Hive interview Questions & Answers

1. What is the definition of Hive? What is the present version of Hive?

Apache Hive is an **open-source data warehousing tool** for performing distributed processing and data analysis. It was developed by **Facebook** to reduce the work of writing the Java MapReduce program.

Apache Hive uses a **Hive Query language**, which is a declarative language similar to SQL. Hive translates the hive queries into MapReduce programs.

It supports developers to perform processing and analyses on structured and semi-structured data by replacing complex java MapReduce programs with hive queries.

One who is familiar with SQL commands can easily write the hive queries.

Hive makes the job easy for performing operations like

- Analysis of huge datasets
- Ad-hoc queries
- Data encapsulation

present version of Hive: **hive-4.0.0-alpha-1**

2. Is Hive suitable to be used for OLTP systems? Why?

No.

Hive is mainly used for batch processing i.e. OLAP and it is not used for OLTP because of the real-time operations of the database.

3. How is HIVE different from RDBMS? Does hive support ACID transactions. If not then give the proper reason.

RDBMS	HIVE
It is used to maintain database.	It is used to maintain data warehouse.
It uses SQL (Structured Query Language).	It uses HQL (Hive Query Language).
Schema is fixed in RDBMS.	Schema varies in it.
Normalized data is stored.	Normalized and de-normalized both type of data is stored.
Tables in RDBMS are sparse.	Table in HIVE are dense.
It doesn’t support partitioning.	It supports automation partition.
No partition method is used.	Shrading method is used for partition.

Older versions of Hive doesn’t support ACID transactions on tables. Though in newer versions it supports by default ACID transactions are disabled and you need to enable it before start using it.

Below are the properties you need to enable ACID transactions.

```
SET hive.support.concurrency=true;
SET hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
# The follwoing are not required if you are using Hive 2.0
SET hive.enforce.bucketing=true;
SET hive.exec.dynamic.partition.mode=nostrict;
# The following parameters are required for standalone hive metastore
SET hive.compactor.initiator.on=true;
SET hive.compactor.worker.threads=1
```

Besides this, you also need to create a Transactional table by using TBLPROPERTIES ('transactional'='true') at the time of creating Managed table.
Managed Table should use ORC format.

4. Explain the hive architecture and the different components of a Hive architecture?

The major components of Hive and its interaction with the Hadoop is demonstrated in the figure below and all the components are described further:

- **User Interface (UI) –**
  As the name describes User interface provide an interface between user and hive. It enables user to submit queries and other operations to the system. Hive web UI, Hive command line, and Hive HD Insight (In windows server) are supported by the user interface.

- **Hive Server** – It is referred to as Apache Thrift Server. It accepts the request from different clients and provides it to Hive Driver.

- **Driver** – Queries of the user after the interface are received by the driver within the Hive. Concept of session handles is implemented by driver. Execution and Fetching of APIs modelled on JDBC/ODBC interfaces is provided by the user.

- **Compiler** – Queries are parses, semantic analysis on the different query blocks and query expression is done by the compiler. Execution plan with the help of the table in the database and partition metadata observed from the metastore are generated by the compiler eventually.

- **Metastore –** All the structured data or information of the different tables and partition in the warehouse containing attributes and attributes level information are stored in the metastore. Sequences or de-sequences necessary to read and write data and the corresponding HDFS files where the data is stored. Hive selects corresponding database servers to stock the schema or Metadata of databases, tables, attributes in a table, data types of databases, and HDFS mapping.

- **Execution Engine –** Execution of the execution plan made by the compiler is performed in the execution engine. The plan is a DAG of stages. The dependencies within the various stages of the plan is managed by execution engine as well as it executes these stages on the suitable system components

.

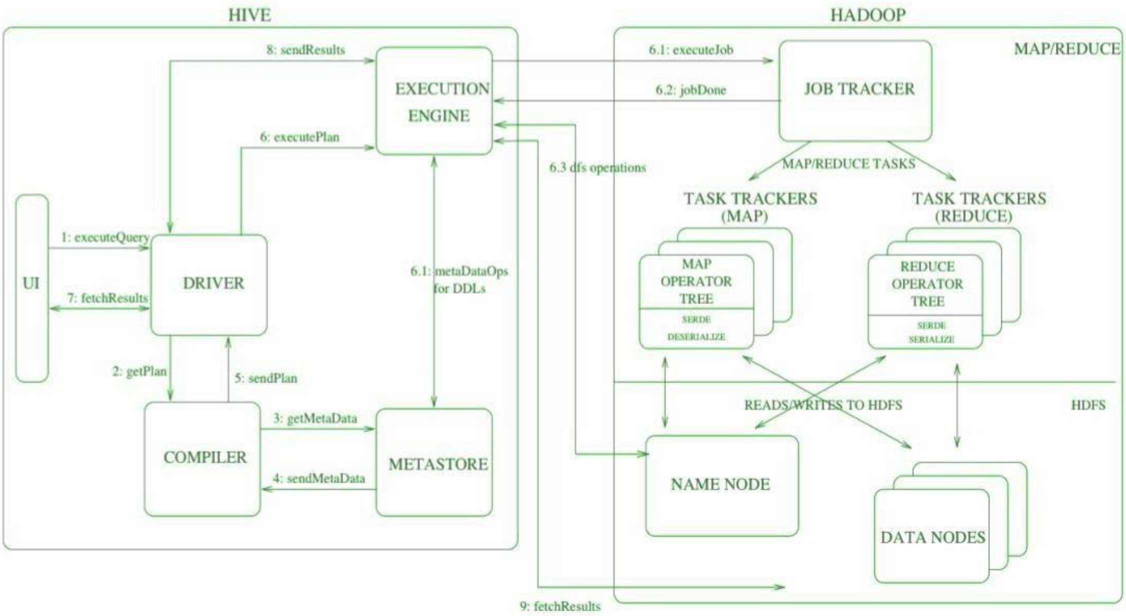


Diagram – Architecture of Hive that is built on the top of Hadoop

In the above diagram along with architecture, *job execution flow in Hive with Hadoop is demonstrated step by step*.

- **Step-1: Execute Query –**
  Interface of the Hive such as Command Line or Web user interface delivers query to the driver to execute. In this, UI calls the execute interface to the driver such as ODBC or JDBC.
- **Step-2: Get Plan –**
  Driver designs a session handle for the query and transfer the query to the compiler to make execution plan. In other words, driver interacts with the compiler.
- **Step-3: Get Metadata –**
  In this, the compiler transfers the metadata request to any database and the compiler gets the necessary metadata from the metastore.
- **Step-4: Send Metadata –**
  Metastore transfers metadata as an acknowledgment to the compiler.
- **Step-5: Send Plan –**
  Compiler communicating with driver with the execution plan made by the compiler to execute the query.
- **Step-6: Execute Plan –**
  Execute plan is sent to the execution engine by the driver.

  - Execute Job
  - Job Done
  - Dfs operation (Metadata Operation)
  - **Step-7: Fetch Results –**

  Fetching results from the driver to the user interface (UI).

- **Step-8: Send Results –**
  Result is transferred to the execution engine from the driver. Sending results to Execution engine. When the result is retrieved from data nodes to the execution engine, it returns the result to the driver and to user interface (UI).

5. Mention what Hive query processor does? And Mention what are the components of a Hive query processor?

Hive Query Processor uses metadata regarding tables, partitions and databases contained in MetaStore during plan generation.

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)

6. What are the three different modes in which we can operate Hive?

There are three modes for Hive Metastore deployment:

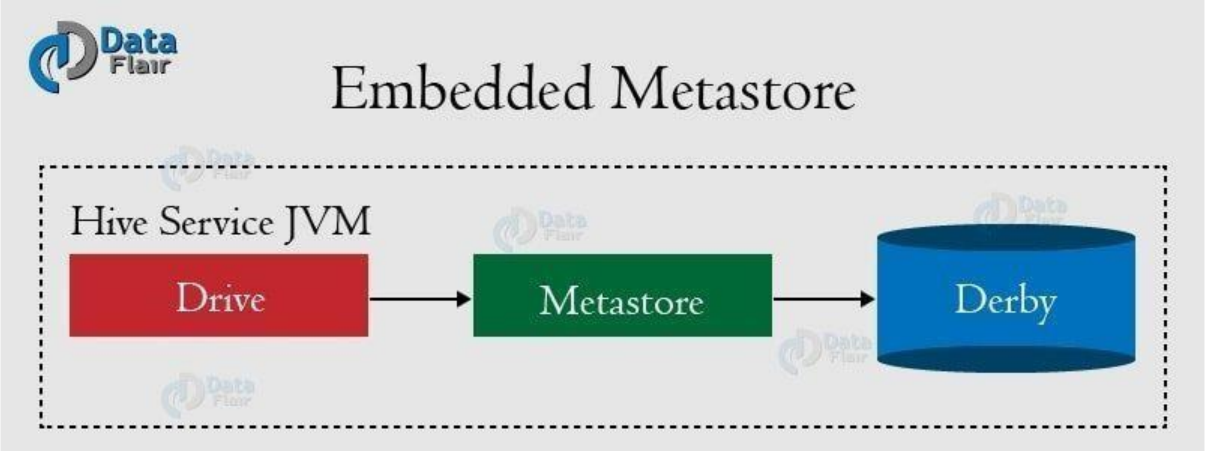
- Embedded Metastore
- Local Metastore
- Remote Metastore

Embedded Metastore:

In Hive by default, metastore service runs in the same JVM as the Hive service. It uses embedded derby database stored on the local file system in this mode. Thus both metastore service and hive service runs in the same JVM by using embedded Derby Database.

But, this mode also has limitation that, as only one embedded Derby database can access the database files on disk at any one time, so only one Hive session could be open at a time.

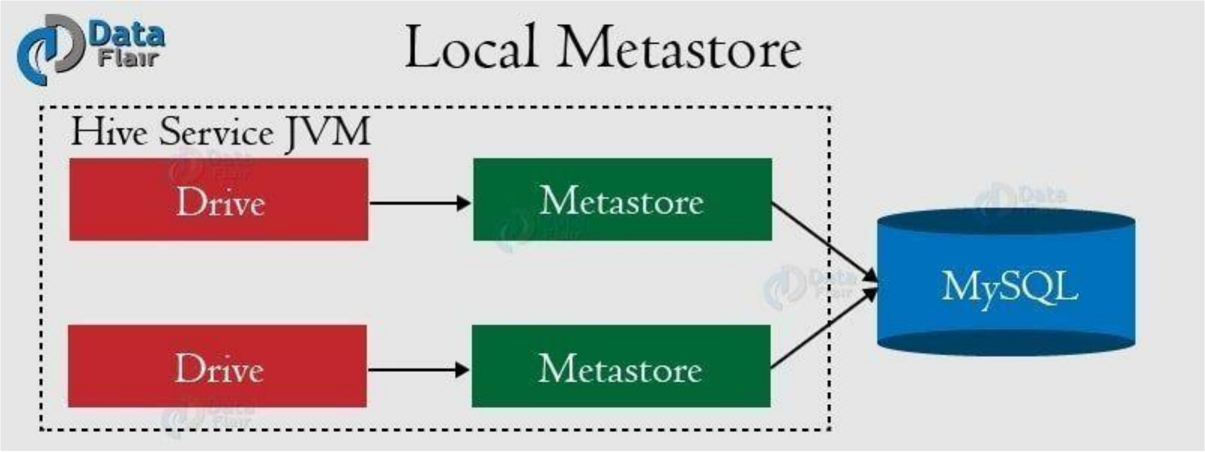
If we try to start the second session it produces an error when it attempts to open a connection to the metastore. So, to allow many services to connect the Metastore, it configures Derby as a network server. This mode is good for unit testing. But it is not good for the practical solutions.



Local Metastore :

Hive is the data-warehousing framework, so hive does not prefer single session. To overcome this limitation of Embedded Metastore, for **Local Metastore** was introduced. This mode allows us to have many Hive sessions i.e. many users can use the metastore at the same time.

We can achieve by using any JDBC compliant like MySQL which runs in a separate JVM or different machines than that of the Hive service and metastore service which are running in the same JVM.



This configuration is called as local metastore because metastore service still runs in the same process as the Hive. But it connects to a database running in a separate process, either on the same machine or on a remote machine.

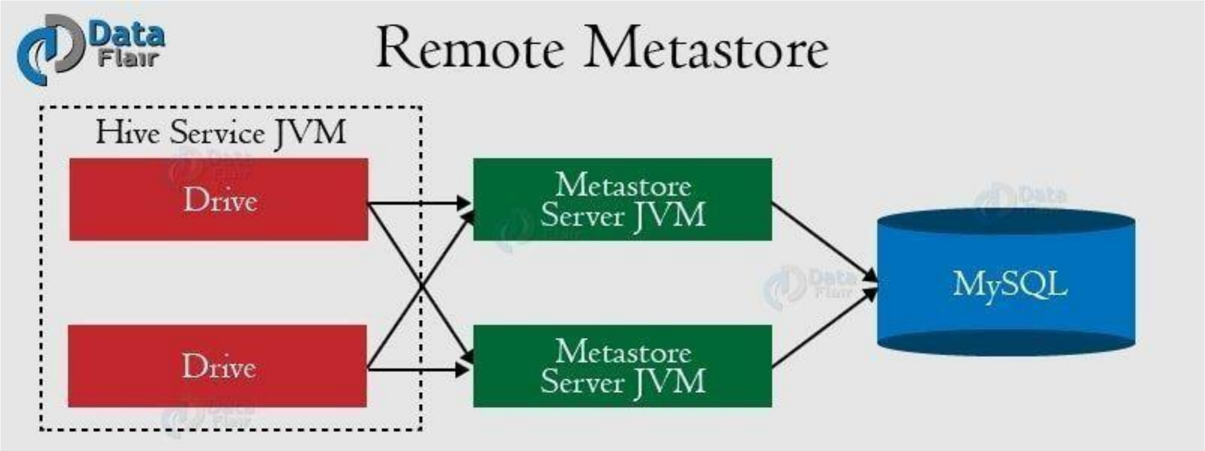
Before starting Apache Hive client, add the JDBC / ODBC driver libraries to the Hive lib folder**.**

MySQL is a popular choice for the standalone metastore. In this case, the ***javax.jdo.option.ConnectionURL*** property is set to ***jdbc:mysql://host/dbname?*** **createDatabaseIfNotExist=true**, and ***javax.jdo.option.ConnectionDriverName*** is set to ***com.mysql.jdbc.Driver.*** The JDBC driver JAR file for MySQL (Connector/J) must be on Hive's classpath, which is achieved by placing it in Hive's lib directory.

Remote Metastore :

Moving further, another metastore configuration called **Remote Metastore**. In this mode, metastore runs on its own separate JVM, not in the Hive service JVM. If other processes want to communicate with the metastore server they can communicate using Thrift Network APIs.

We can also have one more metastore servers in this case to provide more availability. This also brings better manageability/security because the database tier can be completely firewalled off. And the clients no longer need share database credentials with each Hiver user to access the metastore database.



To use this remote metastore, you should configure Hive service by setting *hive.metastore.uris* to the metastore server URI(s). Metastore server URIs are of the form *thrift://host:port,* where the port corresponds to the one set by METASTORE_PORT when starting the metastore server.

7. Features and Limitations of Hive.

⇨ **Features:**

Features	Explanation
Supported Computing Engine	Hive supports MapReduce, Tez, and Spark computing engine.
Framework	Hive is a stable batch-processing framework built on top of the Hadoop Distributed File system and can work as a data warehouse.
Easy To Code	Hive uses HIVE query language to query structure data which is easy to code. The 100 lines of java code we use to query a structure data can be minimized to 4 lines with HQL.
Declarative	HQL is a declarative language like SQL means it is non-procedural.
Structure Of Table	The table, the structure is similar to the RDBMS. It also supports partitioning and bucketing.

Supported data structures	Partition, Bucket, and tables are the 3 data structures that hive supports.
Supports ETL	Apache hive supports ETL i.e. Extract Transform and Load. Before Hive python is used for ETL.
Storage	Hive supports users to access files from HDFS, Apache HBase, Amazon S3, etc.
Capable	Hive is capable to process very large datasets of Petabytes in size.
Helps in processing unstructured data	We can easily embed custom MapReduce code with Hive to process unstructured data.
Drivers	JDBC/ODBC drivers are also available in Hive.
Fault Tolerance	Since we store Hive data on HDFS so fault tolerance is provided by Hadoop.
Area of uses	We can use a hive for data mining, predictive modelling, and document indexing.

⇨ **Limitations:**

Limitation	Explanation
Does not support OLAP	Apache Hive doesn’t support online transaction processing (OLTP) but Online Analytical Processing(OLAP) is supported.
No updation and Deletion	Hive does not support update and delete operation on tables.
Doesn’t support subqueries	Subqueries are not supported.
Latency	The latency in the apache hive query is very high.
Only non-real or cold data is supported	Hive is not used for real-time data querying since it takes a while to produce a result.
Transaction processing is not supported	HQL does not support the Transaction processing feature.

8. How to create a Database in HIVE?

Create Database is a statement used to create a database in Hive. A database in Hive is a **namespace** or a collection of tables. The **syntax** for this statement is as follows:

```
CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>
```

Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command. The following query is executed to create a database named **archi**:

```
hive> CREATE DATABASE [IF NOT EXISTS] archi;
OR
hive> CREATE SCHEMA archi;
```

The following query is used to verify a databases list:

hive> SHOW DATABASES;

9. How to create a table in HIVE?

```
create table customers
(
id int,
name string,
age int,
address string,
salary int
)
row format delimited
fields terminated by ','
tblproperties ("skip.header.line.count" = "1");
```

10. What do you mean by describe and describe extended and describe formatted with respect to database and table.

Describe- This will show table columns

Describe extended - This will show table columns, data types, and other details of the table. Other details will be displayed in single line.

Describe formatted - This will show table columns, data types, and other details of the table. Other details will be displayed into multiple lines.

11. How to skip header rows from a table in Hive?

```
TBLPROPERTIES("skip.header.line.count"="1");
ALTER TABLE tablename
SET TBLPROPERTIES ("skip.header.line.count"="1");
```

12. What is a hive operator? What are the different types of hive operators?

Apache Hive provides various Built-in operators for data operations to be implemented on the tables present inside Apache Hive warehouse.

Hive operators are used for mathematical operations on operands. It returns specific value as per the logic applied.

Types of Hive Built-in Operators

- **Relational Operators**
- **Arithmetic Operators**
- **Logical Operators**
- **String Operators**
- **Operators on Complex Types**

13. Explain about the Hive Built-In Functions

Hive supports the following built-in functions:

Return Type	Signature	Description
BIGINT	round(double a)	It returns the rounded BIGINT value of the double.
BIGINT	floor(double a)	It returns the maximum BIGINT value that is equal or less than the double.
BIGINT	ceil(double a)	It returns the minimum BIGINT value that is equal or greater than the double.
double	rand(), rand(int seed)	It returns a random number that changes from row to row.
string	concat(string A, string B,...)	It returns the string resulting from concatenating B after A.
string	substr(string A, int start)	It returns the substring of A starting from start position till the end of string A.
string	substr(string A, int start, int length)	It returns the substring of A starting from start position with the given length.
string	upper(string A)	It returns the string resulting from converting all characters of A to upper case.
string	ucase(string A)	Same as above.
string	lower(string A)	It returns the string resulting from converting all characters of B to lower case.

string	lcase(string A)	Same as above.

string	trim(string A)	It returns the string resulting from trimming spaces from both ends of A.
string	ltrim(string A)	It returns the string resulting from trimming spaces from the beginning (left hand side) of A.
string	rtrim(string A)	rtrim(string A) It returns the string resulting from trimming spaces from the end (right hand side) of A.
string	regexp_replace(string A, string B, string C)	It returns the string resulting from replacing all substrings in B that match the Java regular expression syntax with C.
int	size(Map<K.V>)	It returns the number of elements in the map type.
int	size(Array<T>)	It returns the number of elements in the array type.
value of <type>	cast(<expr> as <type>)	It converts the results of the expression expr to <type> e.g. cast('1' as BIGINT) converts the string '1' to it integral representation. A NULL is returned if the conversion does not succeed.
string	from_unixtime(int unixtime)	convert the number of seconds from Unix epoch (1970-01-01 00:00:00 UTC) to a string representing the timestamp of that moment in the current system time zone in the format of "1970-01-01 00:00:00"
string	to_date(string timestamp)	It returns the date part of a timestamp string: to_date("1970-01-01 00:00:00") = "1970-01-01"

int	year(string date)	It returns the year part of a date or a timestamp string: year("1970-01-01 00:00:00") = 1970, year("1970-01-01") = 1970
int	month(string date)	It returns the month part of a date or a timestamp string: month("1970-11-01 00:00:00") = 11, month("1970-11-01") = 11
int	day(string date)	It returns the day part of a date or a timestamp string: day("1970-11-01 00:00:00") = 1, day("1970-11-01") = 1
string	get_json_object(string json_string, string path)	It extracts json object from a json string based on json path specified, and returns json string of the extracted json object. It returns NULL if the input json string is invalid.

14. Write hive DDL and DML commands.

Hive DDL commands:

1. CREATE
2. SHOW
3. DESCRIBE
4. USE
5. DROP
6. ALTER
7. TRUNCATE

Hive DML commands:

1. LOAD
2. SELECT
3. INSERT
4. DELETE
5. UPDATE
6. EXPORT
7. IMPORT

15.Explain about SORT BY, ORDER BY, DISTRIBUTE BY and CLUSTER BY in Hive.

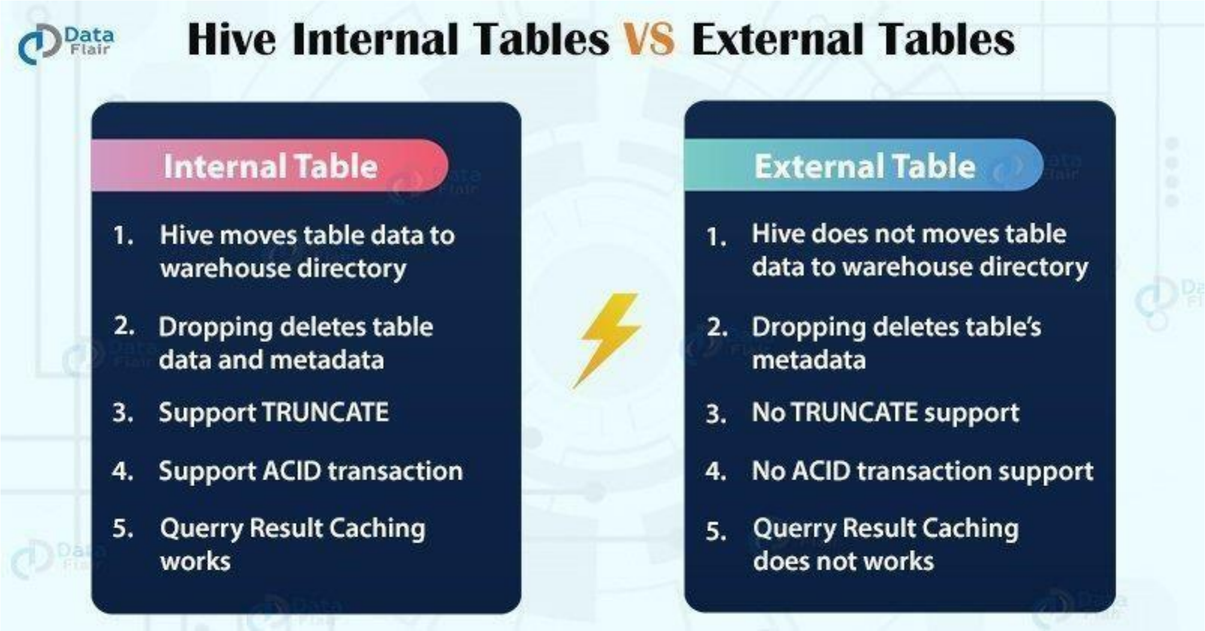
- **ORDER BY x:** guarantees global ordering, but does this by pushing all data through just one reducer. This is basically unacceptable for large datasets. You end up one sorted file as output.

- **SORT BY x:** orders data at each of N reducers, but each reducer can receive overlapping ranges of data. You end up with N or more sorted files with overlapping ranges.

- **DISTRIBUTE BY x:** ensures each of N reducers gets non-overlapping ranges of x, but doesn't sort the output of each reducer. You end up with N or more unsorted files with non-overlapping ranges.

- **CLUSTER BY x:** ensures each of N reducers gets non-overlapping ranges, then sorts by those ranges at the reducers. This gives you global ordering, and is the same as doing (DISTRIBUTE BY x and SORT BY x). You end up with N or more sorted files with non-overlapping ranges.

16.Difference between "Internal Table" and "External Table" and Mention when to choose “Internal Table” and “External Table” in Hive?



Hive Internal Tables VS External Tables

Internal Table

1. Hive moves table data to warehouse directory
2. Dropping deletes table data and metadata
3. Support TRUNCATE
4. Support ACID transaction
5. Querry Result Caching works

External Table

1. Hive does not moves table data to warehouse directory
2. Dropping deletes table’s metadata
3. No TRUNCATE support
4. No ACID transaction support
5. Querry Result Caching does not works

Use an internal table if:

1. Data is temporary and doesn’t affect businesses in real time.

2. If you want the hive to manage the data and the tables.

Use an external table if:

1. You want to use data outside HIVE for performing a different operation such as loading and merging.

2. The data is of production quality.

17. Where does the data of a Hive table get stored?

Hive data are stored in one of **Hadoop compatible filesystem**:
S3, HDFS or other compatible filesystem.
HDFS path : user/hive/warehouse

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

The LOCATION keyword, we can change the default location of Managed tables while creating the managed table in hive. However, to do so, the user needs to specify the storage path of the managed table as the value to the LOCATION keyword, that will help to change the default location of a managed table.

19. What is a metastore in Hive? What is the default database provided by Apache Hive for metastore?

The Hive metastore is simply a relational database. It stores metadata related to the tables/schemas you create to easily query big data stored in HDFS. When you create a new Hive table, the information related to the schema (column names, data types) is stored in the Hive metastore relational database. Other information like input/output formats, partitions, HDFS locations are all stored in the metastore.

Derby is the default database for the embedded metastore.
Local/Embedded Metastore Database (Derby)

20. Why does Hive 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.

21. What is a partition in Hive? And Why do we perform partitioning in Hive?

Hive table partition is a way to split a large table into smaller logical tables based on one or more partition keys. These smaller logical tables are not visible to users and users still access the data from just one table. Partition eliminates creating smaller tables, accessing, and managing them separately.

Partition Table Advantages:

- Fast accessed to the data
- Provides the ability to perform an operation on a smaller dataset

22.What is the difference between dynamic partitioning and static partitioning?

Static partitioning :

In static partitioning, we need to specify the partition column value in each and every LOAD statement.
suppose we are having partition on column country for table t1(userid, name,occupation, country), so each time we need to provide country value

```
hive>LOAD DATA INPATH '/hdfs path of the file' INTO TABLE t1 PARTITION(country="US")
hive>LOAD DATA INPATH '/hdfs path of the file' INTO TABLE t1 PARTITION(country="UK")
```

Dynamic Partitioning:

Dynamic partition allow us not to specify partition column value each time. the approach we follows is as below:
create a non-partitioned table t2 and insert data into it.
now create a table t1 partitioned on intended column(say country).
load data in t1 from t2 as below:
hive> INSERT INTO TABLE t2 PARTITION(country) SELECT * from T1;
make sure that partitioned column is always the last one in non partitioned table(as we are having country column in t2)

When to use static partitioning and when to use dynamic partitioning?

- Use static partitioning when data is already physically categorized/grouped/partitioned and ready to be added as a partition to a table.
- Static partitioning will not result in MapReduce job execution since the data is already physically categorized/partitioned.
- Use dynamic partitioning when data is not already physically categorized/grouped/partitioned.
- Dynamic partitioning will result in MapReduce job execution to group the data first and then partitions will be added to the table.

23. How do you check if a particular partition exists?

we can check whether a particular partition exists or not using below query:
SHOW PARTITIONS table_name
PARTITION(partitioned_column='partition_value')

24. How can you stop a partition form being queried?

By using the ENABLE OFFLINE clause with ALTER TABLE statement.
Syntax:
ALTER TABLE t1 PARTITION (PARTITION_SPEC) ENABLE OFFLINE;
Example:
Hive> ALTER TABLE TownsList_Dynamic
PARTITION (country='England') ENABLE OFFLINE;
Now, let's issue the SELECT statement.
*Hive> select * from TownsList_Dynamic where country='England';*

25. Why do we need buckets? How Hive distributes the rows into buckets?

Bucketing is a method to evenly distributed the data across many files.
Create multiple buckets and then place each record into one of the buckets based on some logic mostly some hashing algorithm.
Bucketing feature of Hive can be used to distribute/organize the table/partition data into multiple files such that similar records are present in the same file.
While creating a Hive table, a user needs to give the columns to be used for bucketing and the number of buckets to store the data into. Which records go to which bucket are decided by the Hash value of columns used for bucketing.

26. In Hive, how can you enable buckets?

set.hive.enforce.bucketing=true;

27. How does bucketing help in the faster execution of queries?

In bucketing, the partitions can be subdivided into buckets based on the hash function of a column. It gives extra structure to the data which can be used for more efficient queries.

28. How to optimise Hive Performance? Explain in very detail.

1 Avoid locking of tables

It is extremely important to make sure that the tables are being used in any Hive query as sources are not being used by another process. This can lead to locking of the table and our query can be stuck for an unknown time.

We can use the parameters below for making sure that the tables are not being locked:

set hive.support.concurrency=false; set
hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DummyTxnManager;
set hive.bin.strict.locking.mode=false;

2 Use the Hive execution engine as TEZ

While executing Hive queries, TEZ execution engine is the preferred choice because it eliminates unnecessary disk access. It takes data from disk once, performs calculations, and produces output, thus saving us from multiple disk traversals. We can consider TEZ to be a much more flexible and powerful successor to the map-reduce framework. We can set the parameter below for using TEZ engine:

set hive.execution.engine=tez;

3 Use Hive Cost Based Optimizer (CBO)

Apache Hive provides a cost-based optimizer to improve performance. It generates efficient execution plans like how to order joins, which type of join to perform, the degree of parallelism etc. by examining the query cost. These decisions are collected by ANALYZE statements or the metastore itself, ultimately cutting down on query execution time and reducing resource utilization.

We can set the parameter using :

set hive.cbo.enable=true;

4 Parallel execution at a Mapper & Reducer level

We can improve the performance of aggregations, filters, and joins of our hive queries by using vectorized query execution, which means scanning them in batches of 1024 rows at once instead of single row each time.

We should explore the below parameters which will help to bring in more parallelism and which significantly improves query execution time:

set hive.vectorized.execution.enabled=true;
set hive.exec.parallel=true;

For example:

```
Select a.*, b.* from
(select * from table1 ) a
Join
(select * from table2 ) b
On a.id=b.id ;
```

As we can see the two subqueries are independent so this might increase efficiency.
One important thing to note is, parallel execution will increase cluster utilization. If the cluster utilization of a cluster is already very high, parallel execution will not help much.

5 Use STREAMTABLE option

When we are joining multiple tables, we can use STREAMTABLE option. By default, the right-most table gets streamed.
For example:
If we are joining 2 tables 'huge_table' join 'small_table', by default 'small_table' gets streamed as it is the rightmost table. In this case, 'huge_table' , being the bigger table, will try to get buffered into memory and might cause java heap space issues or the job might run longer. In this case, what we can do is add /*+ STREAMTABLE('huge_table') */ and it will make 'huge_table' to be streamed rather than coming into memory.
Hence, in this way, we can be free of remembering the order of joining tables.

6 Use Map Side JOIN Option

If one of the tables in the join is a small table and can be loaded into memory, we can force a MAPSIDE join like shown below:

```
Select /*+ MAPJOIN(small_table) */ large_table.col1,large_table.col2 from large_table join
small_table on large_table.col1 = small_table.col1;
```

A map side join can be performed within a mapper without using a Map/Reduce step.

Also, We can let the execution engine take care of this by setting auto.convert.join as True.

```
set auto.convert.join = True :
```

7 Avoid Calculated Fields in JOIN and WHERE clause

We should avoid using any calculated fields in JOIN and WHERE clauses as they take a long time to run the Hive query. We can use CTE(Create table expression) to handle those functionalities and can optimize our queries.
For example:

Original Query:

```
select a.coll, b.col2 from table1 as a join table2 as b on (a.coll +50 = b.col2);
```

Optimized query:

```
with CTE as
(select a.col1 + 50 as C1 FROM table1 ) select CTE.C1, b.col2
from CTE join table2 b on (CTE.C1 = b.col2);
```

8 Use SORT BY instead of ORDER BY

Hive supports both ORDER BY and SORT BY causes. ORDER BY works on a single reducer and it causes a performance bottleneck. But, SORT BY orders the data only within each reducer and performs a local ordering where each reducer's output will be sorted ensuring better performance.

9 Select columns which are needed

While we are using Hive, If we need only a few columns from a table, avoid using SELECT * FROM as it adds unnecessary time to the execution.

10 Suitable Input format Selection

Using appropriate file formats on the basis of data can significantly increase our query performance. Hive comes with columnar input formats like RCFile, ORC, etc. On comparing to Text, Sequence, and RC file formats, ORC shows better performance because Hive has a vectorized ORC reader which allows reducing the read operations in analytics queries by allowing each column to be accessed individually.

11 Limit (Filter) the data as early as possible

This is the fundamental principle for any tuning where we filter or drop records ahead in the process so that we can avoid dealing with long-running of queries and dealing with unnecessary results.
For example :
a join b where a.col !=null can
be written as
*(select * from a where a.col!=null) join b*

12 Use Multi Query Inserts

Here we will have a common dataset (Superset) and then we will use that to insert into multiple tables based on specific conditions specified in the WHERE clause. This helps to load multiple tables in parallel when they have the common superset of data.

13 Modularize the code into logical pieces

This helps in terms of the maintenance of the code. Also helps in restarting the jobs in scenarios of failures. This can be used to achieve parallelism by running modules that are independent of each other thus saving time.

29. What is the use of Hcatalog?

HCatalog is a tool that allows you to access Hive metastore tables within Pig, Spark SQL, and/or custom MapReduce applications. HCatalog has a REST interface and command line client that allows you to create tables or do other operations. You then write your applications to access the tables using HCatalog libraries.

30. Explain about the different types of joins in Hive.

JOIN clause is used to combine and retrieve the records from multiple tables. JOIN is same as OUTER JOIN in SQL. A JOIN condition is to be raised using the primary keys and foreign keys of the tables.

The following query executes JOIN on the CUSTOMER and ORDER tables, and retrieves the records:

```
hive> SELECT c.ID, c.NAME, c.AGE, o.AMOUNT FROM CUSTOMERS c JOIN ORDERS o ON (c.ID =
o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

```
+----+----------+-----+--------+
| ID | NAME     | AGE | AMOUNT |
+----+----------+-----+--------+
| 3  | kaushik  | 23  | 3000   |
| 3  | kaushik  | 23  | 1500   |
| 2  | Khilan   | 25  | 1560   |
| 4  | Chaitali | 25  | 2060   |
+----+----------+-----+--------+
```

LEFT OUTER JOIN

The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no matches in the right table. This means, if the ON clause matches 0 (zero) records in the right table, the JOIN still returns a row in the result, but with NULL in each column from the right table.
A LEFT JOIN returns all the values from the left table, plus the matched values from the right table, or NULL in case of no matching JOIN predicate.

The following query demonstrates LEFT OUTER JOIN between CUSTOMER and ORDER tables:

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE
FROM CUSTOMERS c
LEFT OUTER JOIN ORDERS o
ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

```
+----+----------+--------+---------------------+
| ID | NAME     | AMOUNT | DATE                |
+----+----------+--------+---------------------+
| 1  | Ramesh   | NULL   | NULL                |
| 2 | Khilan    | 1560   | 2009-11-20 00:00:00 |
| 3  | kaushik  | 3000   | 2009-10-08 00:00:00 |
| 3  | kaushik  | 1500   | 2009-10-08 00:00:00 |
| 4  | Chaitali | 2060   | 2008-05-20 00:00:00 |
| 5  | Hardik   | NULL   | NULL                |
| 6  | Komal    | NULL   | NULL                |
| 7  | Muffy    | NULL   | NULL                |
+----+----------+--------+---------------------+
```

RIGHT OUTER JOIN

The HiveQL RIGHT OUTER JOIN returns all the rows from the right table, even if there are no matches in the left table. If the ON clause matches 0 (zero) records in the left table, the JOIN still returns a row in the result, but with NULL in each column from the left table.
A RIGHT JOIN returns all the values from the right table, plus the matched values from the left table, or NULL in case of no matching join predicate.

The following query demonstrates RIGHT OUTER JOIN between the CUSTOMER and ORDER tables.

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM CUSTOMERS c RIGHT OUTER JOIN ORDERS o
ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

```
+------+----------+--------+---------------------+
| ID   | NAME     | AMOUNT | DATE                |
+------+----------+--------+---------------------+
| 3    | kaushik  | 3000   | 2009-10-08 00:00:00 |
| 3    | kaushik  | 1500   | 2009-10-08 00:00:00 |
| 2    | Khilan   | 1560   | 2009-11-20 00:00:00 |
| 4    | Chaitali | 2060   | 2008-05-20 00:00:00 |
+------+----------+--------+---------------------+
```

FULL OUTER JOIN

The HiveQL FULL OUTER JOIN combines the records of both the left and the right outer tables that fulfil the JOIN condition. The joined table contains either all the records from both the tables, or fills in NULL values for missing matches on either side.

The following query demonstrates FULL OUTER JOIN between CUSTOMER and ORDER tables: hive>

```
SELECT c.ID, c.NAME, o.AMOUNT, o.DATE

FROM CUSTOMERS c
FULL OUTER JOIN ORDERS o
ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

```
+------+----------+--------+---------------------+
| ID   | NAME     | AMOUNT | DATE                |
+------+----------+--------+---------------------+
| 1    | Ramesh   | NULL   | NULL                |
| 2   | Khilan   | 1560   | 2009-11-20 00:00:00 |
| 3    | kaushik  | 3000   | 2009-10-08 00:00:00 |
| 3    | kaushik  | 1500   | 2009-10-08 00:00:00 |
| 4   | Chaitali | 2060   | 2008-05-20 00:00:00 |
| 5    | Hardik   | NULL   | NULL                |
| 6    | Komal    | NULL   | NULL                |
| 7    | Muffy    | NULL   | NULL                |
| 3    | kaushik  | 3000   | 2009-10-08 00:00:00 |
| 3    | kaushik  | 1500   | 2009-10-08 00:00:00 |
| 2    | Khilan   | 1560   | 2009-11-20 00:00:00 |
| 4    | Chaitali | 2060   | 2008-05-20 00:00:00 |
+------+----------+--------+---------------------+
```

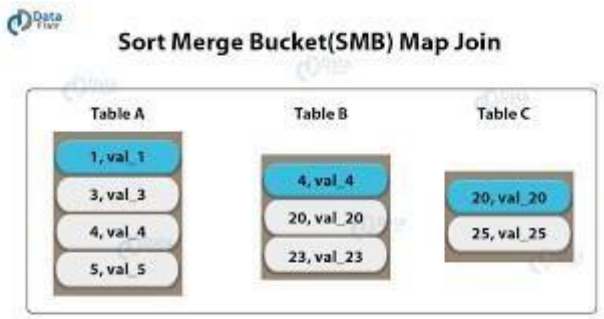
31. Is it possible to create a Cartesian join between 2 tables, using Hive?

Yes

```
join_condition
 | table_reference [CROSS] JOIN table_reference join_condition
```

32. Explain the SMB Join in Hive?

Basically, in Mapper, only Join is done. Moreover, all the buckets are joined with each other at the mapper which are corresponding. In Hive, while each mapper reads a bucket from the first table and the corresponding bucket from the second table, in SMB join. Basically, then we perform a merge sort join feature. Moreover, we mainly use it when there is no limit on file or partition or table join. Also, when the tables are large we can use Hive Sort Merge Bucket join. However, using the join columns, all join the columns are bucketed and sorted in SMB. Although, make sure in SMB join all tables should have the same number of buckets.



33. What is the difference between order by and sort by which one we should use?

Hive supports SORT BY which sorts the data per reducer. The difference between "order by" and "sort by" is that the former guarantees total order in the output while the latter only guarantees ordering of the rows within a reducer. If there are more than one reducer, "sort by" may give partially ordered final results.

Note: It may be confusing as to the difference between SORT BY alone of a single column and CLUSTER BY. The difference is that CLUSTER BY partitions by the field and SORT BY if there are multiple reducers partitions randomly in order to distribute data (and load) uniformly across the reducers. Basically, the data in each reducer will be sorted according to the order that the user specified.

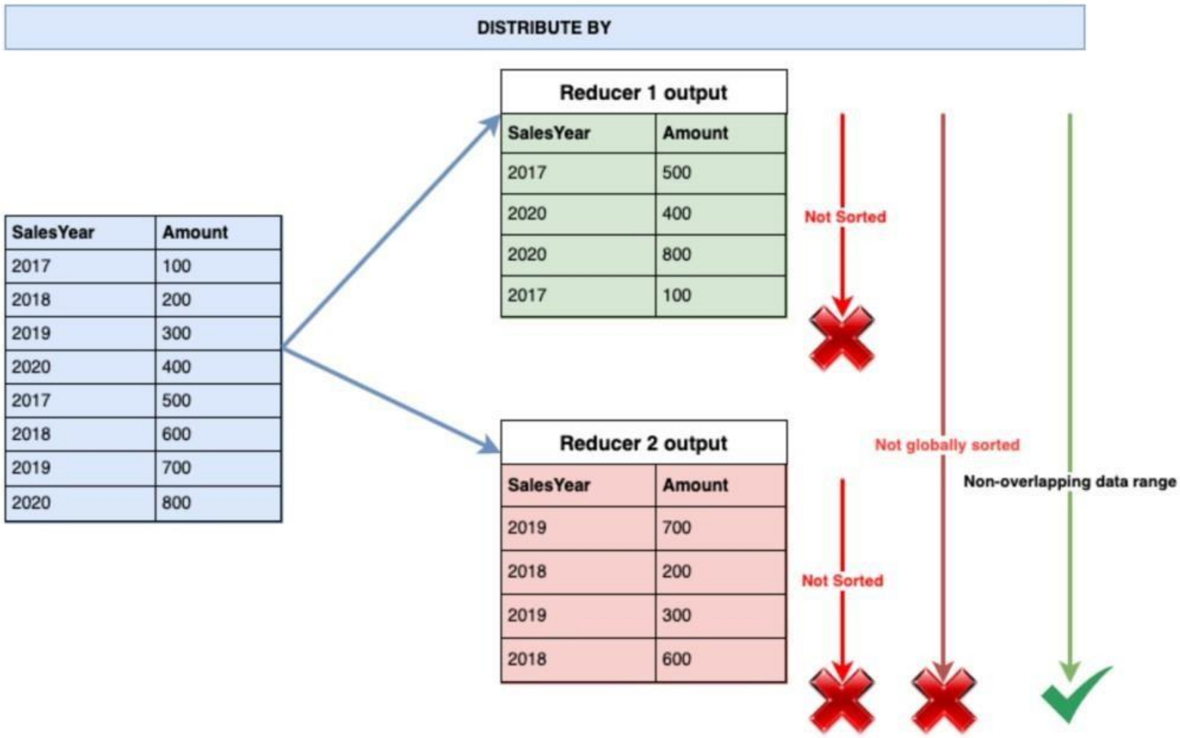
The following example shows :

SELECT key, value FROM src SORT BY key ASC, value DESC

34. What is the usefulness of the DISTRIBUTED BY clause in Hive?

DISTRIBUTE BY clause is used to **distribute the input rows among reducers**. It ensures that all rows for the same key columns are going to the same reducer. So, if we need to partition the data on some key column, we can use the DISTRIBUTE BY clause in the hive queries. However, the DISTRIBUTE BY clause **does not sort the data either at the reducer level or globally**. Also, the same key values might not be placed next to each other in the output dataset.
As a result, the DISTRIBUTE BY clause may **output N number of unsorted files** where N is the number of reducers used in the query processing. But, the output files do not contain overlapping data ranges.
The syntax of the DISTRIBUTE BY clause in hive is as below:
SELECT Col1, Col2,……ColN FROM TableName DISTRIBUTE BY Col1, Col2, ….. ColN

SELECT SalesYear, Amount **FROM** tbl_Sales DISTRIBUTE **BY** SalesYear;



SalesYear	Amount
2017	100
2018	200
2019	300
2020	400
2017	500
2018	600
2019	700
2020	800

Reducer 1 output

SalesYear	Amount
2017	500
2020	400
2020	800
2017	100

Reducer 2 output

SalesYear	Amount
2019	700
2018	200
2019	300
2018	600

35. How does data transfer happen from HDFS to Hive?

To query data in HDFS in Hive, you apply a schema to the data and then store data in ORC format. Incrementally update the imported data.

Updating imported tables involves importing incremental changes made to the original table using Sqoop and then merging changes with the tables imported into Hive.

36. Wherever (Different Directory) I run the hive query, it creates a new metastore_db, please explain the reason for it?

Basically, it creates the local metastore, while we run the hive in embedded mode. Also, it looks whether metastore already exist or not before creating the metastore. The property of interest here is *javax.jdo.option.ConnectionURL.* The default value of this property is *jdbc:derby:;databaseName=metastore_db;create=true.*

This value specifies that you will be using embedded derby as your Hive metastore and the location of the metastore is metastore_db. Also, the metastore will be created if it doesn't already exist. Note that the location of the metastore (metastore_db) is a relative path. Therefore, it gets created where you launch Hive from. If you update this property (in your hive-site.xml) to be, say an absolute path to a location, the metastore will be used from that location.

37. What will happen in case you have not issued the command: 'SET hive.enforce.bucketing=true;' before bucketing a table in Hive?

We need to set the property hive.enforce.bucketing = true, so that Hive knows to create the number of buckets declared in the table definition to populate the bucketed table.

38. Can a table be renamed in Hive?

You can rename the table name in the hive.
You need to use the alter command.
This command allows you to change the table name as shown below.
$ ALTER TABLE name RENAME TO new_name

39. Write a query to insert a new column(new_col INT) into a hive table at a position before an existing column (x_col)

```
ALTER TABLE table_name
CHANGE COLUMN new_col
INT BEFORE x_col;
```

40. What is serde operation in HIVE?

SerDe means Serializer and Deserializer. Hive uses SerDe and FileFormat to read and write table rows. Main use of SerDe interface is for IO operations. A SerDe allows hive to read the data from the table and write it back to the HDFS in any custom format. If we have unstructured data, then we use RegEx SerDe which will instruct hive how to handle that record.We can also write our own Custom SerDe in any format. Let us see the definition of Serializer and Deserailizer.

Deserailizer

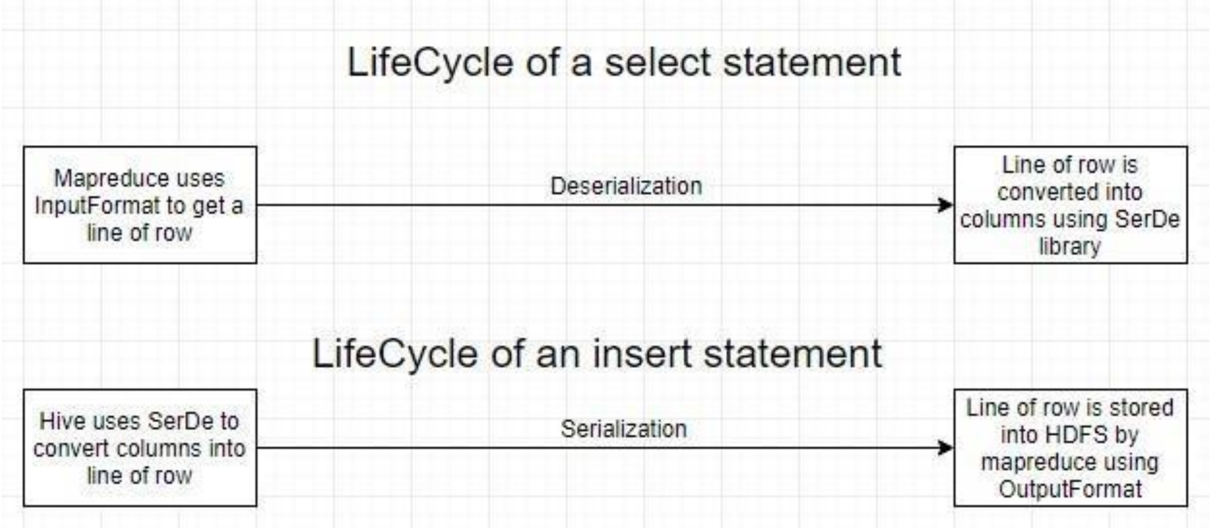
Deserializer is conversion of string or binary data into java object when we any submit any query.

Serailizer

Serializer converts java object into string or binary object. It is used when writing the data such as insert- select statements.Hive currently uses these FileFormat classes to read and write HDFS files:

- TextInputFormat/HiveIgnoreKeyTextOutputFormat: These 2 classes read/write data in plain text file format.
- SequenceFileInputFormat/SequenceFileOutputFormat: These 2 classes read/write data in Hadoop SequenceFile format.

41. Explain how Hive Deserializes and serialises the data?


LifeCycle of a select statement
Mapreduce uses InputFormat to get a line of row
Deserialization
Line of row is converted into columns using SerDe library
LifeCycle of an insert statement
Hive uses SerDe to convert columns into line of row
Serialization
Line of row is stored into HDFS by mapreduce using OutputFormat

42. Write the name of the built-in serde in hive.

The Hive SerDe library is in org.apache.hadoop.hive.serde2. (The old SerDe library in org.apache.hadoop.hive.serde is deprecated.)

43. What is the need of custom Serde?

Despite Hive SerDe users want to write a Deserializer in most cases. It is because users just want to read their own data format instead of writing to it By using the configuration parameter 'regex', the RegexDeserializer will deserialize the data, and possibly a list of column names (see serde2.MetadataTypedColumnsetSerDe).

Some important points about Writing Hive SerDe:

Basically, Hive SerDe, not the DDL, defines the table schema. Since some of the SerDe in Hive are implementations use the DDL for configuration. However, the SerDe can also override that. Moreover, Column types can be arbitrarily nested arrays, maps, and structures. However, with CASE/IF or when using complex or nested types the callback design of ObjectInspector allows lazy deserialization.

44. Can you write the name of a complex data type (collection data types) in Hive?

In addition to primitive data types, Hive also supports a few complex data types: Struct, MAP , and Array . Complex data types are also known as collection data types.

45. Can hive queries be executed from script files? How?

Yes.
Hive> source /path/to/file/file_with_query.hql

46. What are the default record and field delimiter used for hive text files?

The default record delimiter is - \n

And the filed delimiters are - \001,\002,\003

47. **How do you list all databases in Hive whose name starts with s?**

SHOW (DATABASES|SCHEMAS) [LIKE identifier_with_wildcards];

SHOW DATABASES LIKE 's%';

48. What is the difference between LIKE and RLIKE operators in Hive?

LIKE is an operator similar to LIKE in SQL. We use LIKE to search for string with similar text.

RLIKE (Right-Like) is a special function in Hive where if any substring of A matches with B then it evaluates to true. It also obeys Java regular expression pattern.

49. How to change the column data type in Hive?

ALTER TABLE table_name CHANGE column_name column_name new_datatype;

50. How will you convert the string '51.2' to a float value in the particular column?

select cast ('51.2' as float)

51. What will be the result when you cast 'abc' (string) as INT?

It prints NULL

52. What does the following query do? a. INSERT OVERWRITE TABLE employees b. PARTITION (country, state) c. SELECT ..., se.cnty, se.st d. FROM staged_employees se;

a. inserts or overwrites values in table 'employees'

b. it is used for a windows function where partitions are required for country and state attributes

c. selects 'cnty' column from 'staged_employees' table declared as 'se'

53. Write a query where you can overwrite data in a new table from the existing table.

create table B like A;

INSERT OVERWRITE TABLE B SELECT A.value FROM A;

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

By default, **the columns metadata for Hive does not specify a maximum data length for STRING columns**. The driver has the parameter DefaultStringColumnLength, default is 255 maximum value.

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 roworiented file with a general binary format.

55. What File Formats and Applications Does Hive Support?

Hive and Impala tables in HDFS can be created using text files. Sequence files, ORC files, Avro data files, and Parquet file formats. Data serialization is a way of representing data in memory as a series of bytes. Avro is an efficient data serialization framework and is widely supported throughout Hadoop and its ecosystem.

56. How do ORC format tables help Hive to enhance its performance?

Using the ORC format leads to a reduction in the size of the data stored, as this file format has high compression ratios. As the data size is reduced, the time to read and write the data is also reduced.

57. How can Hive avoid mapreduce while processing the query?

When you perform a "select * from <tablename>", Hive fetches the whole data from file as a FetchTask rather than a mapreduce task which just dumps the data as it is without doing anything on it. This is similar to "hadoop dfs -text <filename>"

However, while using "select <column> from <tablename>", Hive requires a map-reduce job since it needs to extract the 'column' from each row by parsing it from the file it loads.

58. What is view and indexing in hive?

The goal of Hive indexing is to improve the speed of query lookup on certain columns of a table. Without an index, queries with predicates like

'WHERE tab1. col1 = 10' loads the entire table or partition and process all the rows;

Views are generated based on user requirements. You can save any result set data as a view. The usage of view in Hive is same as that of the view in SQL. It is a standard RDBMS concept. We can execute all DML operations on a view.

Creating a View

You can create a view at the time of executing a SELECT statement.

The syntax is as follows:

CREATE VIEW [IF NOT EXISTS] view_name [(column_name [COMMENT column_comment], ...)] [COMMENT table_comment] AS SELECT ...

An Index is nothing but a pointer on a particular column of a table.

Creating an index means creating a pointer on a particular column of a table.

CREATE INDEX inedx_salary ON TABLE employee(salary) AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler';

59. Can the name of a view be the same as the name of a hive table?

No, we cannot use same name for a table and view in Hive. So we have to select a unique name for a view in Hive.

60. What types of costs are associated in creating indexes on hive tables?

There is a processing cost in arranging the values of the column on which index is created since Indexes occupies space.

61. Give the command to see the indexes on a table.

SHOW INDEX ON table_name;

62. Explain the process to access subdirectories recursively in Hive queries.

We can use following commands in Hive to recursively access sub-directories:

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

Set hive.mapred.supports.subdirectories=true;

Once above options are set to true, Hive will recursively access sub-directories of a directory in MapReduce.

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

Query you are reading every column, no filtering is required. Hence no Map phase

*select * FROM TABLE –Q1 And select col1,col2 FROM TABLE –Q12*

To understand the reason, first we need to know what map and reduce phases mean:

Map: Basically a filter which filters and organizes data in sorted order. For e.g. It will filter col1_name, col2_name from a row in the second query. query you are reading every column, no filtering is required. Hence no Map phase.

Reduce: Reduce is just summary operation data across the rows. for e.g. sum of a coloumn!

In both the queries you don't need any summary data. Hence no reducer.

64. What are the uses of Hive Explode?

Explode is a **User Defined Table generating Function**(UDTF) in Hive. It takes an array (or a map) as an input and outputs the elements of the array (or a map) as separate rows. UDTFs can be used in the SELECT expression list and as a part of LATERAL VIEW.

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

There are following ways by which you can connect with the Hive Server:

1. Thrift Client: Using thrift you can call hive commands from a various programming languages e.g. C++, Java, PHP, Python and Ruby.

2. JDBC Driver : It supports the Type 4 (pure Java) JDBC Driver.

3. ODBC Driver: It supports ODBC protocol.

66. Can the default location of a managed table be changed in Hive?

Using the LOCATION keyword, we can change the default location of Managed tables while creating the managed table in Hive.

67. What is the Hive ObjectInspector function?

Hive uses ObjectInspector to analyze the internal structure of the row object and also the structure of the individual columns.

ObjectInspector provides a uniform way to access complex objects that can be stored in multiple formats in the memory, including: Instance of a Java class (Thrift or native Java)

68. What is UDF in Hive?

UDF stands for User Defined Functions.

Hive is a powerful tool that allows us to provision sql queries on top of stored data for basic querying and/or analysis. And on top of an already rich set of built-in functions, it allows us to extend its functionality by writing custom functions of our own.

Types of UDFs available in Hive:

1. **UDF** (User Defined Function): Operates on a single row and generates a single output.
2. **UDAF** (User Defined Aggregate Function): Operates on a group of rows and generates a single output. Generally, used with an accompanying ‘group by’ clause.
3. **UDTF** (User Defined Tabular Function): Operates on a single row and generates multiple rows as an output.

69. Write a query to extract data from hdfs to hive.

load data inpath '<hdfs location>' into table <hive table name>;

70. What is TextInputFormat and SequenceFileInputFormat in hive.

TextInputFormat :

TEXTFILE format is a famous input/output format used in **Hadoop**. In Hive if we define a table as TEXTFILE it can load data of from CSV (Comma Separated Values), delimited by Tabs, Spaces, and JSON data. This means fields in each record should be separated by comma or space or tab or it may be JSON(JavaScript Object Notation) data.

By default, if we use TEXTFILE format then each line is considered as a record.

We can create a TEXTFILE format in Hive as follows:

create table table_name (schema of the table) row format delimited fields terminated by ',' | stored as TEXTFILE.

At the end, we need to specify the type of **file format**. If we do not specify anything it will consider the file format as TEXTFILE format.

The TEXTFILE input and TEXTFILE output format are present in the Hadoop package as shown below:

org.apache.hadoop.mapred.TextInputFormat org.apache.hadoop.mapred.TextOutputFormat

SequenceFileInputFormat:

We know that Hadoop's performance is drawn out when we work with a small number of files with big size rather than a large number of files with small size. If the size of a file is smaller than the typical block size in Hadoop, we consider it as a small file. Due to this, a number of metadata increases which will become an overhead to the NameNode. To solve this problem sequence files are introduced in Hadoop. Sequence files act as a container to store the small files.

Sequence files are flat files consisting of binary key-value pairs. When Hive converts queries to MapReduce jobs, it decides on the appropriate key-value pairs to be used for a given record. Sequence files are in the binary format which can be split and the main use of these files is to club two or more smaller files and make them as a one sequence file.

In Hive we can create a sequence file by specifying STORED AS SEQUENCEFILE in the end of a CREATE TABLE statement.

There are three types of sequence files:

- Uncompressed key/value records.
- Record compressed key/value records - only 'values' are compressed here
- Block compressed key/value records - both keys and values are collected in 'blocks' separately and compressed. The size of the 'block' is configurable.

Hive has its own SEQUENCEFILE reader and SEQUENCEFILE writer libraries for reading and writing through sequence files.

In Hive we can create a sequence file format as follows:

create table table_name (schema of the table) row format delimited fileds terminated by ',' | stored as SEQUENCEFILE

Hive uses the SEQUENCEFILE input and output formats from the following packages:

org.apache.hadoop.mapred.SequenceFileInputFormat

org.apache.hadoop.hive.ql.io.HiveSequenceFileOutputFormat

71. **How can you prevent a large job from running for a long time in a hive?**

This can be achieved by setting the MapReduce jobs to execute in strict mode

set hive.mapred.mode=strict;

The strict mode ensures that the queries on partitioned tables cannot execute without defining a WHERE clause.

72. When do we use explode in Hive?

We need to use Explode in Hive to convert complex data types into desired table formats. explode UDTF basically emits all the elements in an array into multiple rows.

73. Can Hive process any type of data formats? Why? Explain in very detail.

HiveQL handles structured data only.

By default, Hive has derby database to store the data in it. We can configure Hive with MySQL database. As mentioned, HiveQL can handle only structured data. Data is eventually stored in files.

Hive supports several file formats:

- Text File
- SequenceFile
- RCFile
- Avro Files
- ORC Files
- Parquet
- Custom INPUTFORMAT and OUTPUTFORMAT

The hive.default.fileformat configuration parameter determines the format to use if it is not specified in a CREATE TABLE or ALTER TABLE statement. Text file is the parameter's default value.

Hive Text File Format

Hive Text file format is a default storage format. You can use the text format to interchange the data with other client application. The text file format is very common most of the applications. Data is stored in lines, with each line being a record. Each lines are terminated by a newline character (\n).

The text format is simple plane file format. You can use the compression (*BZIP2*) on the text file to reduce the storage spaces.

Hive Sequence File Format

Sequence files are Hadoop flat files which stores values in binary key-value pairs. The sequence files are in binary format and these files are able to split. The main advantages of using sequence file is to merge two or more files into one file.

Hive RC File Format

RCFile is row columnar file format. This is another form of Hive file format which offers high row level compression rates. If you have requirement to perform multiple rows at a time then you can use RCFile format.

The RCFile are very much similar to the sequence file format. This file format also stores the data as key-value pairs.

Hive AVRO File Format

AVRO is open source project that provides data serialization and data exchange services for Hadoop. You can exchange data between Hadoop ecosystem and program written in any programming languages. Avro is one of the popular file format in Big Data Hadoop based applications.

Hive ORC File Format

The **ORC file** stands for Optimized Row Columnar file format. The ORC file format provides a highly efficient way to store data in Hive table. This file system was actually designed to overcome limitations of the other Hive file formats. The Use of ORC files improves performance when Hive is reading, writing, and processing data from large tables.

Hive Parquet File Format

Parquet is a column-oriented binary file format. The parquet is highly efficient for the types of largescale queries. Parquet is especially good for queries scanning particular columns within a particular table. The Parquet table uses compression Snappy, gzip; currently Snappy by default.

74. Whenever we run a Hive query, a new metastore_db is created. Why?

Whenever you run the hive in embedded mode, it creates the local metastore. And before creating the metastore it looks whether metastore already exist or not. This property is defined in configuration file hive-site.xml. Property is “javax.jdo.option.ConnectionURL” with default value “jdbc:derby:;databaseName=metastore_db;create=true”. So to change the behavior change the location to absolute path, so metastore will be used from that location.

75. Can we change the data type of a column in a hive table? Write a complete query.

ALTER TABLE table_name CHANGE column_name column_name new_datatype;

76. While loading data into a hive table using the LOAD DATA clause, how do you specify it is a hdfs file and not a local file ?

Hive provides us the functionality to load pre-created table entities either from our local file system or from HDFS. The LOAD DATA statement is used to load data into the hive table.

Syntax:

LOAD DATA [LOCAL] INPATH '' [OVERWRITE] INTO TABLE ;

Note: The LOCAL Switch specifies that the data we are loading is available in our Local File System. If the LOCAL switch is not used, the hive will consider the location as an HDFS path location. The OVERWRITE switch allows us to overwrite the table data.

77. What is the precedence order in Hive configuration?

In Hive we can use following precedence order to set the configurable properties.

- Hive SET command has the highest priority
- -hiveconf option from Hive Command Line
- hive-site.xml file
- hive-default.xml file
- hadoop-site.xml file
- hadoop-default.xml file

78. Which interface is used for accessing the Hive metastore?

WebHCat API web interface can be used for Hive commands. It is a REST API that allows applications to make HTTP requests to access the Hive metastore (HCatalog DDL). It also enables users to create and queue Hive queries and commands.

79. Is it possible to compress json in the Hive external table ?

Just gzip your files and put them as is (*.gz) into the table location.

80. What is the difference between local and remote metastores?

- **Local Metastore**: Here metastore service still runs in the same JVM as Hive but it connects to a database running in a separate process either on same machine or on a remote machine.

- **Remote Metastore**: Metastore runs in its own separate JVM not on hive service JVM.

81. What is the purpose of archiving tables in Hive?

Due to the design of HDFS, the number of files in the filesystem directly affects the memory consumption in the namenode. While normally not a problem for small clusters, memory usage may hit the limits of accessible memory on a single machine when there are >50-100 million files. In such situations, it is advantageous to have as few files as possible.

The use of Hadoop Archives is one approach to reducing the number of files in partitions. Hive has built-in support to convert files in existing partitions to a Hadoop Archive (HAR) so that a partition that may once have consisted of 100's of files can occupy just ~3 files (depending on settings). However, the trade-off is that queries may be slower due to the additional overhead in reading from the HAR.

82. What is DBPROPERTY in Hive?

DBPROPERTIES – Optional but used to specify any properties of database in the form of (key, value) separated pairs.

```
CREATE DATABASE IF NOT EXISTS test_db
        COMMENT "Test Database created for tutorial"
        WITH DBPROPERTIES(
                'Date' = '2014-12-03',
                'Creator' = 'Bala G',
                'Email' = 'bala@somewhere.com'
                );
```

83. Differentiate between local mode and MapReduce mode in Hive.

MapReduce mode:

In MapReduce mode, Hive script is executed on Hadoop cluster. The Hive scripts are converted into MapReduce jobs and then executed on Hadoop cluster (hdfs)

Local mode:

In this mode, Hive script runs on a Single machine without the need of Hadoop cluster or hdfs. Local mode is used for development purpose to see how the script would behave in an actual environment.