Apache Hive Interview Questions

1. Define the difference between Hive and HBase?

HBASE	HIVE
1. HBase is built on the top of HDFS	1. It is a data warehousing infrastructure
2. HBase operations run in a real-time on its database	2. Hive queries are executed as MapReduce jobs internally
3. Provides low latency to single rows from huge datasets	3. Provides high latency for huge datasets
4. Provides random access to data	4. Provides random access to data

2. What kind of applications is supported by Apache Hive?

Hive supports all those client applications that are written in:

Java

PHP

Python

C++

Ruby

by exposing its Thrift server.

3. Where does the data of a Hive table gets stored?

By default, the Hive table is stored in an HDFS directory – /user/hive/warehouse. One can change it by specifying the desired directory in hive.metastore.warehouse.dir configuration parameter present in the hive-site.xml.

4. What is a metastore in Hive?

Metastore in Hive 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.

5. Why Hive does not store metadata information in HDFS?

Hive stores metadata information in the metastore using RDBMS instead of HDFS. The reason for choosing RDBMS is to achieve low latency as HDFS read/write operations are time consuming processes.

6. What is the difference between local and remote metastore?

Local Metastore:

In local metastore configuration, the metastore service runs in the same JVM in which the Hive service is running and connects to a database running in a separate JVM, either on the same machine or on a remote machine.

Remote Metastore:

In the remote metastore configuration, the metastore service runs on its own separate JVM and not in the 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.

7. What is the default database provided by Apache Hive for metastore?

By default, Hive provides an embedded Derby database instance backed by the local disk for the metastore. This is called the embedded metastore configuration.

8. Scenario:

Suppose I have installed Apache Hive on top of my Hadoop cluster using default metastore configuration. Then, what will happen if we have multiple clients trying to access Hive at the same time?

The default metastore configuration allows only one Hive session to be opened at a time for accessing the metastore. Therefore, if multiple clients try to access the metastore at the same time, they will get an error. One has to use a standalone metastore, i.e. Local or remote metastore configuration in Apache Hive for allowing access to multiple clients concurrently.

Following are the steps to configure MySQL database as the local metastore in Apache Hive:

One should make the following changes in hive-site.xml:

javax.jdo.option.ConnectionURL property should be set to jdbc:mysql://host/dbname?createDatabaseIfNotExist=true.

javax.jdo.option.ConnectionDriverName property should be set to com.mysql.jdbc.Driver.

One should also set the username and password as:

javax.jdo.option.ConnectionUserName is set to desired username.

javax.jdo.option.ConnectionPassword is set to the desired password.

The JDBC driver JAR file for MySQL must be on the Hive's classpath, i.e. The jar file should be copied into the Hive's lib directory.

Now, after restarting the Hive shell, it will automatically connect to the MySQL database which is running as a standalone metastore.

9. What is the difference between external table and managed table?

Here is the key difference between an external table and managed table:

In case of managed table, If one drops a managed table, the metadata information along with the table data is deleted from the Hive warehouse directory.

On the contrary, in case of an external table, Hive just deletes the metadata information regarding the table and leaves the table data present in HDFS untouched.

10. Is it possible to change the default location of a managed table?

Yes, it is possible to change the default location of a managed table. It can be achieved by using the clause – LOCATION '<hdfs_path>'.

11. When should we use SORT BY instead of ORDER BY?

We should use SORT BY instead of ORDER BY when we have to sort huge datasets because SORT BY clause sorts the data using multiple reducers whereas ORDER BY sorts all of the data together using a single reducer. Therefore, using ORDER BY against a large number of inputs will take a lot of time to execute.

12. What is a partition in Hive?

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. Physically, a partition is nothing but a sub-directory in the table directory.

13. Why do we perform partitioning in Hive?

Partitioning provides granularity in a Hive table and therefore, reduces the query latency by scanning only relevant partitioned data instead of the whole data set.

For example, we can partition a transaction log of an e – commerce website based on month like Jan, February, etc. So, any analytics regarding a particular month, say Jan, will have to scan the Jan partition (sub – directory) only instead of the whole table data.

14. What is dynamic partitioning and when is it used?

In dynamic partitioning values for partition columns are known in the runtime, i.e. It is known during loading of the data into a Hive table.

One may use dynamic partition in following two cases:

Loading data from an existing non-partitioned table to improve the sampling and therefore, decrease the query latency.

When one does not know all the values of the partitions before hand and therefore, finding these partition values manually from a huge data sets is a tedious task.

15. Scenario:

Suppose, I create a table that contains details of all the transactions done by the customers of year 2016: CREATE TABLE transaction_details (cust_id INT, amount FLOAT, month STRING, country STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’ ;

Now, after inserting 50,000 tuples in this table, I want to know the total revenue generated for each month. But, Hive is taking too much time in processing this query. How will you solve this problem and list the steps that I will be taking in order to do so?

We can solve this problem of query latency by partitioning the table according to each month. So, for each month we will be scanning only the partitioned data instead of whole data sets.

As we know, we can’t partition an existing non-partitioned table directly. So, we will be taking following steps to solve the very problem:

1. Create a partitioned table, say partitioned_transaction:

CREATE TABLE partitioned_transaction (cust_id INT, amount FLOAT, country STRING) PARTITIONED BY (month STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’ ;

2. Enable dynamic partitioning in Hive:

SET hive.exec.dynamic.partition = true;

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

3. Transfer the data from the non – partitioned table into the newly created partitioned table:

INSERT OVERWRITE TABLE partitioned_transaction PARTITION (month) SELECT cust_id, amount, country, month FROM transaction_details;

Now, we can perform the query using each partition and therefore, decrease the query time.

16. How can you add a new partition for the month December in the above partitioned table?

For adding a new partition in the above table partitioned_transaction, we will issue the command give below:

ALTER TABLE partitioned_transaction ADD PARTITION (month='Dec') LOCATION '/partitioned_transaction';

17. What is the default maximum dynamic partition that can be created by a mapper/reducer? How can you change it?

By default the number of maximum partition that can be created by a mapper or reducer is set to 100. One can change it by issuing the following command:

SET hive.exec.max.dynamic.partitions.pernode = <value>

Note: You can set the total number of dynamic partitions that can be created by one statement by using: SET hive.exec.max.dynamic.partitions = <value>

18. Scenario:

I am inserting data into a table based on partitions dynamically. But, I received an error – FAILED ERROR IN SEMANTIC ANALYSIS: Dynamic partition strict mode requires at least one static partition column. How will you remove this error?

To remove this error one has to execute following commands:

SET hive.exec.dynamic.partition = true;

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

Things to Remember:

By default, hive.exec.dynamic.partition configuration property is set to False in case you are using Hive whose version is prior to 0.9.0.

hive.exec.dynamic.partition.mode is set to strict by default. Only in non – strict mode Hive allows all partitions to be dynamic.

19. Why do we need buckets?

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

A map side 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 that of join key? 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.

20. How Hive distributes the rows into buckets?

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 integer data type, the hash_function will be:

hash_function (int_type_column)= value of int_type_column

21. What will happen in case you have not issued the command: 'SET hive.enforce.bucketing=true;' before bucketing a table in Hive in Apache Hive 0.x or 1.x?

The command: 'SET hive.enforce.bucketing=true;' allows one to have the correct number of reducer while using 'CLUSTER BY' clause for bucketing a column. In case it's not done, one may find the number of files that will be generated in the table directory to be not equal to the number of buckets. As an alternative, one may also set the number of reducer equal to the number of buckets by using set mapred.reduce.task = num_bucket.

22. What is indexing and why do we need it?

One of the Hive query optimization methods is Hive index. Hive index is used to speed up the access of a column or set of columns in a Hive database because with the use of index the database system does not need to read all rows in the table to find the data that one has selected.

23. Scenario:

Suppose, I have a CSV file – 'sample.csv' present in '/temp' directory with the following entries:

id first_name last_name email gender ip_address

1 Hugh Jackman hughjackman@cam.ac.uk Male 136.90.241.52

2 David Lawrence dlawrence1@gmail.com Male 101.177.15.130

3 Andy Hall andyhall2@yahoo.com Female 114.123.153.64

4 Samuel Jackson samjackson231@sun.com Male 89.60.227.31

5 Emily Rose rose.emily4@surveymonkey.com Female 119.92.21.19

How will you consume this CSV file into the Hive warehouse using built SerDe?

SerDe stands for serializer/deserializer. A SerDe allows us to convert the unstructured bytes into a record that we can process using Hive. SerDes are implemented using Java. Hive comes with several built-in SerDes and many other third-party SerDes are also available.

Hive provides a specific SerDe for working with CSV files. We can use this SerDe for the sample.csv by issuing following commands:

CREATE EXTERNAL TABLE sample (id int, first_name string, last_name string, email string, gender string, ip_address string)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

STORED AS TEXTFILE LOCATION '/temp';

Now, we can perform any query on the table 'sample':

SELECT first_name FROM sample WHERE gender = 'male';

24. Scenario:

Suppose, I have a lot of small CSV files present in /input directory in HDFS and I want to create a single Hive table corresponding to these files. The data in these files are in the format: {id, name, e-mail, country}. Now, as we know, Hadoop performance degrades when we use lots of small files.

So, how will you solve this problem where we want to create a single Hive table for lots of small files without degrading the performance of the system?

One can use the SequenceFile format which will group these small files together to form a single sequence file. The steps that will be followed in doing so are as follows:

Create a temporary table:

CREATE TABLE temp_table (id INT, name STRING, e-mail STRING, country STRING)

ROW FORMAT FIELDS DELIMITED TERMINATED BY ',' STORED AS TEXTFILE;

Load the data into temp_table:

LOAD DATA INPATH '/input' INTO TABLE temp_table;

Create a table that will store data in SequenceFile format:

CREATE TABLE sample_seqfile (id INT, name STRING, e-mail STRING, country STRING)

ROW FORMAT FIELDS DELIMITED TERMINATED BY ‘,’ STORED AS SEQUENCEFILE;

Transfer the data from the temporary table into the sample_seqfile table:

INSERT OVERWRITE TABLE sample SELECT * FROM temp_table;

Hence, a single SequenceFile is generated which contains the data present in all of the input files and therefore, the problem of having lots of small files is finally eliminated.