HIVE

To carry out operations like querying and analyzing on a huge amount of data, Hadoop offers an open-source data warehouse system called Hive or Apache Hive.Hive is a mechanism through which we can access the data stored in Hadoop Distributed File System (HDFS).It provides an interface, similar to SQL, which enables you to create database and table to store data. In this way, we can achieve the MapReduce concept without explicitly writing the source code for it.

Hive also support a language called HiveQL, which is considered as the primary data processing method for Treasure Data. HiveQL automatically translates SQL-like queries into MapReduce jobs executed on Hadoop.

**Introducing Hive:**

Hive was created to make it possible for analysts with strong SQL skills (but meager Java programming skills) to run queries on the huge volumes of data that Facebook stored in HDFS. Today, Hive is a successful Apache project used by many organizations as a general-purpose, scalable data processing platform. Of course, SQL isn’t ideal for every big data problem—it’s not a good fit for building complex machine learning algorithms, for example—but it’s great for many analyses, and it has the huge advantage of being very well known in the industry. It should be noted that Hive is not a complete database, and it is not meant to be used in Online Transaction Processing System,such as online ticketing ,bank transaction,etc.It is mostly used in data warehousing kind of applications, where you need to perform batch processing on a huge amount of data. Typical examples of this kind of data include Weblogs ,cell data records, weather data ,etc.

**Hive architecture:**

****

The architecture of hive consists of various components. These components are described as follows:

**Command Line Interface(CLI)** – Allows you to submit queries to hive system for execution.

**Thrift Client-**The Hive Thrift Client makes it easy to run Hive commands from a wide range ofprogramming languages. Thrift bindings for Hive are available for C++, Java, PHP,Python, and Ruby.

**Driver-** Receives the submitted queries. This driver component creates a session handle for the submitted and then send the query to the compiler to generate an execution plan.

**Metastore- S**tores all the information related to the structure of the various tables and partitions in the data warehouse. It also includes column and column type information.It also contains information about the corresponding HDFS files where your data is stored.

**JDBC/ODBC**- This is a JDBC client that allows users to connect to HIVE and submit their job.

**Getting Started with Hive**

Type hive to launch the Hive shell:

hive>

The Hive Shell:

The shell is the primary way that we will interact with Hive, by issuing commands in HiveQL. HiveQL is Hive’s query language, a dialect of SQL. It is heavily influenced by MySQL, so if you are familiar with MySQL you should feel at home using Hive.

**Create Database**

Hive contains a default database named default.

Hive>show databases;

default

Like SQL, HiveQL is generally case-insensitive (except for string comparisons), so SHOW DATABASES; works equally well here.

hive> CREATE DATABASE mydb;

hive> SHOW DATABASES;

default

mydb

**Drop Database:**

Drop Database is a statement that drops all the tables and deletes the database. Its syntax is as follows:

**Hive primitive data types:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Literal examples** |
| TINYINT | 1-byte (8-bit) signed integer, from -128 to  127 | 1 |
| SMALLINT | 2-byte (16-bit) signed integer, from  -32,768 to 32,767 | 1 |
| INT | 4-byte (32-bit) signed integer, from  -2,147,483,648 to 2,147,483,647 | 1 |
| BIGINT | 8-byte (64-bit) signed integer, from  -9,223,372,036,854,775,808 to  9,223,372,036,854,775,807 | 1 |
| FLOAT | 4-byte (32-bit) single-precision floating point  number | 1.0 |
|  |  |  |
| DOUBLE | 8-byte (64-bit) double-precision floating point | 1.0 |
| BOOLEAN | Number  true/false value | TRUE |
| STRING | Character string | 'a', "a" |
| TIMESTAMP | Timestamp with nanosecond precision | 1325502245000, '2012-01-02  03:04:05.123456789' |
|  |  |  |

**Hive Complex data types:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Literal examples** |
| ARRAY | An ordered collection of fields. The fields must all be of the same type. | array(1, 2) |
| MAP | An unordered collection of key-value pairs. Keys must be primitives; values may be any type. For a particular map, the keys must be the same type, and the values must be the same type. | map('a', 1, 'b', 2) |
| STRUCT | A collection of named fields. The fields may be of different types. | struct('a', 1, 1.0) |

**Hive Table**

There is basically two kinds of tables in Hive Managed Tables and External Table.When you create a table in Hive, by default Hive will manage the data, which means that Hive moves the data into its warehouse directory. Alternatively, you may create an external table, which tells Hive to refer to the data that is at an existing location outside the warehouse directory.The difference between the two types of the table is seen in the LOAD and DROP semantics.Let’s consider a managed table first.When you load data into a managed table, it is moved into Hive’s warehouse directory.

**Managed Table:**

**Example:**

**CREATE TABLE managed\_table (dummy STRING);**

**LOAD DATA INPATH '/user/tom/data.txt' INTO table managed\_table;**

Will move the file hdfs://user/tom/data.txt into Hive’s warehouse directory for the managed\_table table, which is hdfs://user/hive/warehouse/managed\_table.

If the table is later dropped, using:

**DROP TABLE managed\_table;**

Then the table, including its metadata and its data, is deleted. It bears repeating that since the initial LOAD performed a move operation, and the DROP performed a delete operation, the data no longer exists anywhere. This is what it means for Hive to manage the data.

**External Table:**

An external table behaves differently. You control the creation and deletion of the data.The location of the external data is specified at table creation time:

**CREATE EXTERNAL TABLE external\_table (dummy STRING)**

**LOCATION '/user/tom/external\_table';**

**LOAD DATA INPATH '/user/tom/data.txt' INTO TABLE external\_table;**

With the **EXTERNAL** keyword, Hive knows that it is not managing the data, so it doesn’t move it to its warehouse directory. Indeed, it doesn’t even check if the external location exists at the time it is defined. This is a useful feature since it means you can create the data lazily after creating the table.When you drop an external table, Hive will leave the data untouched and only delete the metadata.So how do you choose which type of table to use? In most cases, there is not much difference between the two (except of course for the difference in DROP semantics), so it is a just a matter of preference. As a rule of thumb, if you are doing all your processing with Hive, then use managed tables, but if you wish to use Hive and other tools on the same dataset, then use external tables. A common pattern is to use an external table to access an initial dataset stored in HDFS (created by another process), then use a Hive. transform to move the data into a managed Hive table. This works the other way around, too—an external table (not necessarily on HDFS) can be used to export data from Hive for other applications to use.Another reason for using external tables is when you wish to associate multiple schemas with the same dataset.

**Load Data into Table:**

Generally, after creating a table in SQL, we can insert data using the Insert statement. But in Hive, we can insert data using the LOAD DATA statement .While inserting data into Hive, it is better to use LOAD DATA to store bulk records. There are two ways to load data: one is from local file system and second is from Hadoop file system. The syntax for load data is as follows:

**Local mode:**

LOAD DATA LOCAL INPATH 'filepath' INTO TABLE tablename;

**HDFS mode:**

LOAD DATA INPATH 'hdfs\_filepath' INTO TABLE tablename;

**HIVE Queries**

Let us assume you need to create a table named emp using CREATE TABLE statement.The following table lists the fields and their data types in emp table:

Sl.no field Name Data Type

1 Eid int

2 Name String

3 Salary Float

4 designation String

**Create emp Table**

hive> CREATE TABLE IF NOT EXISTS emp ( eid int, name String,

> salary String, destination String)

> COMMENT ‘Employee details’

> ROW FORMAT DELIMITED

> FIELDS TERMINATED BY ‘\t’

> LINES TERMINATED BY ‘\n’ --Optional

> STORED AS TEXTFILE; --Optional

If we add the option IF NOT EXISTS, Hive ignores the statement in case the table already exists.

**Load data into emp table:**

We will insert the following data into the table. It is a text file named **sample.txt** in **/home/user** directory.

hive> LOAD DATA LOCAL INPATH '/home/user/sample.txt'

> INTO TABLE emp;