**What is HIVE?**

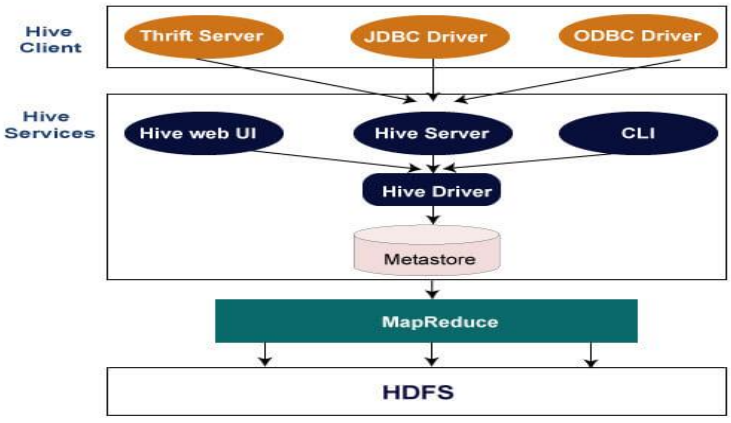
Hive is a data warehouse system which is used to analyze structured data. It is built on the top of Hadoop. It was developed by Facebook. Hive provides the functionality of reading, writing, and managing large datasets residing in distributed storage. It runs SQL like queries called HQL (Hive query language) which gets internally converted to MapReduce jobs.

**Hive Features**

- Hive is fast and scalable.
- It provides SQL-like queries (i.e., HQL) that are implicitly transformed to MapReduce or Spark jobs.
- It is capable of analyzing large datasets stored in HDFS.
- It allows different storage types such as plain text, RCFile, and HBase.
- It uses indexing to accelerate queries.
- It can operate on compressed data stored in the Hadoop ecosystem.
- It supports user-defined functions (UDFs) where user can provide its functionality.

**Limitations of Hive**

- Hive is not capable of handling real-time data.
- It is not designed for online transaction processing.
- Hive queries contain high latency.



**Prerequisite**

- Java Installation - Check whether the Java is installed or not using the following command.

$ java -version

- Hadoop Installation - Check whether the Hadoop is installed or not using the following command.

$hadoop version

- Download the Apache Hive tar file.

http://mirrors.estointernet.in/apache/hive/hive-1.2.2/

- Unzip the downloaded tar file.

tar -xvf apache-hive-1.2.2-bin.tar.gz

Open the bashrc file.

$ sudo nano ~/.bashrc

- Now, provide the following HIVE_HOME path.

**export HIVE_HOME=/home/codegyani/apache-hive-1.2.2-bin**

**export PATH=$PATH:/home/codegyani/apache-hive-1.2.2-bin/bin**

- Update the environment variable.

$ source ~/.bashrc

- Let's start the hive by providing the following command.

$ hive

**Integer Types**

| Type | Size | Range |
|---|---|---|
| TINYINT | 1-byte signed integer | -128 to 127 |
| SMALLINT | 2-byte signed integer | 32,768 to 32,767 |
| INT | 4-byte signed integer | 2,147,483,648 to 2,147,483,647 |
| BIGINT | 8-byte signed integer | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |

**Decimal Type**

| Type | Size | Range |
|---|---|---|
| FLOAT | 4-byte | Single precision floating point number |
| DOUBLE | 8-byte | Double precision floating point number |

**Date/Time Types**

**TIMESTAMP**

- It supports traditional UNIX timestamp with optional nanosecond precision.
- As Integer numeric type, it is interpreted as UNIX timestamp in seconds.
- As Floating point numeric type, it is interpreted as UNIX timestamp in seconds with decimal precision.
- As string, it follows java.sql.Timestamp format "YYYY-MM-DD HH:MM:SS.fffffffff" (9 decimal place precision)

**DATES**

The Date value is used to specify a particular year, month and day, in the form YYYY--MM--DD. However, it didn't provide the time of the day. The range of Date type lies between 0000--01--01 to 9999--12--31.

**STRING**

The string is a sequence of characters. It values can be enclosed within single quotes (') or double quotes (").

**Varchar**

The varchar is a variable length type whose range lies between 1 and 65535, which specifies that the maximum number of characters allowed in the character string.

**CHAR**

The char is a fixed-length type whose maximum length is fixed at 255.

**Complex Type**

| Type | Size | Range |
|---|---|---|
| Struct | It is similar to C struct or an object where fields are accessed using the "dot" notation. | struct('James','Roy') |
| Map | It contains the key-value tuples where the fields are accessed using array notation. | map('first','James','last','Roy') |
| Array | It is a collection of similar type of values that indexable using zero-based integers. | array('James','Roy') |

**Hive - Create Database**

In Hive, the database is considered as a catalog or namespace of tables. So, we can maintain multiple tables within a database where a unique name is assigned to each table. Hive also provides a default database with a name default.

**hive> show databases;**
hive> create database demo;

hive> create a database **if not exists** demo;

hive> drop database demo;

hive> show databases;

if we want to suppress the warning generated by Hive on creating the database with the same name, follow the below command

hive> drop database if exists demo;

**Create Tables**

hive> create table if not exists demo.employee (Id int, Name string , Salary float)

row format delimited

fields terminated by ',' ;

- While creating a table, we can add the comments to the columns and can also define the table properties.

hive> create table demo.new_employee (Id int comment 'Employee Id', Name string comment 'Employee Name', Salary float comment 'Employee Salary')
comment 'Table Description'
TBLProperties ('creator'='Ahmed Salem', 'created_at' = '2019-06-06 11:00:00');

- Let's see the metadata of the created table by using the following command: -

hive> describe new_employee;

- Hive allows creating a new table by using the schema of an existing table.

hive> create table if not exists demo.copy_employee

like demo.employee;

Here, we can say that the new table is a copy of an existing table.

**External Table**

The external table allows us to create and access a table and a data externally. The **external** keyword is used to specify the external table, whereas the **location** keyword is used to determine the location of loaded data.

As the table is external, the data is not present in the Hive directory. Therefore, if we try to drop the table, the metadata of the table will be deleted, but the data still exists.

To create an external table, follow the below steps: -

- Let's create a directory on HDFS by using the following command: -

```
hdfs dfs -mkdir /HiveDirectory
```

- Now, store the file on the created directory.

```
hdfs dfs  -put  hive/emp_details  /HiveDirectory
```

- Let's create an external table using the following command: -

```
hive> create external table emplist (Id int, Name string , Salary float)
row format delimited
 fields terminated by ','
location '/HiveDirectory';
```

Now, we can use the following command to retrieve the data: -

```
select * from emplist;
```

**Hive - Load Data**

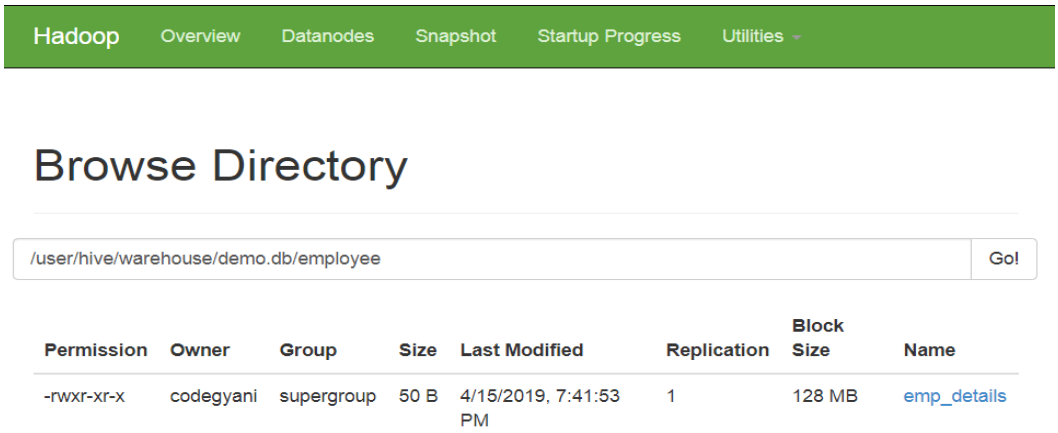
Once the internal table has been created, the next step is to load the data into it. So, in Hive, we can easily load data from any file to the database.

```
load data local inpath '/home/codegyani/hive/emp_details' into table demo.employee;
```

Here, **emp_details** is the file name that contains the data.

- Now, we can use the following command to retrieve the data from the database.

hive>select * from demo.employee;




- If we want to add more data into the current database, execute the same query again by just updating the new file name.

load data local inpath '/home/codegyani/hive/emp_details1' into table demo.employee;

- Let's check the data of an updated table: -

- In Hive, if we try to load unmatched data (i.e., one or more column data doesn't match the data type of specified table columns), it will not throw any exception. However, it stores the Null value at the position of unmatched tuple.
- Let's add one more file to the current table. This file contains the unmatched data.

Here, the third column contains the data of string type, and the table allows the float type data. So, this condition arises in an unmatched data situation.

- Now, load the data into the table.

load data local inpath '/home/codegyani/hive/emp_details2' into table demo.employee;

**Drop Table**

Hive facilitates us to drop a table by using the SQL **drop table** command. Let's follow the below steps to drop the table from the database.

- Let's check the list of existing databases by using the following command: -

```
hive> show databases;
```

- Now select the database from which we want to delete the table by using the following command: -

```
hive> use demo;
```

- Let's check the list of existing tables in the corresponding database.

```
hive> show tables;
```

- Now, drop the table by using the following command: -

```
hive> drop table new_employee;
```

- Let's check whether the table is dropped or not.

```
hive> show tables;
```

### Hive - Alter Table

In Hive, we can perform modifications in the existing table like changing the table name, column name, comments, and table properties. It provides SQL like commands to alter the table.

### Rename a Table

If we want to change the name of an existing table, we can rename that table by using the following signature: -

```
Alter table old_table_name rename to new_table_name;
```

- Let's see the existing tables present in the current database.

- Now, change the name of the table by using the following command: -

```
Alter table emp rename to employee_data;
```

- Let's check whether the name has changed or not.

**Adding column**

Alter table table_name add columns(column_name datatype);

- Let's see the schema of the table.
- Let's see the data of columns exists in the table.
- Now, add a new column to the table by using the following command: -

Alter table employee_data add columns (age int);

Let's see the updated schema of the table.

- Let's see the updated data of the table.

As we didn't add any data to the new column, hive consider NULL as the value.

**Change Column**

In Hive, we can rename a column, change its type and position. Here, we are changing the name of the column by using the following signature: -

Alter table table_name change old_column_name new_column_name  datatype;

- Let's see the existing schema of the table.
- Now, change the name of the column by using the following command: -

**Alter table employee_data change name first_name string;**

Let's check whether the column name has changed or not.

**Delete or Replace Column**

Hive allows us to delete one or more columns by replacing them with the new columns. Thus, we cannot drop the column directly.

- Let's see the existing schema of the table.
- Now, drop a column from the table.

alter table employee_data replace columns( id string, first_name string, age int);

Let's check whether the column has dropped or not.