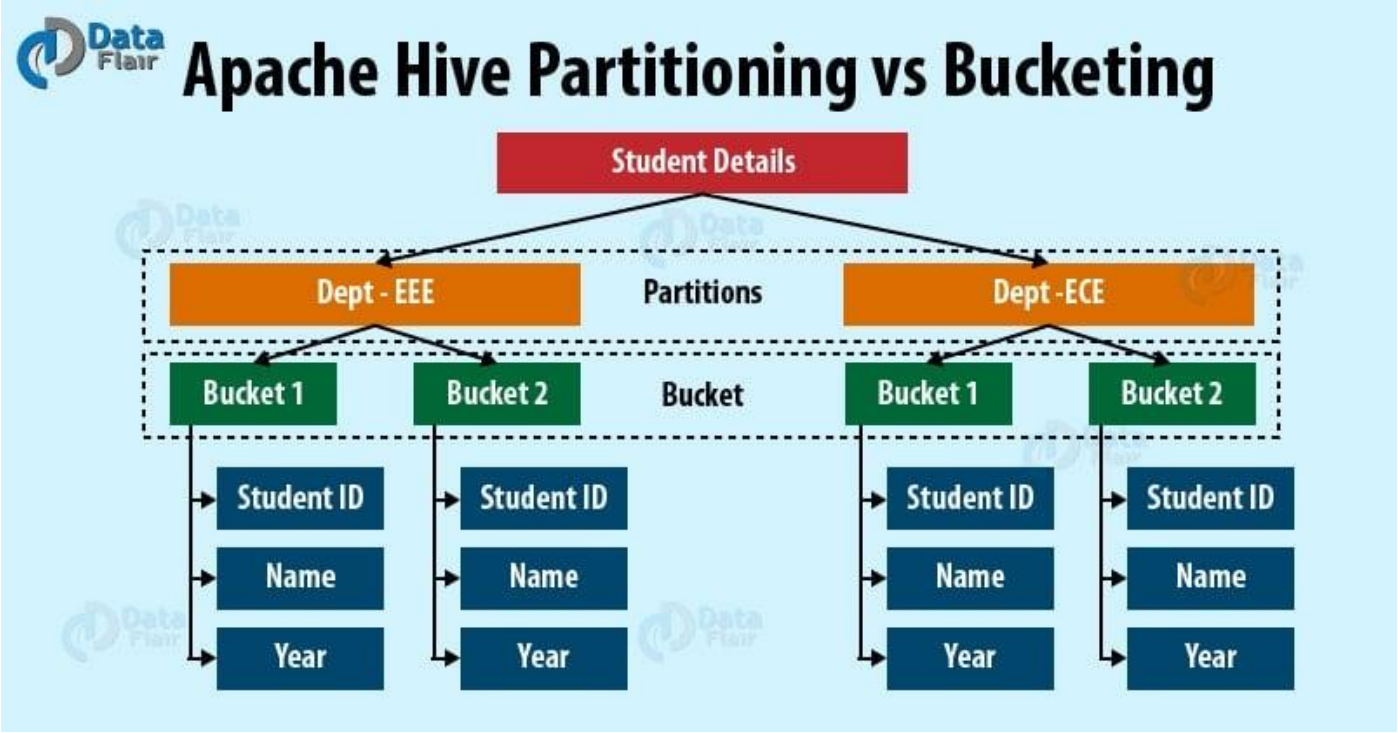
```
dataflair@admin1-All-Series: ~/apache-hive-3.1.2-bin
File  Edit  View  Search  Terminal  Help
0: jdbc:hive2://localhost:10000> SELECT * FROM student;
INFO  : Compiling command(queryId=dataflair_20200224171936_37d1120e-e1ce-4e85-aa8a-51ae9842c276): SELECT *
FROM student
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:student.roll_no, type:int, comment:nul
l), FieldSchema(name:student.name, type:string, comment:null), FieldSchema(name:student.branch, type:varcha
r(15), comment:null), FieldSchema(name:student.cgpa, type:varchar(5), comment:null)], properties:null)
INFO  : Completed compiling command(queryId=dataflair_20200224171936_37d1120e-e1ce-4e85-aa8a-51ae9842c276);
 Time taken: 0.085 seconds
INFO  : Executing command(queryId=dataflair_20200224171936_37d1120e-e1ce-4e85-aa8a-51ae9842c276): SELECT *
FROM student
INFO  : Completed executing command(queryId=dataflair_20200224171936_37d1120e-e1ce-4e85-aa8a-51ae9842c276);
 Time taken: 0.0 seconds
INFO  : OK
+------------------+----------------+-------------------+----------------+
| student.roll_no  | student.name   | student.branch    | student.cgpa   |
+------------------+----------------+-------------------+----------------+
| 101              | Callen         | IT                | 7.8            |
| 103              | Joseph         | CS                | 8.2            |
| 105              | Alex           | IT                | 7.9            |
+------------------+----------------+-------------------+----------------+
3 rows selected (0.137 seconds)
0: jdbc:hive2://localhost:10000>
```

4. DELETE command

The DELETE statement in Hive deletes the table data. If the WHERE clause is specified, then it deletes the rows that satisfy the condition in where clause

DELETE FROM tablename [WHERE expression];

Hive Partitioning vs Bucketing: Apache Hive is an open source data warehouse system used for querying and analyzing large datasets. Data in Apache Hive can be categorized into Table, Partition, and Bucket. The table in Hive is logically made up of the data being stored.



i. Partitioning and Bucketing Commands in Hive

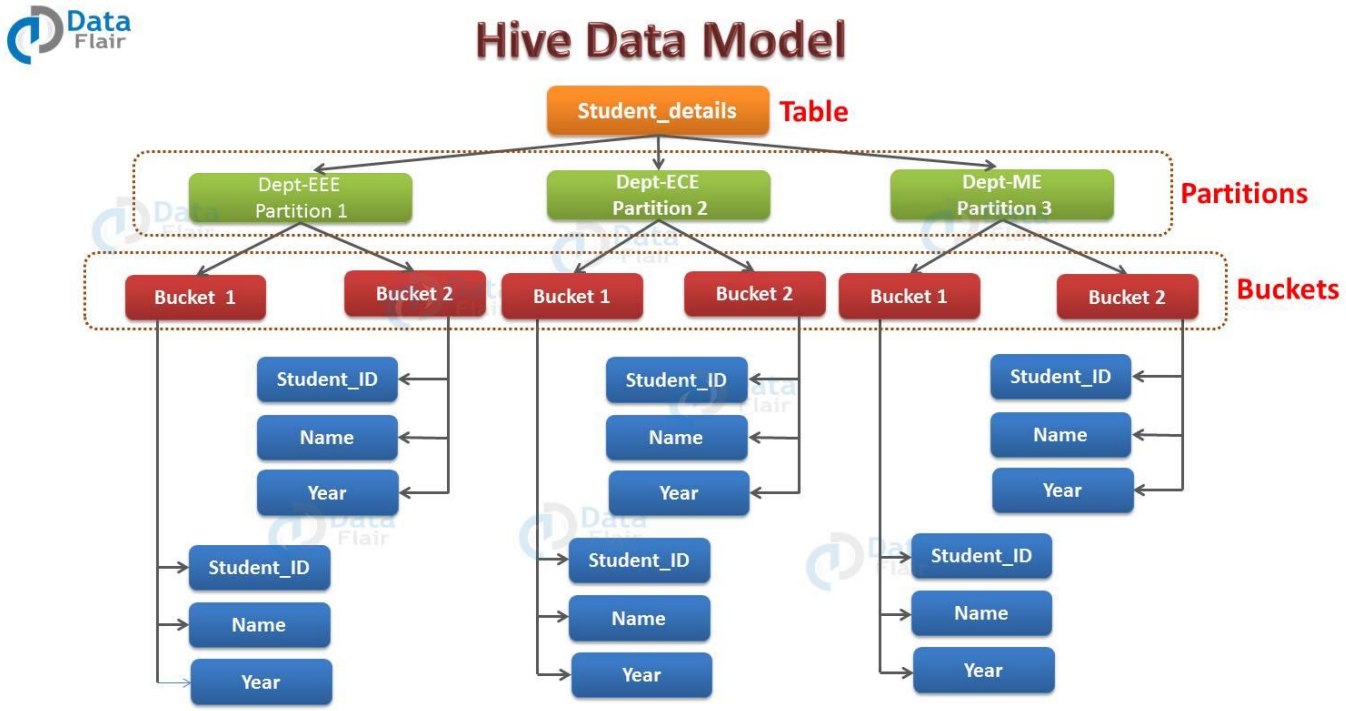
a) Partitioning

The Hive command for Partitioning is:

CREATE TABLE table_name (column1 data_type, column2 data_type) PARTITIONED BY (partition1 data_type, partition2 data_type,....);

b). Bucketing:

CREATE TABLE table_name PARTITIONED BY (partition1 data_type, partition2 data_type,....) CLUSTERED BY (column_name1, column_name2, ...) SORTED BY (column_name [ASC|DESC], ...)] INTO num_buckets BUCKETS;



a) Hive Partitioning Example

For example, we have a table employee_details containing the employee information of some company like **employee_id, name, department, year,** etc. Now, if we want to perform partitioning on the basis of department column.

Then the information of all the employees belonging to a particular department will be stored together in that very partition. Physically, a partition in Hive is nothing but just a sub-directory in the table directory.

For example, we have data for three departments in our employee_details table – Technical, Marketing and Sales. Thus we will have three partitions in total for each of the departments as we can see clearly in diagram below.

For each department we will have all the data regarding that very department residing in a separate sub – directory under the table directory.

So for example, all the employee data regarding Technical departments will be stored in user/hive/warehouse/employee_details/dept.=Technical. So, the

queries regarding Technical employee would only have to look through the data present in the Technical partition.

Therefore from above example, we can conclude that partitioning is very useful. It reduces the query latency by scanning only **relevant** partitioned data instead of the whole data set.

b) Hive Bucketing Example

Hence, from the above diagram, we can see that how each partition is bucketed into 2 buckets. Therefore each partition, says Technical, will have two files where each of them will be storing the Technical employee's data

- ### a) Pros and Cons of Hive Partitioning

  Pros:

- It distributes execution load horizontally.

- In partition faster execution of queries with the low volume of data takes place. For example, search population from Vatican City returns very fast instead of searching entire world population.

Cons:

- There is the possibility of too many small partition creations- too many directories.
- Partition is effective for low volume data. But there some queries like group by on high volume of data take a long time to execute. For example, grouping population of China will take a long time as compared to a grouping of the population in Vatican City.

b) Pros and Cons of Hive Bucketing

Pros:

- It provides faster query response like portioning.
- In bucketing due to equal volumes of data in each partition, joins at Map side will be quicker.

Cons:

- We can define a number of buckets during table creation. But loading of an equal volume of data has to be done manually by programmers.