Interview Questions

1. **What are the NoSQL Databases?**
   NoSQL or "not only SQL," is an alternative to traditional relational databases where data is placed in tables and data schema is designed before the database is built. NoSQL databases are useful for working with large sets of distributed data. HBase is one such database. You can read more about it from the segments of the module “NoSQL Databases and Apache HBase”.

2. **When to use NoSQL Databases?**
   Following are some reasons for using a NoSQL database-
   ● NoSQL data stores are distributed and are efficient in storing and handling huge volumes of data. Based on the targeted use cases, every NoSQL database has its data model for storing data.
   ● NoSQL data stores provide scalability, i.e. additional storage can be easily created by simply adding nodes to the cluster.
   ● Unlike RDBMS, NoSQL data stores are flexible and do not restrict themselves to a fixed schema. Hence, NoSQL data stores can adapt to changes in the schema of data Dynamically.

3. **What is a column family in HBase?**
   A column family in HBase is a set of related columns. You can read about the HBase Data model in the HBase module.

4. **What is the difference between RDBMS and HBase?**

HBase	RDBMS
HBase is a distributed database	RDBMS generally are single node databases
HBase has Column-oriented table schema.	RDBMS offers Row-oriented table schema.
Good for both semi-structured data and structured data.	Good for only structured data.
HBase offers Flexible schema meaning we can add columns on the Fly without any overhead.	RDBMS systems have Fixed schema, although you can modify schema but the operation(alter) is very expensive.
Its columnar schema makes it Good with sparse tables, Which results in a memory efficient database.	With the fixed schema, the space of each element in the row is fixed which results in the allocation of space for null values in the row

	too, hence Not optimized for sparse tables.
Apart from basic DML and lookup commands, HBase doesn't provide a native query language.	RDBMS provides a powerful query language known as SQL.
The columnar table schema results in Wide tables.	RDBMS tables are Narrow because of the row based schema.
Since HBase doesn't have a query engine analytics operations such as Joins are generally performed using MR, which are not optimized. [compare to RDBMS joins]	Native SQL provides optimized for operations i.e. Joins(small, fast ones) etc
HBase has tight integration with MR for data processing applications	No integration with MR.
HBase can easily be scaled Horizontally By adding more nodes.	Hard to share and scale.
Provides Consistency and Partition tolerance.	Being a single node system offers Consistency, Availability. [can say partition tolerance too].

5. **What are Bloom filters?**
   A Bloom filter is an efficient data structure that is used to test whether an element is a member of a set. It is both time and space efficient implementation for searching an element.

6. **Differentiate between Major and Minor Compaction in HBase.**

Minor Compaction	Major Compaction
In Minor Compaction, HBase picks only some of the smaller HFiles and rewrites them into a few larger HFiles	Major Compaction, all HFiles of a store are picked and rewritten into a single large HFile.
Minor compaction helps in reducing the number of HFiles by rewriting smaller HFiles into fewer but larger HFiles, performing a merge sort.	During Major compaction, the values with tombstone markers (deleted data) and expired values(whose TTL is over) are removed from the HFiles.
Less resource intensive hence scheduled more frequently.	Resource intensive and major compactions are scheduled when the load

	the server is minimal.

7. **What is the CAP theorem?**

The CAP theorem: For a distributed system it is impossible to simultaneously provide more than two out of the three guarantees - Consistency, Availability and Partition tolerance.

- Consistency: It guarantees that every node in the distributed system returns the same, most successful, recent write.
- Availability: When every request receives a response without the guarantee that it contains the most recent write.
- Partition tolerance: When the system continues to function and upholds its guarantees in spite of network partitions.

8. **How deletions are handled inside HBase?**

Delete is a special type UpDate in HBase, where the values for which the delete request is submitted are not deleted immediately. Rather these values are masked by assigning a tombstone marker to them. Every request to read these values(with tombstone markers) returns null to the client, which gives the client the impression that the values are already deleted(Consistency).

The reason why HBase does this, is because HFiles are immutable(Recall: HDFS doesn't allow modifying data of a file). All the values with the tombstone marker are permanently removed during the next Major Compaction.

There are three types of tombstone markers:

- Version Delete Marker: which is used to mark a single version of a column value.
- Column Delete Marker: Marks all versions of a column.
- Family Delete Marker: Marks all versions of all columns for a column family.

Finally, during the next Major compaction, the values with tombstone markers (deleted data) along with expired values(whose TTL is over) are removed from the HBase.

9. **What are the limitations of HBase?**

Following are some of the limitation of HBase-

● HBase allows random and fast lookups on top of HDFS thus making it a very resource intensive database; hence it requires regular maintenance.

● HBase doesn’t support secondary indexes. In case you want to search from more than one field or other than Row key, scan performance would be very slow. To handle such scenarios efficiently, MapReduce framework or Apache Phoenix can be used.

● Unlike RDBMS, HBase supports only one default sort per table, i.e., w.r.t the row key.

● It doesn't support SQL functions like join, group by etc., these functionalities can be provided by integrating it with Apache Phoenix or by Map-Reduce.