**Apache Hive Interview Questions**

Here is the comprehensive list of the most frequently asked Apache Hive Interview Questions that have been framed after deep research and discussion with the industry experts.

**1. Define the difference between Hive and HBase?**

|  |  |
| --- | --- |
| **Hive vs 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 rather | 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***](https://www.edureka.co/blog/hive-tutorial/) 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?createDataba  
    *seIfNotExist=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.

***Note:*** I would suggest you to go through the blog on [***Hive Tutorial***](https://www.edureka.co/blog/hive-tutorial/?#data_model)to learn more about Managed Table and External Table in Hive.

**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.

**[Big Data Hadoop Certification Training](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)**

* *[Instructor-led Sessions](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)*
* *[Real-life Case Studies](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)*
* *[Assessments](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)*
* *[Lifetime Access](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)*

[Explore Curriculum](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)

**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’;*

***Note:*** I suggest you to go through the dedicated blog on [***Hive Commands***](https://www.edureka.co/blog/hive-commands-with-examples) where all the commands present in Apache Hive have been explained with an example.

**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:***

Next

* 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***](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 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)*

**[Big Data Hadoop Certification Training](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)**

[Weekday / Weekend BatchesSee Batch Details](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)

*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.

**Hbase Interview Question Part 1**

**1. What are the key components of HBase?**

The key components of HBase are Zookeeper, RegionServer and HBase Master.

|  |  |
| --- | --- |
| **Key components of HBase** | |
| **Component** | **Description** |
| **Region Server** | A table can be divided into several regions. A group of regions is served to the clients by a Region Server |
| **HMaster** | It coordinates and manages the Region Servers (similar as NameNode manages DataNodes in HDFS). |
| **ZooKeeper** | Zookeeper acts like as a coordinator inside HBase distributed environment. It helps in maintaining server state inside the cluster by communicating through sessions. |

**2. When would you use HBase?**

* HBase is used in cases where we need random read and write operations and it can perform a number of operations per second on a large data sets.
* HBase gives strong data consistency.
* It can handle very large tables with billions of rows and millions of columns on top of commodity hardware cluster.

**3. What is the use of get() method?**

get() method is used to read the data from the table.

**4. Define the difference between Hive and HBase?**

Apache Hive is a data warehousing infrastructure built on top of Hadoop. It helps in querying data stored in HDFS for analysis using Hive Query Language (HQL), which is a SQL-like language, that gets translated into MapReduce jobs. Hive performs batch processing on Hadoop.

Apache HBase is NoSQL key/value store which runs on top of HDFS. Unlike Hive, HBase operations run in real-time on its database rather than MapReduce jobs. HBase partitions the tables, and the tables are further splitted into column families.

Hive and HBase are two different Hadoop based technologies – Hive is an SQL-like engine that runs MapReduce jobs, and HBase is a NoSQL key/value database of Hadoop. We can use them together. Hive can be used for analytical queries while HBase for real-time querying. Data can even be read and written from HBase to Hive and vice-versa.

**5. Explain the data model of HBase.**

HBase comprises of:

* Set of tables.
* Each table consists of column families and rows.
* Row key acts as a Primary key in HBase.
* Any access to HBase tables uses this Primary Key.
* Each column qualifier present in HBase denotes attributes corresponding to the object which resides in the cell.

**6. Define column families?**

Column Family is a collection of columns, whereas row is a collection of column families.

**7. Define standalone mode in HBase?**

It is a default mode of HBase. In standalone mode, HBase does not use HDFS—it uses the local filesystem instead—and it runs all HBase daemons and a local ZooKeeper in the same JVM process.

**8. What is decorating Filters?**

It is useful to modify, or extend, the behavior of a filter to gain additional control over the returned data. These types of filters are known as decorating filter. It includes SkipFilter and WhileMatchFilter.

**9. What is RegionServer?**

A table can be divided into several regions. A group of regions is served to the clients by a Region Server.

**10. What are the data manipulation commands of HBase?**

Data Manipulation commands of HBase are:

* **put** – Puts a cell value at a specified column in a specified row in a particular table.
* **get** – Fetches the contents of a row or a cell.
* **delete** – Deletes a cell value in a table.
* **deleteall** – Deletes all the cells in a given row.
* **scan** – Scans and returns the table data.
* **count** – Counts and returns the number of rows in a table.
* **truncate** – Disables, drops, and recreates a specified table.

**11. Which code is used to open a connection in HBase?**

Following code is used to open a HBase connection, here *users* is my HBase table:

|  |  |
| --- | --- |
| 1  2 | Configuration myConf = HBaseConfiguration.create();  HTable table = new HTable(myConf, “users”); |

**12. What is the use of truncate command?**

It is used to disable, drop and recreate the specified tables.

[[](https://www.edureka.co/big-data-hadoop-training-certification)](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)

**[Big Data Hadoop Certification Training](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)**

* *[Instructor-led Sessions](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)*
* *[Real-life Case Studies](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)*
* *[Assessments](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)*
* *[Lifetime Access](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)*

[Explore Curriculum](https://www.edureka.co/big-data-hadoop-training-certification" \t "_blank)

***♣ Tip:****To delete table first disable it, then delete it.*

**13. What happens when you issue a delete command in HBase?**

Once you issue a delete command in HBase for cell, column or column family, it is not deleted instantly. A tombstone marker in inserted. Tombstone is a specified data, which is stored along with standard data. This tombstone makes hides all the deleted data.

The actual data is deleted at the time of major compaction. In Major compaction, HBase merges and recommits the smaller HFiles of a region to a new HFile. In this process, the same column families are placed together in the new HFile. It drops deleted and expired cell in this process. All the results from scan and get filters the deleted cells.

**14. What are different tombstone markers in HBase?**

There are three types of tombstone markers in HBase:

* Version Marker: Marks only one version of a column for deletion.
* Column Marker: Marks the whole column (i.e. all version) for deletion.
* Family Marker: Marks the whole column family (i.e. all the columns in the column family) for deletion

**15. HBase blocksize is configured on which level?**

The blocksize is configured per column family and the default value is 64 KB. This value can be changed as per requirements.

**16. Which command is used to run HBase Shell?**

*./bin/hbase* *shell* command is used to run the HBase shell. Execute this command in HBase directory.

**17. Which command is used to show the current HBase user?**

whoami command is used to show HBase user.

**18. What is the full form of MSLAB?**

MSLAB stands for Memstore-Local Allocation Buffer. Whenever a request thread needs to insert data into a MemStore, it doesn’t allocates the space for that data from the heap at large, but rather allocates memory arena dedicated to the target region.

**19. Define LZO?**

Lempel-Ziv-Oberhumer (LZO) is a lossless data compression algorithm that focuses on decompression speed.

**20. What is HBase Fsck?**

HBase comes with a tool called hbck which is implemented by the HBaseFsck class. HBaseFsck (hbck) is a tool for checking for region consistency and table integrity problems and repairing a corrupted HBase. It works in two basic modes – a read-only inconsistency identifying mode and a multi-phase read-write repair mode.

**21. What is REST?**

Rest stands for Representational State Transfer which defines the semantics so that the protocol can be used in a generic way to address remote resources. It also provides support for different message formats, offering many choices for a client application to communicate with the server.

**22. What is Thrift?**

Apache Thrift is written in C++, but provides schema compilers for many programming languages, including Java, C++, Perl, PHP, Python, Ruby, and more.

**23. What is Nagios?**

Nagios is a very commonly used support tool for gaining qualitative data regarding cluster status. It polls current metrics on a regular basis and compares them with given thresholds.

**24. What is the use of ZooKeeper?**

The ZooKeeper is used to maintain the configuration information and communication between region servers and clients. It also provides distributed synchronization. It helps in maintaining server state inside the cluster by communicating through sessions.

Every Region Server along with HMaster Server sends continuous heartbeat at regular interval to Zookeeper and it checks which server is alive and available. It also provides server failure notifications so that, recovery measures can be executed.

**25. Define catalog tables in HBase?**

Catalog tables are used to maintain the metadata information.

**26. Define compaction in HBase?**

HBase combines HFiles to reduce the storage and reduce the number of disk seeks needed for a read. This process is called compaction. Compaction chooses some HFiles from a region and combines them. There are two types of compactions.

* **Minor Compaction**: HBase automatically picks smaller HFiles and recommits them to bigger HFiles.
* **Major Compaction**: In Major compaction, HBase merges and recommits the smaller HFiles of a region to a new HFile.

**27. What is the use of HColumnDescriptor class?**

HColumnDescriptor stores the information about a column family like compression settings, number of versions etc. It is used as input when creating a table or adding a column.

**28. Which filter accepts the pagesize as the parameter in hBase?**

PageFilter accepts the pagesize as the parameter. Implementation of Filter interface that limits results to a specific page size. It terminates scanning once the number of filter-passed the rows greater than the given page size.

Syntax: PageFilter (<page\_size>)

**29. How will you design or modify schema in HBase programmatically?**

HBase schemas can be created or updated using the Apache HBase Shell or by using Admin in the Java API.

Creating table schema:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | Configuration config = HBaseConfiguration.create();  HBaseAdmin admin = new HBaseAdmin(conf); // execute command through admin</span></pre>    // Instantiating table descriptor class  HTableDescriptor t1 = new HTableDescriptor(TableName.valueOf("employee"));    // Adding column families to t1  t1.addFamily(new HColumnDescriptor("professional"));  t1.addFamily(new HColumnDescriptor("personal"));    // Create the table through admin  admin.createTable(t1); |

*♣ Tip: Tables must be disabled when making ColumnFamily modifications.*

For modification:

|  |  |
| --- | --- |
| 1  2  3  4 | String table = “myTable”;  admin.disableTable(table);  admin.modifyColumn(table, cf2); // modifying existing ColumnFamily  admin.enableTable(table); |

**30.What are the filters are available in Apache HBase?**

The filters that are supported by HBase are:

* **ColumnPrefixFilter**: takes a single argument, a column prefix. It returns only those key-values present in a column that starts with the specified column prefix.
* **TimestampsFilter**: takes a list of timestamps. It returns those key-values whose timestamps match any of the specified timestamps.
* **PageFilter**: takes one argument, a page size. It returns page size, number of rows from the table.
* **MultipleColumnPrefixFilter**: takes a list of column prefixes. It returns key-values that are present in a column that starts with any of the specified column prefixes.
* **ColumnPaginationFilter**: takes two arguments, a limit and an offset. It returns limit number of columns after offset number of columns. It does this for all the rows.
* **SingleColumnValueFilter**: takes a column family, a qualifier, a comparison operator and a comparator. If the specified column is not found, all the columns of that row will be emitted. If the column is found and the comparison with the comparator returns true, all the columns of the row will be emitted.
* **RowFilter**: takes a comparison operator and a comparator. It compares each row key with the comparator using the comparison operator and if the comparison returns true, it returns all the key-values in that row.
* **QualifierFilter**: takes a comparison operator and a comparator. It compares each qualifier name with the comparator using the comparison operator and if the comparison returns true, it returns all the key-values in that column.
* **ColumnRangeFilter**: takes either minColumn, maxColumn, or both. Returns only those keys with columns that are between minColumn and maxColumn. It also takes two boolean variables to indicate whether to include the minColumn and maxColumn or not. If you don’t want to set the minColumn or the maxColumn, you can pass in an empty argument.
* **ValueFilter**: takes a comparison operator and a comparator. It compares each value with the comparator using the compare operator and if the comparison returns true, it returns that key-value.
* **PrefixFilter**: takes a single argument, a prefix of a row key. It returns only those key-values present in a row that start with the specified row prefix.
* **SingleColumnValueExcludeFilter**: takes the same arguments and behaves same as SingleColumnValueFilter. However, if the column is found and the condition passes, all the columns of the row will be omitted except for the tested column value.
* **ColumnCountGetFilter**: takes one argument, a limit. It returns the first limit number of columns in the table.
* **InclusiveStopFilter**: takes one argument, a row key on which to stop scanning. It returns all key-values present in rows up to and including the specified row.
* **DependentColumnFilter**: takes two arguments required arguments, a family and a qualifier. It tries to locate this column in each row and returns all key-values in that row that have the same timestamp.
* **FirstKeyOnlyFilter**: takes no arguments. Returns the key portion of the first key-value pair.
* **KeyOnlyFilter**: takes no arguments. Returns the key portion of each key-value pair.
* **FamilyFilter**: takes a comparison operator and comparator. It compares each family name with the comparator using the comparison operator and if the comparison returns true, it returns all the key-values in that family.
* **CustomFilter**: You can create a custom filter by implementing the Filter class.

**31. How do we back up a HBase cluster?**

There are two broad strategies for performing HBase backups: backing up with a full cluster shutdown, and backing up on a live cluster. Each approach has benefits and limitation.

**Full Shutdown Backup**

Some environments can tolerate a periodic full shutdown of their HBase cluster, for example, if it is being used as a back-end process and not serving front-end webpages.

* **Stop HBase**: Stop the HBase services first.
* **Distcp**: Distcp could be used to either copy the contents of the HBase directory in HDFS to either the same cluster in another directory, or to a different cluster.
* **Restore**: The backup of the HBase directory from HDFS is copied onto the ‘real’ HBase directory via distcp. The act of copying these files, creates new HDFS metadata, which is why a restore of the NameNode edits from the time of the HBase backup isn’t required for this kind of restore, because it’s a restore (via distcp) of a specific HDFS directory (i.e., the HBase part) not the entire HDFS file-system.

**Live Cluster Backup**

The environments which cannot handle downtime uses Live Cluster Backup.

* **CopyTable**: Copy table utility could either be used to copy data from one table to another on the same cluster, or to copy data to another table on another cluster.
* **Export**: Export approach dumps the content of a table to HDFS on the same cluster.

**32. How HBase Handles the write failure?**

Failures are common in large distributed systems, and HBase is no exception.

If the server hosting a MemStore that has not yet been flushed crashes. The data that was in memory, but not yet persisted are lost. HBase safeguards against that by writing to the WAL before the write completes. Every server that’s part of the.

HBase cluster keeps a WAL to record changes as they happen. The WAL is a file on the underlying file system. A write isn’t considered successful until the new WAL entry is successfully written. This guarantee makes HBase as durable as the file system backing it. Most of the time, HBase is backed by the Hadoop Distributed Filesystem (HDFS). If HBase goes down, the data that were not yet flushed from the MemStore to the HFile can be recovered by replaying the WAL.

**33. While reading data from HBase, from which three places data will be reconciled before returning the value?**

The read process will go through the following process sequentially:

* For reading the data, the scanner first looks for the Row cell in Block cache. Here all the recently read key value pairs are stored.
* If Scanner fails to find the required result, it moves to the MemStore, as we know this is the write cache memory. There, it searches for the most recently written files, which has not been dumped yet in HFile.
* At last, it will use bloom filters and block cache to load the data from the HFile.

**34. Can you explain data versioning?**

In addition to being a schema-less database, HBase is also versioned.

Every time you perform an operation on a cell, HBase implicitly stores a new version. Creating, modifying and deleting a cell are all treated identically, they are all new versions. When a cell exceeds the maximum number of versions, the extra records are dropped during the major compaction.

Instead of deleting an entire cell, you can operate on a specific version within that cell. Values within a cell are versioned and it is identified the timestamp. If a version is not mentioned, then the current timestamp is used to retrieve the version. The default number of cell version is three.

## ****35. What is a Bloom filter and how does it help in searching rows?****

HBase supports Bloom Filter to improve the overall throughput of the cluster. A HBase Bloom Filter  is a space efficient mechanism to test whether a HFile contains a specific row or row-col cell.

Without Bloom Filter, the only way to decide if a row key is present in a HFile  is to check the HFile’s block index, which stores the start row key of each block in the HFile. There are many rows drops between the two start keys. So, HBase has to load the block and scan the block’s keys to figure out if that row key actually exists.

**Hive Interview Question and Answer**

### ****2. How to skip header rows from a table in Hive?****

Imagine that header records in a table are as follows:

System=…

Version=…

Sub-version=…

Suppose, we do not want to include the above three lines of headers in our Hive query. To skip the header lines from our table in Hive, we will set a table property.

CREATE EXTERNAL TABLE employee (

name STRING,

job STRING,

dob STRING,

id INT,

salary INT)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘ ‘ STORED AS TEXTFILE

LOCATION ‘/user/data’

TBLPROPERTIES("skip.header.line.count"="2”);

### ****3. What is a Hive variable? What do we use it for?****

Hive variables are basically created in the Hive environment that is referenced by Hive scripting languages. They allow to pass some values to a Hive query when the query starts executing. They use the source command.

### ****4. Explain the process to access subdirectories recursively in Hive queries.****

By using the below commands, we can access subdirectories recursively in Hive:

hive> Set mapred.input.dir.recursive=true;

hive> Set hive.mapred.supports.subdirectories=true;

Hive tables can be pointed to the higher level directory, and this is suitable for the directory structure like:

/data/country/state/city/

### ****5. Can we change the settings within a Hive session? If yes, how?****

Yes, we can change the settings within a Hive session using the SET command. It helps change the Hive job settings for an exact query. For example, the following command shows that buckets are occupied according to the table definition:

hive> SET hive.enforce.bucketing=true;

We can see the current value of any property by using SET with the property name. SET will list all the properties with their values set by Hive.

hive> SET hive.enforce.bucketing;

hive.enforce.bucketing=true

This list will not include the defaults of Hadoop. So, we should use the below code:

SET -v

It will list all the properties including the Hadoop defaults in the system.

***Interested in learning Hive? Well, we have a comprehensive master’s***[***Big Data Hadoop Course***](https://intellipaat.com/big-data-hadoop-training/)***to give you an idea of all concepts in Hive!***

### ****6. Is it possible to add 100 nodes when we already have 100 nodes in Hive? If yes, how?****

**Yes, we can add the nodes by following the below steps:**

**Step 1**: Take a new system; create a new username and password  
**Step 2**: Install SSH and with the master node setup SSH connections  
**Step 3**: Add ssh public\_rsa id key to the authorized keys file  
**Step 4**: Add the new DataNode hostname, IP address, and other details in /etc/hosts slaves file:

192.168.1.102 slave3.in slave3

**Step 5**: Start the DataNode on a new node  
**Step 6**: Login to the new node like suhadoop or:

ssh -X hadoop@192.168.1.103

**Step 7**: Start HDFS of the newly added slave node by using the following command:

./bin/hadoop-daemon.sh start data node

**Step 8**: Check the output of the jps command on the new node

***Go through this***[***Hadoop Training in London***](https://intellipaat.com/big-data-hadoop-training-london/)***to get a clear understanding of Hadoop!***

### ****7. Explain the concatenation function in Hive with an example.****

The concatenate function will join the input strings. We can specify  
‘n’ number of strings separated by a comma.

**Example**:

CONCAT ('Intellipaat','-','is','-','a','-','eLearning',’-’,’provider’);

**Output**:

Intellipaat-is-a-eLearning-provider

Every time, we set the limits of the strings by ‘-‘. If it is common for every string, then Hive provides another command:

CONCAT\_WS

In this case, we have to specify the set limits of the operator first as follows:

CONCAT\_WS ('-',’Intellipaat’,’is’,’a’,’eLearning’,‘provider’);

**Output**:

Intellipaat-is-a-eLearning-provider

**8. Explain the Trim and Reverse functions in Hive with examples.**

The trim function will delete the spaces associated with a string.

**Example**:

TRIM(‘ INTELLIPAAT ‘);

**Output**:

INTELLIPAAT

To remove the leading space:

LTRIM(‘ INTELLIPAAT’);

To remove the trailing space:

RTRIM(‘INTELLIPAAT ‘);

In the reverse function, characters are reversed in the string.  
**Example**:

REVERSE(‘INTELLIPAAT’);

**Output**:

TAAPILLETNI

**9. How to change the column data type in Hive? Explain RLIKE in Hive.**

We can change the column data type by using ALTER and CHANGE as follows:

ALTER TABLE table\_name CHANGE column\_namecolumn\_namenew\_datatype;

For example, if we want to change the data type of the salary column from integer to bigint in the employee table, we can use the following:

ALTER TABLE employee CHANGE salary salary BIGINT;

**RLIKE**: Its full form is Right-Like and it is a special function in Hive. It helps examine two substrings, i.e., if the substring of A matches with B, then it evaluates to true.

**Example**:

‘Intellipaat’ RLIKE ‘tell’ ◊ True

‘Intellipaat’ RLIKE ‘^I.\*’ ◊ True (this is a regular expression)

***Learn more about***[***Apache Hive***](https://intellipaat.com/blog/what-is-apache-hive/)***from this detailed blog post now!***

**10. What are the components used in Hive Query Processor?**

Following are the components of a Hive Query Processor:

* Parse and Semantic Analysis (ql/parse)
* Metadata Layer (ql/metadata)
* Type Interfaces (ql/typeinfo)
* Sessions (ql/session)
* Map/Reduce Execution Engine (ql/exec)
* Plan Components (ql/plan)
* Hive Function Framework (ql/udf)
* Tools (ql/tools)
* Optimizer (ql/optimizer)

**11. What are Buckets in Hive?**

Buckets in Hive are used in segregating Hive table data into multiple files or directories. They are used for efficient querying.

**12. What kind of data warehouse application is suitable for Hive? What are the types of tables in Hive?**

Hive is not considered a full database. The design rules and regulations of Hadoop and HDFS have put restrictions on what Hive can do. However, Hive is most suitable for data warehouse applications because it:

* Analyzes relatively static data
* Has less responsive time
* Does not make rapid changes in data

Although Hive doesn’t provide fundamental features required for Online Transaction Processing (OLTP), it is suitable for data warehouse applications in large datasets. There are two types of tables in Hive:

* Managed tables
* External tables

***Get a better understanding of Hive by going through this***[***Hive Tutorial***](https://intellipaat.com/blog/tutorial/hadoop-tutorial/apache-hive/)***now!***

**13. What is the definition of Hive? What is the present version of Hive? Explain ACID transactions in Hive.**

Hive is an open-source data warehouse system. We can use Hive for analyzing and querying large datasets. It’s similar to SQL. The present version of Hive is 0.13.1. Hive supports ACID (Atomicity, Consistency, Isolation, and Durability) transactions. ACID transactions are provided at row levels. Following are the options Hive uses to support ACID transactions:

* Insert
* Delete
* Update

***Want to learn more about Hive? Go through this insightful blog ‘***[***What is Hive?***](https://intellipaat.com/blog/what-is-apache-hive/)***‘***

**14. What is the maximum size of a string data type supported by Hive? Explain how Hive supports binary formats.**

The maximum size of a string data type supported by Hive is 2 GB. Hive supports the text file format by default, and it also supports the binary format sequence files, ORC files, Avro data files, and Parquet files.

* **Sequence file**: It is a splittable, compressible, and row-oriented file with a general binary format.
* **ORC file**: Optimized row columnar (ORC) format file is a record-columnar and column-oriented storage file. It divides the table in row split. Each split stores the value of the first row in the first column and follows subsequently.
* **Avro data file**: It is the same as a sequence file that is splittable, compressible, and row-oriented but without the support of schema evolution and multilingual binding.
* **Parquet file:** In Parquet format, along with storing rows of data adjacent to one another, we can also store column values adjacent to each other such that both horizontally and vertically datasets are partitioned.

***Learn more about Hadoop from this***[***Hadoop Training in New York***](https://intellipaat.com/big-data-hadoop-training-new-york/)***to get ahead in your career!***

**15. What is the precedence order of Hive configuration?**

We are using a precedence hierarchy for setting properties:

1. The SET command in Hive
2. The command-line –hiveconf option
3. Hive-site.XML
4. Hive-default.xml
5. Hadoop-site.xml
6. Hadoop-default.xml

**16. If you run a select \* query in Hive, why doesn't it run MapReduce?**

The hive.fetch.task.conversion property of Hive lowers the latency of MapReduce overhead, and in effect when executing queries such as SELECT, FILTER, LIMIT, etc. it skips the MapReduce function.

***If you have any doubts or queries related to Hive, get them clarified from Hadoop experts on our***[***Hive Community***](https://intellipaat.com/community/search?q=hive)***!***

**Advanced Interview Questions**

**17. How can we improve the performance with ORC format tables in Hive?**

We can store Hive data in a highly efficient manner in an Optimized Row Columnar (ORC) file format. It can simplify many Hive file format limitations. We can improve the performance by using ORC files while reading, writing, and processing data.

Set hive.compute.query.using.stats-true;

Set hive.stats.dbclass-fs;

CREATE TABLE orc\_table (

idint,

name string)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ‘\:’

LINES TERMINATED BY ‘\n’

STORES AS ORC;

***Need a reason to learn Apache Hadoop and Hive? Well, go through this blog post to find out***[***Why Hadoop is the new black?***](https://intellipaat.com/blog/hadoop-is-the-new-black/)

**18. Explain the functionality of ObjectInspector.**

ObjectInspector helps analyze the internal structure of a row object and the individual structure of columns in Hive. It also provides a uniform way to access complex objects that can be stored in multiple formats in the memory.

* An instance of Java class
* A standard Java object
* A lazily initialized object

ObjectInspector tells the structure of the object and also the ways to access the internal fields inside the object.

**19. Whenever we run a Hive query, a new metastore\_db is created. Why?**

A local metastore is created when we run Hive in an embedded mode. Before creating, it checks whether the metastore exists or not, and this metastore property is defined in the configuration file, hive-site.xml. The property is:

javax.jdo.option.ConnectionURL

with the default value:

jdbc:derby:;databaseName=metastore\_db;create=true

Therefore, we have to change the behavior of the location to an absolute path so that from that location the metastore can be used.

***Interested in learning Hadoop? Check out the***[***Hadoop Training in Sydney***](https://intellipaat.com/big-data-hadoop-training-sydney/)***!***

**20. Differentiate between Hive and HBase.**

|  |  |
| --- | --- |
| **Hive** | **HBase** |
| Enables most SQL queries | Does not allow SQL queries |
| Operations do not run in real time | Operations run in real time |
| A data warehouse framework | A NoSQL database |
| Runs on top of MapReduce | Runs on top of HDFS |

**21. How can we access the subdirectories recursively?**

By using the below commands, we can access subdirectories recursively in Hive:

hive> Set mapred.input.dir.recursive=true;

hive> Set hive.mapred.supports.subdirectories=true;

Hive tables can be pointed to the higher level directory, and this is suitable for the directory structure:

/data/country/state/city/

**22. What are the uses of Hive Explode?**

Hadoop Developers consider an array as their input and convert it into a separate table row. To convert complicated data types into desired table formats, Hive uses Explode.

***Learn end-to-end Hadoop concepts through the***[***Hadoop Course in Hyderabad***](https://intellipaat.com/big-data-hadoop-training-hyderabad/)***to take your career to a whole new level!***

**23. What is the available mechanism for connecting applications when we run Hive as a server?**

* **Thrift Client**: Using Thrift, we can call Hive commands from various programming languages, such as C++, PHP, Java, Python, and Ruby.
* **JDBC Driver**: JDBC Driver enables accessing data with JDBC support, by translating calls from an application into SQL and passing the SQL queries to the Hive engine.
* **ODBC Driver**: It implements the ODBC API standard for the Hive DBMS, enabling ODBC-compliant applications to interact seamlessly with Hive.

**24. How do we write our own custom SerDe?**

Mostly, end-users prefer writing a Deserializer instead of using SerDe as they want to read their own data format instead of writing to it, e.g., RegexDeserializer deserializes data with the help of the configuration parameter ‘regex’ and with a list of column names.

If our SerDe supports DDL (i.e., SerDe with parameterized columns and column types), we will probably implement a protocol based on DynamicSerDe, instead of writing a SerDe. This is because the framework passes DDL to SerDe through the ‘Thrift DDL’ format and it’s totally unnecessary to write a “Thrift DDL” parser.

***Are you interested in learning Hadoop from experts? Enroll in our***[***Hadoop Course in Bangalore***](https://intellipaat.com/big-data-hadoop-training-bangalore/)***now!***

**25. Mention various date types supported by Hive.**

The timestamp data type stores date in the java.sql.timestamp format.

**Three collection data types in Hive are:**

* Arrays
* Maps
* Structs

**26. Can we run UNIX shell commands from Hive? Can Hive queries be executed from script files? If yes, how? Give an example.**

Yes, we can run UNIX shell commands from Hive using an ‘**!**‘ mark before the command. For example, **!pwd** at Hive prompt will display the current directory.  
We can execute Hive queries from the script files using the source command.

**Example**:

Hive> source /path/to/file/file\_with\_query.hql

**Hive Interview Question**

**1) Explain what is Hive?**

Hive is an ETL and Data warehousing tool developed on top of Hadoop Distributed File System (HDFS). It is a data warehouse framework for querying and analysis of data that is stored in HDFS. Hive is an open-source-software that lets programmers analyze large data sets on Hadoop.

**2) When to use Hive?**

* Hive is useful when making data warehouse applications
* When you are dealing with static data instead of dynamic data
* When application is on high latency (high response time)
* When a large data set is maintained
* When we are using queries instead of scripting

**3) Mention what are the different modes of Hive?**

Depending on the size of data nodes in Hadoop, Hive can operate in two modes.

These modes are,

* Local mode
* Map reduce mode

**4) Mention when to use Map reduce mode?**

Map reduce mode is used when,

* will perform on large amount of data sets and query going to execute in a parallel way
* Hadoop has multiple data nodes, and data is distributed across different node we use Hive in this mode
* Processing large data sets with better performance needs to be achieved

**5) Mention key components of Hive Architecture?**

Key components of Hive Architecture includes,

* User Interface
* Compiler
* Metastore
* Driver
* Execute Engine

**6) Mention what are the different types of tables available in Hive?**

There are two types of tables available in Hive.

* **Managed table**: In managed table, both the data and schema are under control of Hive
* **External table**: In the external table, only the schema is under the control of Hive.

**7) Explain what is Metastore in Hive?**

Metastore is a central repository in Hive.  It is used for storing schema information or metadata in the external database.

**8) Mention what Hive is composed of ?**

Hive consists of 3 main parts,

1. Hive Clients
2. Hive Services
3. Hive Storage and Computing

**9) Mention what are the type of database does Hive support ?**

For single user metadata storage, Hive uses derby database and for multiple user Metadata or shared Metadata case Hive uses MYSQL.

**10) Mention Hive default read and write classes?**

Hive default read and write classes are

1. TextInputFormat/HiveIgnoreKeyTextOutputFormat
2. SequenceFileInputFormat/SequenceFileOutputFormat

**11) Mention what are the different modes of Hive?**

Different modes of Hive depends on the size of data nodes in Hadoop.

These modes are,

* Local mode
* Map reduce mode

**12) Why is Hive not suitable for OLTP systems?**

Hive is not suitable for OLTP systems because it does not provide insert and update function at the row level.

**13) Mention what is the difference between Hbase and Hive?**

Difference between Hbase and Hive is,

* Hive enables most of the SQL queries, but HBase does not allow SQL queries
* Hive does not support record level insert, update, and delete operations on table
* Hive is a data warehouse framework whereas HBase is NoSQL database
* Hive run on the top of MapReduce, HBase runs on the top of HDFS

**14) Explain what is a Hive variable? What for we use it?**

Hive variable is created in the Hive environment that can be referenced by Hive scripts. It is used to pass some values to the hive queries when the query starts executing.

**15) Mention what is ObjectInspector functionality in Hive?**

ObjectInspector functionality in Hive is used to analyze the internal structure of the columns, rows, and complex objects.  It allows to access the internal fields inside the objects.

**16) Mention what is (HS2) HiveServer2?**

It is a server interface that performs following functions.

* It allows remote clients to execute queries against Hive
* Retrieve the results of mentioned queries

Some advanced features Based on Thrift RPC in its latest version include

* Multi-client concurrency
* Authentication

**17) Mention what Hive query processor does?**

Hive query processor convert graph of MapReduce jobs with the execution time framework.  So that the jobs can be executed in the order of dependencies.

**18) Mention what are the components of a Hive query processor?**

The components of a Hive query processor include,

* Logical Plan Generation
* Physical Plan Generation
* Execution Engine
* Operators
* UDF’s and UDAF’s
* Optimizer
* Parser
* Semantic Analyzer
* Type Checking

**19) Mention what is Partitions in Hive?**

Hive organizes tables into partitions.

* It is one of the ways of dividing tables into different parts based on partition keys.
* Partition is helpful when the table has one or more Partition keys.
* Partition keys are basic elements for determining how the data is stored in the table.

**20) Mention when to choose “Internal Table” and “External Table” in Hive?**

In Hive you can choose internal table,

* If the processing data available in local file system
* If we want Hive to manage the complete lifecycle of data including the deletion

You can choose External table,

* If processing data available in HDFS
* Useful when the files are being used outside of Hive

**21) Mention if we can name view same as the name of a Hive table?**

No. The name of a view must be unique compared to all other tables and as views present in the same database.

**22) Mention what are views in Hive?**

In Hive, Views are Similar to tables. They are generated based on the requirements.

* We can save any result set data as a view in Hive
* Usage is similar to as views used in SQL
* All type of DML operations can be performed on a view

**23) Explain how Hive Deserialize and serialize the data?**

Usually, while read/write the data, the user first communicate with inputformat. Then it connects with Record reader to read/write record.  To serialize the data, the data goes to row. Here deserialized custom serde use object inspector to deserialize the data in fields.

**24) What is Buckets in Hive?**

* The data present in the partitions can be divided further into Buckets
* The division is performed based on Hash of particular columns that is selected in the table.

**25) In Hive, how can you enable buckets?**

In Hive, you can enable buckets by using the following command,

set.hive.enforce.bucketing=true;

**26) In Hive, can you overwrite Hadoop MapReduce configuration in Hive?**

Yes, you can overwrite Hadoop MapReduce configuration in Hive.

**27) Explain how can you change a column data type in Hive?**

You can change a column data type in Hive by using command,

ALTER TABLE table\_name CHANGE column\_name column\_name new\_datatype;

**28) Mention what is the difference between order by and sort by in Hive?**

* SORT BY will sort the data within each reducer. You can use any number of reducers for SORT BY operation.
* ORDER BY will sort all of the data together, which has to pass through one reducer. Thus, ORDER BY in hive uses a single

**29) Explain when to use explode in Hive?**

Hadoop developers sometimes take an array as input and convert into a separate table row. To convert complex data types into desired table formats, Hive use explode.

**30) Mention how can you stop a partition form being queried?**

You can stop a partition form being queried by using the ENABLE OFFLINE clause with ALTER TABLE statement