**OVERVIEW OF HIVE (HQL)**

**APACHE HIVE**

**INTRODUCTION:**

Apache Hive is a Data Warehousing tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy. Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive.

Hive is targeted towards users who are comfortable with SQL. It is similar to SQL and called HiveQL, used for managing and querying structured data. Apache Hive is used to abstract complexity of Hadoop. This language also allows traditional map/reduce programmers to plug in their custom mappers and reducers. The popular feature of Hive is that there is no need to learn Java.

Hive is a SQL engine on top of hadoop designed for SQL many people to run map reduce jobs through SQL like queries. Hive allows developers to impose a logical relational schema on various file formats and physical storage mechanisms within or outside the hadoop cluster. SQL like queries are run against those schemas as Hadoop Map Reduce jobs. With limited write capabilities and interactivity, Hive is meant for the execution of batch transformations and large analytical queries.

Hive is best suited for batch jobs instead of working with web log data and append-only data. Hive cannot work for Online Transaction Processing work types since it does not provide querying in real time and row-level updates.

**HIVE IS NOT**

* A relational database
* A design for Online Transaction Processing (OLTP)
* A language for real-time queries and row-level updates

**IMPORTANT CHARACTERISTICS OF HIVE**

* In Hive, tables and databases are created first and then data is loaded into these tables.
* It stores schema in a database and processed data into HDFS.
* Hive supports four file formats those are **TEXTFILE, SEQUENCEFILE, ORC and RCFILE** (Record Columnar File).
* Hadoop's programming works on flat files. So, Hive can use directory structures to "partition" data to improve performance on certain queries.
* Hive also supports table compressions like **snappy, zlip** and **gzip** etc.
* A new and important component of Hive i.e. Metastore used for storing schema information.
* We can run Ad-hoc queries for the data analysis using Hive.
* Hive also supports bucketing concept so it helps in performance gain.
* It is familiar, fast, scalable, and extensible.
* Hive also supports integration with JDBC and other BI tools.
* Hive supports custom specific UDF (User Defined Functions) for data cleansing, filtering, etc. According to the requirements of the programmers one can define Hive UDFs.

**HIVE VS SQL**

|  |  |
| --- | --- |
| **HIVE** | **SQL** |
| Hive is a SQL-like scripting language built on Map Reduce. | According to ANSI, SQL is the standard language for RDMBS, used to communicate with databases |
| Used for analytics | Used for transactional processing(OLTP) & analytics |
| Data per query in PBs | Data per query in GBs |
| Faster execution while performing analytics on Huge data sets compared to SQL | Slower execution while performing analytics on huge data sets compared to HIVE |
| No Normalization required | Supports Normalization |
| It’s very easily scalable at low cost | Not much scalable, costly scale up |

**Hive server 2:**

HiveServer2 (HS2) is a service that enables clients to execute queries against Hive. HiveServer2 is the successor to HiveServer1 which has been deprecated. HS2 supports multi-client concurrency and authentication. It is designed to provide better support for open API clients like JDBC and ODBC.

Hive Service is nothing but daemon which runs on your client node which sends requests to Hive Server.

**Hive Thrift server**:

Thrift is an RPC framework for building cross-platform services. Its stack consists of 4 layers: Server, Transport, Protocol, and Processor.

The Thrift-based Hive service is the core of HS2 and responsible for servicing the Hive queries (e.g., from Beeline). In simple terms Hive server is based on thrift protocols which sends queries from hive client i.e., your command line interface or from HUE interface to the underlying data which can be in your HDFS or any other data sources.

Usage

* When you query any hive tables or database, in background automatically your requests is transferred between hive service and hive server.
* When you want to create your own service or project you can use thrift protocols which will help you in creating layers, think this as you are creating your user defined functions using libraries, so in that case libraries will be thrift.

Architecture of Hive

The following component diagram depicts the architecture of Hive:



This component diagram contains different units. The following table describes

each unit:

**1) 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).

**2) 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.

**3) 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.

**4)** **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.

**5) 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 steps defines how Hive interacts with Hadoop framework.

**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.

**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:**

Hive supports different data types to be used in table columns. The data types supported by Hive can be broadly classified in Primitive and Complex data types.

The primitive data types supported by Hive are listed below:

1) Numeric Types:

* TINYINT (1-byte signed integer, from -128 to 127)
* SMALLINT (2-byte signed integer, from -32,768 to 32,767)
* INT (4-byte signed integer, from -2,147,483,648 to 2,147,483,647)
* BIGINT (8-byte signed integer, from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
* FLOAT (4-byte single precision floating point number)
* DOUBLE (8-byte double precision floating point number)
* DECIMAL (Hive 0.13.0 introduced user definable precision and scale)

#### 2) Date/Time Types:

* TIMESTAMP
* DATE

3) String Types:

* STRING
* VARCHAR
* CHAR

4) Misc Types:

* BOOLEAN
* BINARY

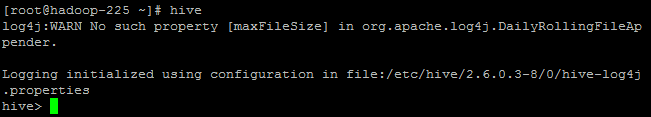
#### 5) Complex Types:

* arrays: ARRAY<data\_type>
* maps: MAP<primitive\_type, data\_type>
* structs: STRUCT<col\_name : data\_type [COMMENT col\_comment], ...>
* union: UNIONTYPE<data\_type, data\_type ...>

## ****Hive Shell:****

## Hive provides a default interface, where it allows users to run Hive commands. The CLI (Command Line Interface) is the default Hive shell service which allows users to work on Hive programs.

The following are the commands used to start the Hive shell:

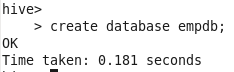


## ****Creating a Database:****

The purpose of creating a database before creating a table is that the database in Hive is a namespace, where a table or collection of tables should be placed to work on Hive queries. Here the database and the schema represents the same category.

The command to create a database are as follows:

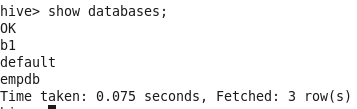
hive> CREATE DATABASE EMPdb;



## ****Listing Databases:****

The show databases command displays the list of databases in HDFS so that the user can select the database he wants to work with.

hive> SHOW DATABASES;



## ****Using a Database:****

The command to use database is USE. The use database command allows the user to utilize the mentioned database to work with, where he can create new tables in that database and alter particular table contents too.

hive> USE EMPdb;

https://s3.amazonaws.com/acadgildsite/wordpress_images/bigdatadeveloper/Hive+Beginner%27s+guide/using+databases.PNG

**Creating a Table:**

The command for creating a table is ‘Create’.

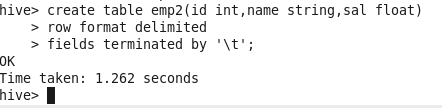
Before creating a table, a user must know the following details:

* The create table command allows the user to create a new table with user input attributes/columns.
* Row format delimited Fields terminated by ‘\t’ – This line informs Hive that each column in the file is separated by a tab.

hive> Create table emp(id int, name string, sal float)

        > row format delimited

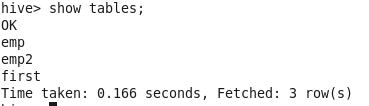
        > fields terminated by ‘\t’ ;



## ****List Tables:****

The ‘show tables’ command displays the list of tables present in a particular database.

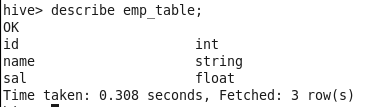
Hive> show Tables;



## ****Describe Schema of the Table:****

The describe command is similar to adesc command in SQL. The describe command displays table definition/metadata of the mentioned table and the users can see column names and data types associated to that particular column.

Hive>DESCRIBE emp\_table;



## ****Load a File from the Local File System:****

We know that Hadoop is designed to work only on huge datasets instead of few selected rows/columns. So, the user needs to load the entire file to HDFS and process it using Hive commands.

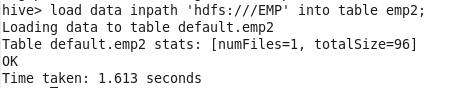
The command used to load file from local system to HDFS is as follows:

Hive>load data local inpath<filename> into table<tablename>

## ****Load File from HDFS:****

The command used for loading file from HDFS is ‘load data’.

Hive>load data inpath<filename> into table<tablename>

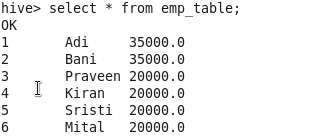


## ****Show Table Contents:****

## 

Once the selected file is loaded to HDFS, the contents of the table can be checked using the select statement.

The select \* statement of a table displays all the columns which are present in that table.



## ****Commonly Used Alter Commands:****

Some of the commonly used alter commands in Hive are as follows:

**Renaming the Current Table:**

hive> ALTER TABLE EMP RENAME TO EMP\_TABLE

https://i2.wp.com/s3.amazonaws.com/acadgildsite/wordpress_images/bigdatadeveloper/Hive+Beginner%27s+guide/alter+command+for+renaming+a+existing+table+name.PNG?resize=368%2C37&ssl=1

**Adding New Columns to an Existing Table:**

hive> ALTER TABLE EMP\_TABLE ADD COLUMNS (YOJ DATE)

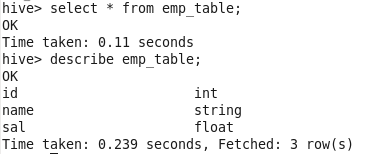
https://i1.wp.com/s3.amazonaws.com/acadgildsite/wordpress_images/bigdatadeveloper/Hive+Beginner%27s+guide/alter+command+to+add+a+new+column.PNG?resize=428%2C39&ssl=1

## ****Truncating a Table:****

The Truncating table command allows users to delete only the content values in the table and not the table schema.

https://i1.wp.com/s3.amazonaws.com/acadgildsite/wordpress_images/bigdatadeveloper/Hive+Beginner%27s+guide/truncate+table+1.PNG?resize=258%2C52&ssl=1

After performing the Truncating command, we can see that the table schema is unaffected but the contents of the table have been deleted. This is how you can truncate/delete table contents without affecting the table schema.



## ****Dropping a Database:****

The command to drop a database is ‘drop database’. The drop database database\_name allows users to drop a particular database from the HDFS memory.

hive> DROP DATABASE EMPdb;

https://s3.amazonaws.com/acadgildsite/wordpress_images/bigdatadeveloper/Hive+Beginner%27s+guide/dropping+a+database.PNG

**Performance Tuning In Hive:**

Apache Hive is a very powerful tool for analyzing data, and it supports batch and interactive data processing. It is one of the most-used techniques by data analysts and data scientists. It is very important that you know how to improve the performance of query when you are processing petabytes of data.

If the Hive code is not written properly, you may face timing in hive query execution. And so hive performance tuning is very important.

When you do Hive query optimization, it helps the query to execute at least by 50%. If your query is not optimized, a simple select statement can take very long to execute.

There are many methods for Hive performance tuning and you should know some of these to do well with the queries

**1) Use ORC File Format:**

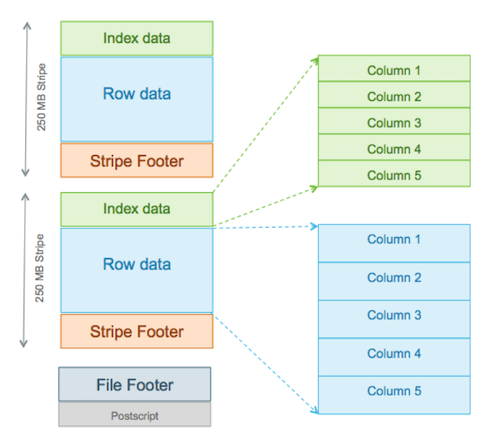
ORC File Format Full Form is Optimized Row Columnar File Format.ORC File format provides very efficient way to store relational data then RC file, by using ORC File format we can reduce the size of original data up to 75%.Comparing to Text, Sequence, RC file formats ORC is better.

Using ORC files improves performance when Hive is reading, writing, and processing data. Comparing to Text, Sequence and RC and ORC shows better performance than Text and Sequence File formats.

Again Comparing to RC and ORC File formats always ORC is better.ORC takes less time to access the data comparing to RC File Format and ORC takes less space to store data. However, the ORC file increases CPU overhead by increasing the time it takes to decompress the relational data.ORC File format feature comes with the Hive 0.11 version and cannot be used with previous versions.

The ORC file format provides the following advantages:

* **Efficient compression**: Stored as columns and compressed, which leads to smaller disk reads. The columnar format is also ideal for vectorization optimizations in Tez.
* **Fast reads**: ORC has a built-in index, min/max values, and other aggregates that cause entire stripes to be skipped during reads. In addition, predicate pushdown pushes filters into reads so that minimal rows are read. And Bloom filters further reduce the number of rows that are returned.
* **Proven in large-scale deployments**: Facebook uses the ORC file format for a 300+ PB deployment.



### Stripe Structure

As shown in the diagram, each stripe in an ORC file holds index data, row data, and a stripe footer.

The **stripe footer** contains a directory of stream locations. **Row data** is used in table scans.

**Index data** includes min and max values for each column and the row positions within each column. (A bit field or bloom filter could also be included.) Row index entries provide offsets that enable seeking to the right compression block and byte within a decompressed block.  Note that ORC indexes are used only for the selection of stripes and row groups and not for answering queries.

Having relatively frequent row index entries enables row-skipping within a stripe for rapid reads, despite large stripe sizes. By default every 10,000 rows can be skipped.

**Syntax:**

create table employee\_orc(

id int, name string)

row format delimited

fields terminated by '|'

stored as ORC;

**2) Use Table Compressions (snappy):**

Compression techniques reduce the amount of data being transferred and so reduces the data transfer between mappers and reducers.

For better result, you need to perform compression at both mapper and reducer side separately. Although gzip is considered as the best compression format but beware that it is not splittable and so should be applied with caution.

Other formats are snappy, lzo, bzip, etc.

**Syntax:**

CREATE TABLE A\_ORC (

customerID int, name string,etc

) STORED AS ORC tblproperties (“orc.compress" = “SNAPPY”);

**3) Use TEZ Execution Engine:**

Apache TEZ is an execution engine used for faster query execution. It fastens the query execution time to around 1x-3x times.

To use TEZ execution engine, you need to enable it instead of default Map-Reduce execution engine. TEZ can be enabled using the below query.

## Set hive.execution.engine=tez;

If you are using Cloudera/Hortonworks, then you will find TEZ option in the Hive query editor as well.

**4) Use Map side join:**

Map side join is a process where joins between two tables are performed in the Map phase without the involvement of Reduce phase.

Map-side Joins allows a table to get loaded into memory ensuring a very fast join operation, performed entirely within a mapper and that too without having to use both map and reduce phases.

In case your queries frequently run with small table joins, you might see a very substantial decrease in the time taken to compute the queries after usage of map-side joins.

## By setting the following property to true.

## hive.auto.convert.join=true;

You can set the small file size by using the following property:

.

## hive.mapjoin.smalltable.filesize= (default it will be 25MB)

**5) Use Partition:**

Partition is a useful concept in Hive. It is used to divide the large table based on certain column so that the whole data can be divided into small chunks. It allows you to store the data under sub-directory inside a table.

Selecting the partition table is always a critical decision, and you need to take care of future data as well as the volume of data as well. For example, if you have data of a particular location then partition based on state can be one of the ideal choices.

Here is the syntax to create partition table-

CREATE TABLE countrydata\_partition

(Id int, countryname string, population int, description string)

PARTITIONED BY (country VARCHAR(64), state VARCHAR(64))

row format delimited

fields terminated by ‘\t’

stored AS ORC;

**There are two types of partition in Hive-**

* Static partition
* Dynamic partition

Static partition is the default one. To use dynamic partition in Hive, you need to set the following property-

set hive.exec.dynamic.partition=true;

set hive.exec.dynamic.partition.mode=nonstrict;

**6) Use Bucketing:**

If you have more number of columns on which you want the partitions, bucketing in the hive can be a better option. We use CLUSTERED BY command to divide the tables in the bucket.

Here is the syntax to create bucketed table-

CREATE TABLE emp\_bucketed\_table(

ID int, name string, address string, salary string )

COMMENT ‘this is a bucketed table’

PARTITIONED BY (country VARCHAR(64))

CLUSTERED BY (state) INTO 10 BUCKETS

STORED AS TEXTFILE;

To enable bucketing in Hive, you need to set the following property-

SET hive.enforce.bucketing=true;

**7) Vectorization:**

Vectorization improves the query performance of all the operation like scans, aggregations, filters and joins, by performing them in batches of 1024 rows at once instead of single row each time.

Again you will have to set some parameter to enable vectorization-

set hive.vectorized.execution.enabled = true;

set hive.vectorized.execution.reduce.enabled = true;

**8) Parallel execution:**

As we know, Hive converts the queries into different stages during execution. These stages are usually getting executed one after the other and thus increases the time of execution. Below are some of the normal steps involved-

• MapReduce stage

• Sampling stage

• Limit stage

• Merge stage etc.

But the good thing is, you can set some of this independent stage to process parallel. This is a parallel execution in Hive. For this, you need to set the below properties to true-

Set hive.exec.parallel = true;

**9) Cost Based Query Optimization:**

Hive optimizes each query’s logical and physical execution plan before submitting for final execution.

These optimizations are not based on the cost of the query – that is, until now.

A recent addition to Hive, CBO, performs further optimizations based on query cost resulting in potentially different decisions: how to order joins, which type of join to perform, degree of parallelism and others.

To use CBO, set the following parameters at the beginning of your query:

set hive.cbo.enable=true;

set hive.compute.query.using.stats=true;

set hive.stats.fetch.column.stats=true;

set hive.stats.fetch.partition.stats=true;

Then, prepare the data for CBO by running Hive’s “analyze” command to collect various statistics on the tables for which we want to use CBO.  
For example, in a table tweets we want to collect statistics about the table and about 2 columns: “sender” and “topic” analyze table tweets compute statistics  
analyze table tweets compute statistics for columns sender, topic;  
With HIVE 0.14 (on HDP 2.2) the analyze command works much faster, and you don’t need to specify each column, so you can just issue:  
analyze table tweets compute statistics for columns;  
That’s it. Now executing a query using this table should result in a different execution plan that is faster because of the cost calculation and different execution plan created by Hive.

**10)** **Improve join performance on bucketed table**

The next example extends the above discussion of using clustered fields (bucketed fields) to improve join performance.

The optimization is turned off by default for many versions of Hive. Enable the optimization with the following settings.

set hive.optimize.bucketmapjoin=true;

set hive.optimize.bucketmapjoin.sortedmerge=true;

**11) Skew join:**

A skew join is used when there is a table with skew data in the joining column.

A skew table is a table that is having values that are present in large numbers in the table compared to other data.

Skew data is stored in a separate file while the rest of the data is stored in a separate file.

If there is a need to perform a join on a column of a table that is appearing quite often in the table.

The data for that particular column will go to a single reducer, which will become a bottleneck while performing the join.

To reduce this, a skew join is used.

set hive.optimize.skewjoin=true;

set hive.skewjoin.key=100000;