Hive Tutorial – Hive Architecture and NASA Case Study

Recommended by 215 users

Ashish Bakshi (https://www.edureka.co/blog/author/ashishbedureka-co/) | Jul 17,2018

Apache Hive Tutorial: Introduction

In this Hive tutorial blog, we will be discussing about Apache Hive in depth. Apache Hive is a data warehousing tool in the ***Hadoop Ecosystem (https://www.edureka.co/blog/hadoop-ecosystem)***, which provides SQL like language for querying and analyzing Big Data. The motivation behind the development of Hive is the friction-less learning path for SQL developers & analyst. Hive is not only a saviour for people from non-programming background, but it also reduces the work of programmers who spend long hours writing MapReduce programs. In this Apache Hive Tutorial blog, I will talk about:

- **What is Hive?**
- **Story of Apache Hive – From Facebook to Apache**
- **Advantages of Apache Hive**
- **Apache Hive – NASA Case Study**
- **Apache Hive Architecture**
- **Metastore Configuration**
- **Hive Data Model**

Apache Hive Tutorial: What is Hive?

Apache Hive is a data warehouse system built on top of Hadoop and is used for analyzing structured and semi-structured data. Hive abstracts the complexity of Hadoop MapReduce. Basically, it provides a mechanism to project structure onto the data and perform queries written in HQL (Hive Query Language) that are similar to SQL statements. Internally, these queries or HQL gets converted to map reduce jobs by the Hive compiler. Therefore, you don't need to worry about writing complex MapReduce programs to process your data using Hadoop. It is targeted towards users who are comfortable with SQL. Apache Hive supports Data Definition Language (DDL), Data Manipulation Language (DML) and User Defined Functions (UDF).

Hive Tutorial for Beginners | Understanding Hive In Depth | Edureka

SQL + Hadoop MapReduce = HiveQL

Apache Hive Tutorial: Story of Hive – from Facebook to Apache

Announcing the Ridiculously Committed Mentor Award Nominate Now

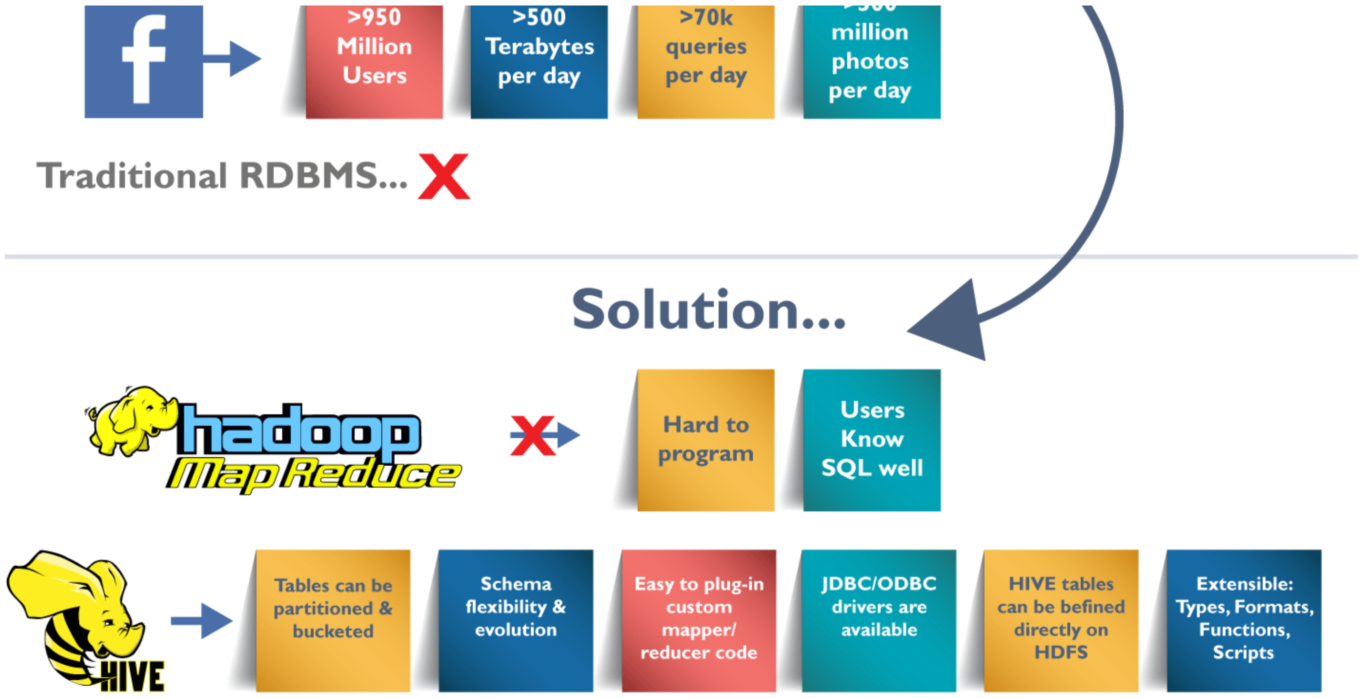

>950 Million Users
>500 Terabytes per day
>70k queries per day
>300 million photos per day
Traditional RDBMS...
Solution...
hadoop MapReduce
Hard to program
Users Know SQL well
HIVE
Tables can be partitioned & bucketed
Schema flexibility & evolution
Easy to plug-in custom mapper/ reducer code
JDBC/ODBC drivers are available
HIVE tables can be befined directly on HDFS
Extensible: Types, Formats, Functions, Scripts

Fig*: Hive Tutorial – Facebook use case*

Challenges at Facebook: Exponential Growth of Data

Before 2008, all the data processing infrastructure in Facebook was built around a data warehouse based on commercial RDBMS. These infrastructures were capable enough to suffice the needs of Facebook at that time. But, as the data started growing very fast, it became a huge challenge to manage and process this huge dataset. According to a Facebook article, the data scaled from a 15 TB data set in 2007 to a 2 PB data in 2009. Also, many Facebook products involve analysis of the data like Audience Insights, Facebook Lexicon, Facebook Ads, etc. So, they needed a scalable and economical solution to cope up with this very problem and, therefore started using the Hadoop framework.

Democratizing Hadoop – MapReduce

But, as the data grew, the complexity of Map-Reduce codes grew proportionally. So, training people with a non-programming background to write MapReduce programs became difficult. Also, for performing simple analysis one has to write a hundred lines of MapReduce code. Since, SQL was widely used by engineers and analysts, including Facebook, therefore, putting SQL on the top of Hadoop seemed a logical way to make Hadoop accessible to users with SQL background.

Hence, the ability of SQL to suffice for most of the analytic requirements and the scalability of Hadoop gave birth to **Apache Hive** that allows to perform SQL like queries on the data present in HDFS. Later, the Hive project was open sourced in August' 2008 by Facebook and is freely available as Apache Hive today.

Now, let us look at the features or advantages of Hive that makes it so popular.

Apache Hive Tutorial: Advantages of Hive

- Useful for people who aren't from a programming background as it eliminates the need to write complex MapReduce program.
- **Extensible** and **scalable** to cope up with the growing volume and variety of data, without affecting performance of the system.
- It is as an efficient ETL (Extract, Transform, Load) tool.
- Hive supports any client application written in Java, PHP, Python, C++ or Ruby by exposing its **Thrift server**. (You can use these client – side languages embedded with SQL for accessing a database such as DB2, etc.).
- As the metadata information of Hive is stored in an RDBMS, it significantly reduces the time to perform semantic checks during query execution.

Learn Hive From Industry Experts
(https://www.edureka.co/big-data-and-hadoop)

Apache Hive Tutorial: Where to use Apache Hive?

Apache Hive takes advantage of both the worlds i.e. SQL Database System and ***Hadoop – MapReduce (https://www.edureka.co/blog/mapreduce-tutorial/)*** framework. Therefore, it is used by a vast multitude of companies. It is mostly used for data warehousing where you can perform analytics and data mining that does not require real time processing. Some of the fields where you can use Apache Hive are as follows:

- Data Warehousing
- Ad-hoc Analysis

As it is said, you can't clap with one hand only i.e. You can't solve every problem with a single tool. Therefore, you can couple Hive with other tools to use it in many other domains. For example, Tableau along with Apache Hive can be used for Data Visualization, Apache Tez integration with Hive will provide you real time processing capabilities, etc.

Moving ahead in this Apache Hive Tutorial blog, let us have a look at a case study of NASA where you will get to know how Hive solved the problem that NASA scientists

Hive Tutorial: NASA Case Study

A climate model is a mathematical representation of climate systems based on various factors that impacts the climate of the Earth. Basically, it describes the interaction of various drivers of climate like ocean, sun, atmosphere, etc. to provide an insight into the dynamics of the climate system. It is used to project climate conditions by simulating the climate changes based on factors that affect climate. NASA's Jet Propulsion Laboratory has developed Regional Climate Model Evaluation System (RCMES) for analysis and evaluation of the climate output model against remote sensing data present in various external repositories.

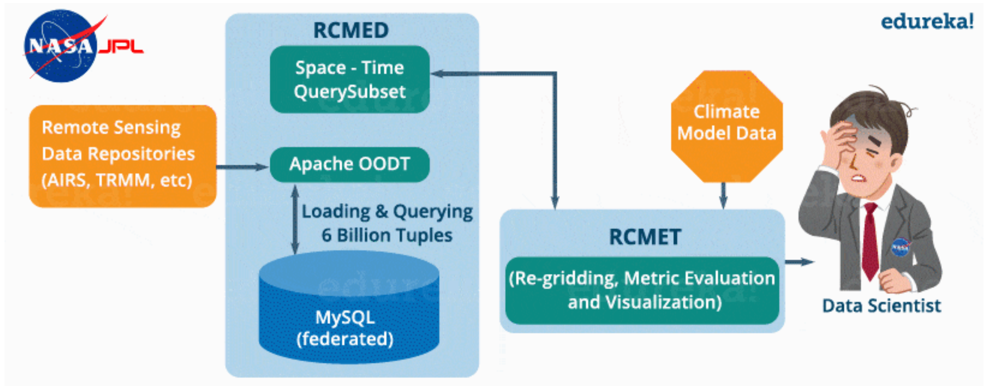
The RCMES (Regional Climate Model Evaluation System) has two components:

- ***RCMED (Regional Climate Model Evaluation Database):***

  It is a scalable cloud database that loads the remote sensing data and reanalysis data which are related to climate using extractors like Apache OODT extractors, Apache Tika, etc. Finally, it transforms the data as the data point model which is of the form (latitude, longitude, time, value, height) and stores it into My SQL database. The client can retrieve the data present in RCMED by performing Space/Time queries. The description of such queries is not relevant for us now.

- ***RCMET (Regional Climate Model Evaluation Toolkit):***

  It provides the user an ability to compare the reference data present in the RCMED with the climate model output data fetched from some other sources to perform different kinds of analysis and evaluation. You can refer to the image given below to understand the architecture of RCMES.


NASA JPL
RCMED
Space - Time QuerySubset
Remote Sensing Data Repositories (AIRS, TRMM, etc)
Apache OODT
Loading & Querying 6 Billion Tuples
MySQL (federated)
edureka!
Climate Model Data
RCMET
(Re-gridding, Metric Evaluation and Visualization)
Data Scientist

The reference data in the RCMED comes from satellite-based remote sensing, according to the different parameters required for climate model evaluation. For example – AIRS (Atmospheric Infrared Sounder) provide parameters like surface air temperature, temperature and Geopotential, TRMM (Tropical Rainfall Measurement Mission) provides monthly precipitation, etc.

Problems faced by NASA using the MySQL Database System:

- After loading the MySQL database with 6 billion tuples of the form (latitude, longitude, time, data point value, height), the system crashed as shown in the image above.
- Even after dividing the whole table into smaller subsets, the system generated huge overhead while processing the data.

So, they needed a scalable solution that can store and process this huge amount of data with SQL like querying capability. Finally, they decided to use Apache Hive to overcome the problems stated above.

How Apache Hive can solve the problem?

Now, let's see, what are those features that convinced NASA's JPL team to include Apache Hive as an integral part in their solution strategy:

- Since, Apache Hive runs on top of Hadoop, it is scalable and can process data in a distributed and parallel fashion.
- It provides Hive Query Language which is similar to SQL and hence easy to learn.

Deployment of Hive:

The following image explains the RCMES Architect with Apache Hive integration:

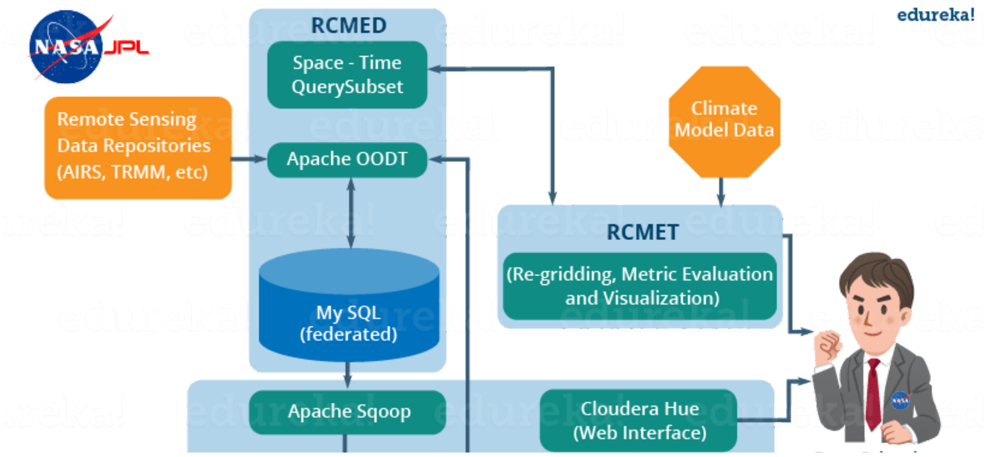

NASA JPL
RCMED
Space - Time QuerySubset
Remote Sensing Data Repositories (AIRS, TRMM, etc)
Apache OODT
My SQL (federated)
edureka!
Climate Model Data
RCMET
(Re-gridding, Metric Evaluation and Visualization)
Apache Sqoop
Cloudera Hue (Web Interface)

Fig*: Hive Tutorial – RCMES Architecture with Apache Hive*

The above image shows the deployment of apache hive in RCMES. Following steps were taken by the NASA team while deploying Apache Hive:

- They installed Hive using Cloudera and Apache Hadoop as shown in the above image.
- They used Apache Sqoop to ingest data into the Hive from MySQL database.
- Apache OODT wrapper was implemented to perform queries on Hive and retrieve the data back to RCMET.

Initial Benchmarking Observations with Hive:

- Initially they loaded 2.5 billion data points into a single table and performed a count query. For example, *Hive*> select count (datapoint_id) from dataPoint. It took 5-6 minutes to count all the records (15–17 minutes for the full 6.8 billion records).
- The reduce phase was fast, but the map phase took 95% of total processing time. They were using six (**4x quad-core**) systems with **24 GB RAM** (approx.) in each of the systems.
- Even after adding more machines, changing the HDFS block size (64 MB, 128 MB, 256 MB) and altering many other configuration variables (io.sort.factor, io.sort.mb), they did not get much success in reducing the time to complete the count.

Inputs from Members of Hive Community:

Finally, members of the Hive community came to the rescue and provided various insights to solve the issues with their current Hive implementations:

- They mentioned that HDFS read speed is approximately **60 MB/s** as compared to **1GB/s** in case of a local disc, depending on network capacity and workload on NameNode.
- The members suggested that **16 mappers** will be required in their current system to match with the I/O performance of a local non-Hadoop task.
- They also suggested to reduce the **split-size** for each mapper to increase the number of mappers and therefore, providing more parallelism.
- Finally, the community members told them to **use count (1)** instead of referring to **count (datapoint_id)**. This is because in case of count (1), there is no reference column and therefore, no decompression and deserialization takes place while performing the count.

Finally, NASA was able to tune their Hive cluster up to their expectations by taking into account all the suggestions given by the Hive community members. And therefore, they were able to query billions of rows in just 15 seconds using the system configurations mentioned above.

Master Hive In Our Hadoop Course
(https://www.edureka.co/big-data-and-hadoop)

Apache Hive Tutorial: Hive Architecture and its Components

The following image describes the Hive Architecture and the flow in which a query is submitted into Hive and finally processed using the MapReduce framework:

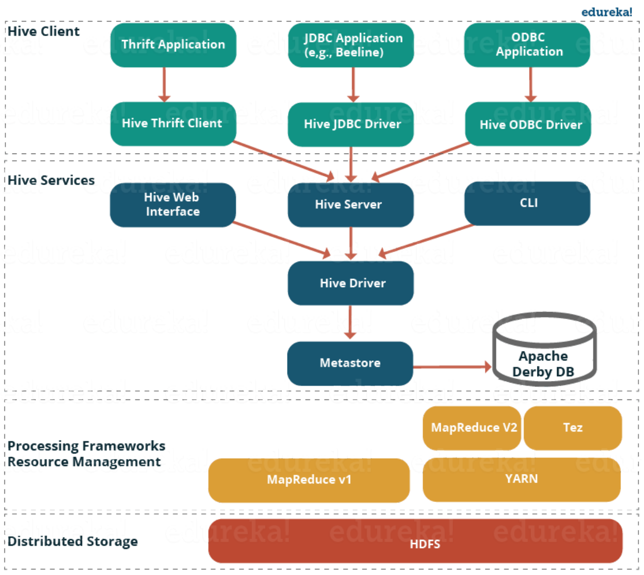


Fig*: Hive Tutorial – Hive Architecture*

As shown in the above image, the Hive Architecture can be categorized into the following components:

- **Hive Clients:** Hive supports application written in many languages like Java, C++, Python etc. using JDBC, Thrift and ODBC drivers. Hence one can always write hive client application written in a language of their choice.
- **Hive Services:** Apache Hive provides various services like CLI, Web Interface etc. to perform queries. We will explore each one of them shortly in this Hive tutorial blog.
- **Processing framework and Resource Management:** Internally, Hive uses Hadoop MapReduce framework as de facto engine to execute the queries. ***Hadoop MapReduce framework (https://www.edureka.co/blog/mapreduce-tutorial/)*** is a separate topic in itself and therefore, is not discussed here.
- **Distributed Storage:** As Hive is installed on top of Hadoop, it uses the underlying HDFS for the distributed storage. You can refer to the ***HDFS blog (https://www.edureka.co/blog/hdfs-tutorial)*** to learn more about it.

Now, let us explore the first two major components in the Hive Architecture:

Apache Hive supports different types of client applications for performing queries on the Hive. These clients can be categorized into three types:

- *Thrift Clients:* As Hive server is based on Apache Thrift, it can serve the request from all those programming language that supports Thrift.
- *JDBC Clients:* Hive allows Java applications to connect to it using the JDBC driver which is defined in the class org.apache.hadoop.hive.jdbc.HiveDriver.
- *ODBC Clients:* The Hive ODBC Driver allows applications that support the ODBC protocol to connect to Hive. (Like the JDBC driver, the ODBC driver uses Thrift to communicate with the Hive server.)

2. Hive Services:

Hive provides many services as shown in the image above. Let us have a look at each of them:

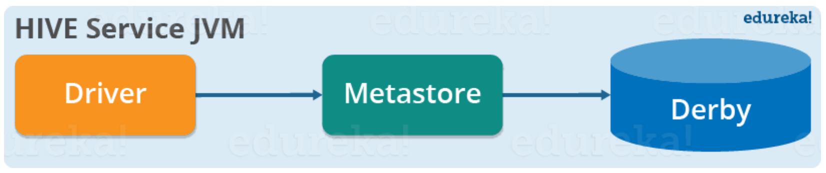
- **Hive CLI (Command Line Interface):** This is the default shell provided by the Hive where you can execute your Hive queries and commands directly.
- **Apache Hive Web Interfaces:** Apart from the command line interface, Hive also provides a web based GUI for executing Hive queries and commands.
- **Hive Server:** Hive server is built on Apache Thrift and therefore, is also referred as Thrift Server that allows different clients to submit requests to Hive and retrieve the final result.
- **Apache Hive Driver:** It is responsible for receiving the queries submitted through the CLI, the web UI, Thrift, ODBC or JDBC interfaces by a client. Then, the driver passes the query to the compiler where parsing, type checking and semantic analysis takes place with the help of schema present in the metastore. In the next step, an optimized logical plan is generated in the form of a DAG (Directed Acyclic Graph) of map-reduce tasks and HDFS tasks. Finally, the execution engine executes these tasks in the order of their dependencies, using Hadoop.
- **Metastore:** You can think metastore as a central repository for storing all the Hive metadata information. Hive metadata includes various types of information like structure of tables and the partitions along with the column, column type, serializer and deserializer which is required for Read/Write operation on the data present in HDFS. The metastore comprises of two fundamental units:
  - A service that provides metastore access to other Hive services.
  - Disk storage for the metadata which is separate from HDFS storage.

Now, let us understand the different ways of implementing Hive metastore in the next section of this Hive Tutorial.

Apache Hive Tutorial: Metastore Configuration

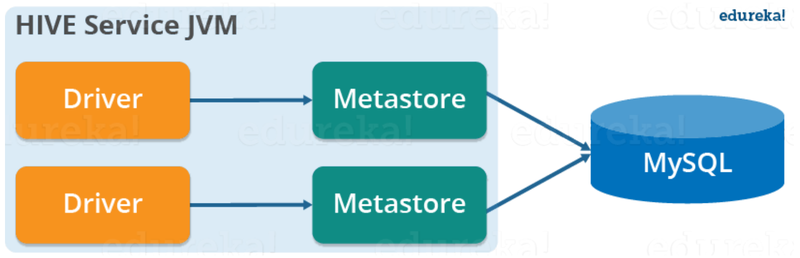
Metastore stores the meta data information using RDBMS and an open source ORM (Object Relational Model) layer called Data Nucleus which converts the object representation into relational schema and vice versa. The reason for choosing RDBMS instead of HDFS is to achieve low latency. We can implement metastore in following three configurations:

1. Embedded Metastore:



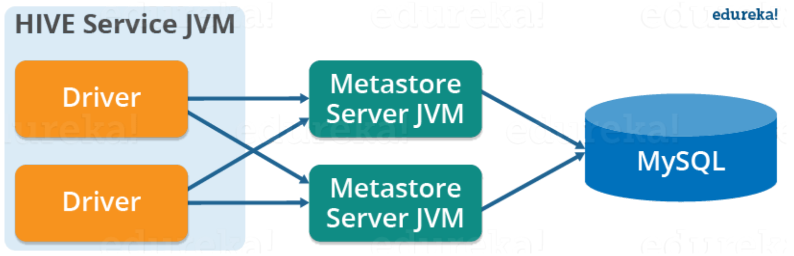
Both the metastore service and the Hive service runs in the same JVM by default using an embedded Derby Database instance where metadata is stored in the local disk. This is called embedded metastore configuration. In this case, only one user can connect to metastore database at a time. If you start a second instance of Hive driver, you will get an error. This is good for unit testing, but not for the practical solutions.

2. Local Metastore:



This configuration allows us to have multiple Hive sessions i.e. Multiple users can use the metastore database at the same time. This is achieved by using any JDBC compliant database like MySQL which runs in a separate JVM or a different machine than that of the Hive service and metastore service which are running in the same JVM as shown above. In general, the most popular choice is to implement a MySQL server as the metastore database.

3. Remote Metastore:



In the remote metastore configuration, the metastore service runs on its own separate JVM and not inthe Hive service JVM. Other processes communicate with the metastore server using Thrift Network APIs. You can have one or more metastore servers in this case to provide more availability. The main advantage of using remote metastore is you do not need to share JDBC login credential with each Hive user to access the metastore database.

Data in Hive can be categorized into three types on the granular level:

- Table
- Partition
- Bucket

Tables:

Tables in Hive are the same as the tables present in a Relational Database. You can perform filter, project, join and union operations on them. There are two types of tables in Hive:

1. Managed Table:

Command:

CREATE TABLE <table_name> (column1 data_type, column2 data_type);

LOAD DATA INPATH <HDFS_file_location> INTO table managed_table;

As the name suggests (managed table), Hive is responsible for managing the data of a managed table. In other words, what I meant by saying, "Hive manages the data", is that if you load the data from a file present in HDFS into a Hive *Managed Table* and issue a DROP command on it, the table along with its metadata will be deleted. So, the data belonging to the dropped *managed_table* no longer exist anywhere in HDFS and you can't retrieve it by any means. Basically, you are moving the data when you issue the LOAD command from the HDFS file location to the Hive warehouse directory.

Note: The default path of the warehouse directory is set to/user/hive/warehouse. The data of a Hive table resides in warehouse_directory**/**table_name (HDFS). You can also specify the path of the warehouse directory in the hive.metastore.warehouse.dir configuration parameter present in the hive-site.xml.

2. External Table:

Command:

CREATE EXTERNAL TABLE <table_name> (column1 data_type, column2 data_type) LOCATION '<table_hive_location>';

LOAD DATA INPATH '<HDFS_file_location>' INTO TABLE <table_name>;

For *external table*, Hive is not responsible for managing the data. In this case, when you issue the LOAD command, Hive moves the data into its warehouse directory. Then, Hive creates the metadata information for the external table. Now, if you issue a DROP command on the *external table*, only metadata information regarding the external table will be deleted. Therefore, you can still retrive the data of that very external table from the warehouse directory using HDFS commands.

Partitions:

Command:

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

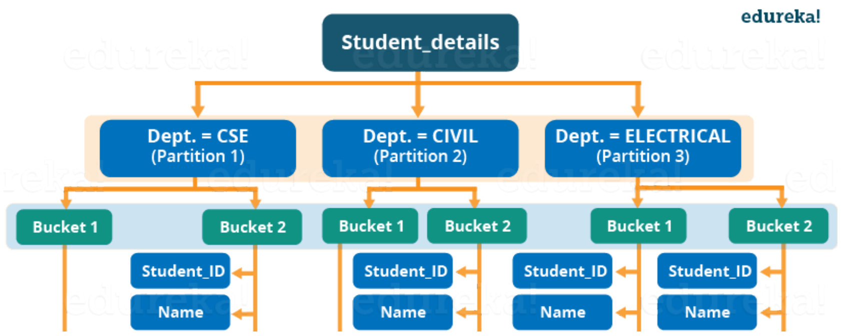
Hive organizes tables into partitions for grouping similar type of data together based on a column or partition key. Each Table can have one or more partition keys to identify a particular partition. This allows us to have a faster query on slices of the data.

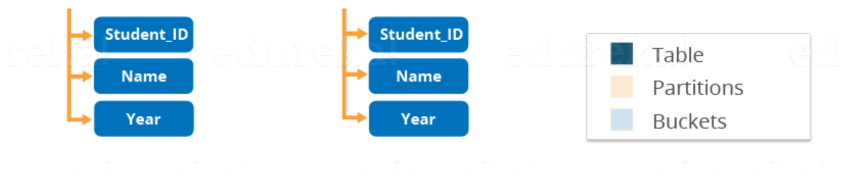
Note: Remember, the most common mistake made while creating partitions is to specify an existing column name as a partition column. While doing so, you will receive an error – "Error in semantic analysis: Column repeated in partitioning columns".

Let us understand partition by taking an example where I have a table student_details containing the student information of some engineering college like student_id, name, department, year, etc. Now, if I perform partitioning based on department column, the information of all the students belonging to a particular department will be stored together in that very partition. Physically, a partition is nothing but a sub-directory in the table directory.

Let's say we have data for three departments in our student_details table – CSE, ECE and Civil. Therefore, we will have three partitions in total for each of the departments as shown in the image below. And, for each department we will have all the data regarding that very department residing in a separate sub – directory under the Hive table directory. For example, all the student data regarding CSE departments will be stored in user/hive/warehouse/student_details/dept.=CSE. So, the queries regarding CSE students would only have to look through the data present in the CSE partition. This makes partitioning very useful as it reduces the query latency by scanning only **relevant** partitioned data instead of the whole data set. In fact, in real world implementations, you will be dealing with hundreds of TBs of data. So, imagine scanning this huge amount of data for some query where **95%** data scanned by you was un-relevant to your query.

I would suggest you to go through the blog on ***Hive commands (https://www.edureka.co/blog/hive-commands-with-examples)*** where you will find different ways of implementing partitions with an example.





Buckets:

Commands:

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;

Now, you may divide each partition or the unpartitioned table into Buckets based on the hash function of a column in the table. Actually, each bucket is just a file in the partition directory or the table directory (unpartitioned table). Therefore, if you have chosen to divide the partitions into n buckets, you will have n files in each of your partition directory. For example, you can see the above image where we have bucketed each partition into 2 buckets. So, each partition, say CSE, will have two files where each of them will be storing the CSE student's data.

How Hive distributes the rows into buckets?

Well, Hive determines the bucket number for a row by using the formula: **hash_function (bucketing_column) modulo (num_of_buckets)**. Here, hash_function depends on the column data type. For example, if you are bucketing the table on the basis of some column, let's say user_id, of INT datatype, the hash_function will be – **hash_function (user_id)= integer value of user_id**. And, suppose you have created two buckets, then Hive will determine the rows going to bucket 1 in each partition by calculating: (value of user_id) modulo (2). Therefore, in this case, rows having user_id ending with an even integer digit will reside in a same bucket corresponding to each partition. The hash_function for other data types is a bit complex to calculate and in fact, for a string it is not even humanly recognizable.

Note: If you are using Apache Hive 0.x or 1.x, you have to issue command – set hive.enforce.bucketing = true; from your Hive terminal before performing bucketing. This will allow you to have the correct number of reducer while using cluster by clause for bucketing a column. In case you have not done it, you may find the number of files that has been generated in your table directory are not equal to the number of buckets. As an alternative, you may also set the number of reducer equal to the number of buckets by using set mapred.reduce.task = num_bucket.

Why do we need buckets?

There are two main reasons for performing bucketing to a partition:

- A ***map side join (https://www.edureka.co/blog/map-side-join-vs-join/)*** requires the data belonging to a unique join key to be present in the same partition. But what about those cases where your partition key differs from join? Therefore, in these cases you can perform a map side join by bucketing the table using the join key.
- Bucketing makes the sampling process more efficient and therefore, allows us to decrease the query time.

I would like to conclude this Hive tutorial blog here. I am pretty sure after going through this Hive tutorial blog, you would have realized the simplicity of Apache Hive. Since, you guys have learned all the Hive fundamentals, it is high time to have some hands on experience with Apache Hive. So, check out the next blog in this Hive Tutorial blog series which is on Hive installation and start working on Apache Hive.

<< Previous Blog (https://www.edureka.co/blog/pig-tutorial/)

Next Blog >> (https://www.edureka.co/blog/hive-commands-with-examples)

Now that you have understood Apache Hive and its features, check out the ***Hadoop training (https://www.edureka.co/big-data-and-hadoop)*** *by Edureka, a trusted online learning company with a network of more than 250,000 satisfied learners spread across the globe. The Edureka Big Data Hadoop Certification Training course helps learners become expert in HDFS, Yarn, MapReduce, Pig, Hive, HBase, Oozie, Flume and Sqoop using real-time use cases on Retail, Social Media, Aviation, Tourism, Finance domain.*

Got a question for us? Please mention it in the comments section and we will get back to you.

About Ashish Bakshi (11 Posts (https://www.edureka.co/blog/author/ashishbedureka-co/))

Share on (https://www.facebook.com/sharer.php?u=https://www.edureka.co/blog/hive-tutorial/) (https://twitter.com/share?url=https://www.edureka.co/blog/hive-tutorial/) (https://www.linkedin.com/shareArticle?mini=true&url=https://www.edureka.co/blog/hive-tutorial/) (https://plus.google.com/share?url=https://www.edureka.co/blog/hive-tutorial/)

PREVIOUS NEXT

Related Posts

Pig Tutorial: Apache Pig Architecture & Twitter Case Study
22.7K
(https://www.edureka.co/blog/pig-tutorial/)

MapReduce Tutorial – Fundamentals of MapReduce with MapReduce Example
69K
(https://www.edureka.co/blog/mapreduce-tutorial/)

Top Hive Commands with Examples in HQL
173.2K
(https://www.edureka.co/blog/hive-commands-with-examples)

Hadoop Tutorial: All you need to know about Hadoop!
92.4K
(https://www.edureka.co/blog/hadoop-tutorial/)

Browse Categories

Big Data NoSQL (https://www.edureka.co/blog/category/big-data-nosql/) Blockchain (https://www.edureka.co/blog/category/blockchain/)

Business Intelligence (https://www.edureka.co/blog/category/business-intelligence/) Cloud Computing (https://www.edureka.co/blog/category/cloud-computing/)

Cyber Security (https://www.edureka.co/blog/category/cyber-security/) Deep Learning (https://www.edureka.co/blog/category/deep-learning/)

Finance (https://www.edureka.co/blog/category/finance/) Frameworks (https://www.edureka.co/blog/category/frameworks/)

Marketing (https://www.edureka.co/blog/category/marketing/) Mobile Development (https://www.edureka.co/blog/category/mobile-development/)

Operations (https://www.edureka.co/blog/category/operations/) Programming (https://www.edureka.co/blog/category/programming/)

Project Management (https://www.edureka.co/blog/category/project-management/) Robotic Process Automation (https://www.edureka.co/blog/category/robotic-process-automation/)

Success Story (https://www.edureka.co/blog/category/success-story/) Systems & Architecture (https://www.edureka.co/blog/category/systems-architecture/)

Systems Engineering (https://www.edureka.co/blog/category/systems-engineering/) Testing (https://www.edureka.co/blog/category/testing/)

Comments

6 Comments

5 Comments https://www.edureka.co/blog/ Rajiv Chaudhuri

Recommend 2 Share Sort by Best

Join the discussion…

Jeet Mwd • a year ago
sir can you provide me the sample project of hive
1 • Reply • Share ›

EdurekaSupport Mod → Jeet Mwd • a year ago
+Jeet Mwd, thanks for checking out our blog.
Here's a tutorial that has a project which has a step by step demo on using Hive and HBase to analyze wiki logs:

Announcing the Ridiculously Committed Mentor Award Nominate Now

. If you share your email ID, we can send you the log file.
You can also get access to projects and assignments of this kind by enrolling into our Hadoop course. Check it out here: https://www.edureka.co/big-....
Hope this helps. Cheers!
Reply • Share ›

EdurekaSupport Mod → Jeet Mwd • a year ago
+Jeet Mwd, thanks for checking out our blog.
Here's a tutorial that has a project which has a step by step demo on using Hive and HBase to analyze wiki logs:

. If you share your email ID, we can send you the log file. Hope this helps. Cheers!
Reply • Share ›

Danak • 9 months ago
How can we import already partitioned table in RDBMS to Hive?
Reply • Share ›

Bhaskar Das • a year ago
Is it possible to create an Encryption Zone in the HDFS or Hive Warehouse, when we will put or load any data or table into encryption zone location then it will get encrypted automatically??
Reply • Share ›

ALSO ON HTTPS://WWW.EDUREKA.CO/BLOG/

Talend Big Data Tutorial – A Revolution In Big Data
2 comments • 8 months ago
Avatar imen elloumi — Thank you very much for this article

Capsule Neural Networks – Set of Nested Neural Layers
1 comment • 9 months ago
Avatar Saurabh Jain — Hi Saurabh ,Would like to know about ..How to compare two images using the Neural net ?? Like i have few brain MRI ...in that case …

Real Time Big Data Applications in Various Domains
2 comments • 7 months ago
Avatar Amber Anen — Yes, I totally agree with this. Industry influencers, academicians, and other thought leaders agree that big data is making impact on business …

Top 10 Reasons Why You Should Learn Selenium
1 comment • 6 months ago
Avatar Partho — Since there is no Global Certification in Selenium what is the benchmark then.....

Subscribe Add Disqus to your siteAdd DisqusAdd Disqus' Privacy PolicyPrivacy PolicyPrivacy

Subscribe
to our newsletter

Enter your Email Address

SUBSCRIBE

Announcing the Ridiculously Committed Mentor Award Nominate Now

(https://www. tutorial/)

Pig Tutorial: Apache Pig Architecture & Twitter Case Study (https://www.edureka.co/blog/pig-tutorial/)

(https://www. tutorial/)

MapReduce Tutorial – Fundamentals of MapReduce with MapReduce Example (https://www.edureka.co/blog/mapreduce-tutorial/)

(https://www. commands-with-examples)

Top Hive Commands with Examples in HQL (https://www.edureka.co/blog/hive-commands-with-examples)

(https://www. tutorial/)

Hadoop Tutorial: All you need to know about Hadoop! (https://www.edureka.co/blog/hadoop-tutorial/)

Get Live Training & Certification!

Big Data HADOOP

LEARN MORE

(http://www.edureka.co/big-data-and-hadoop)

Big Data Analytics Courses

(/big-data-and-hadoop)

Big Data Hadoop Certification Training (/big-data-and-hadoop)

(/apache-spark-scala-training)

Apache Spark and Scala Certification Training (/apache-spark-scala-training)

(/hadoop-admin)

Hadoop Administration Certification Training (/hadoop-admin)

(/splunk)

Splunk Power User & Admin Certification Training (/splunk)

Edureka

About us (https://www.edureka.co/about-us)

Work with us

Careers (https://www.edureka.co/careers)

Become an Instructor (https://www.edureka.co/instructors)

Usefull Links

Reviews (https://www.edureka.co/reviews)

Terms & conditions (https://www.edureka.co/terms-

Follow us on

(https://www.facebook.com/edurekaIN)

Announcing the Ridiculously Committed Mentor Award Nominate Now

(https://www.edureka.co/allm (https://www.edureka.co/affiliate- Privacy policy

Contact us
(https://www.edureka.co/contact-us)
Blog
(https://www.edureka.co/blog/all/)
Community
(https://www.edureka.co/community)

program)
Hire from Edureka
(https://www.edureka.co/hire-from-edureka)

(https://www.edureka.co/privacy-policy)
Sitemap
(https://www.edureka.co/sitemap)

the GO!

(https://itunes.apple.com/in/app/edureka/id1033145415?mt=8)

(https://play.google.com/store/apps/details?id=co.edureka.app)