**HBASE INTERVIEW QUESTIONS**

**Important Links:-**

<https://data-flair.training/blogs/hbase-interview-questions/>

<https://www.guru99.com/hbase-interview-questions.html>

<https://www.edureka.co/blog/interview-questions/hbase-interview-questions/>

<https://intellipaat.com/blog/interview-question/hbase-interview-questions/>

<https://www.tutorialspoint.com/hbase/hbase_interview_questions.htm>

<https://mindmajix.com/hbase-interview-questions>

<https://www.dezyre.com/article/hbase-interview-questions-and-answers-for-2017/281>

<https://www.whizlabs.com/blog/hbase-interview-questions/>

<https://www.educba.com/hbase-interview-questions/>

**Explain what is Hbase?**

Hbase is a column-oriented database management system which runs on top of HDFS (Hadoop Distribute File System). Hbase is not a relational data store, and it does not support structured query language like SQL.In Hbase, a master node regulates the cluster and region servers to store portions of the tables and operates the work on the data.

**2) Explain why to use Hbase?**

* High capacity storage system
* Distributed design to cater large tables
* Column-Oriented Stores
* Horizontally Scalable
* High performance & Availability
* Base goal of Hbase is millions of columns, thousands of versions and billions of rows
* Unlike HDFS (Hadoop Distribute File System), it supports random real time CRUD operations

**3) Mention what are the key components of Hbase?**

* **Zookeeper:** It does the co-ordination work between client and Hbase Master
* **Hbase Master:** Hbase Master monitors the Region Server
* **RegionServer:** RegionServer monitors the Region
* **Region:** It contains in memory data store(MemStore) and Hfile.
* **Catalog Tables:** Catalog tables consist of ROOT and META

**4) Explain what does Hbase consists of?**

* Hbase consists of a set of tables
* And each table contains rows and columns like traditional database
* Each table must contain an element defined as a Primary Key
* Hbase column denotes an attribute of an object

**5) Mention how many operational commands in Hbase?**

Operational command in Hbases is about five types

* Get
* Put
* Delete
* Scan
* Increment

**6) Explain what is WAL and Hlog in Hbase?**

WAL (Write Ahead Log) is similar to MySQL BIN log; it records all the changes occur in data. It is a standard sequence file by[Hadoop](https://www.guru99.com/bigdata-tutorials.html)and it stores HLogkey’s.  These keys consist of a sequential number as well as actual data and are used to replay not yet persisted data after a server crash. So, in case of server failure WAL work as a life-line and retrieves the lost data’s.

**7) When you should use Hbase?**

* **Data size is huge:** When you have tons and millions of records to operate
* **Complete Redesign:** When you are moving RDBMS to Hbase, you consider it as a complete re-design then mere just changing the ports
* **SQL-Less commands:** You have several features like transactions; inner joins, typed columns, etc.
* **Infrastructure Investment:** You need to have enough cluster for Hbase to be really useful

**8) In Hbase what is column families?**

Column families comprise the basic unit of physical storage in Hbase to which features like compressions are applied.

**9) Explain what is the row key?**

Row key is defined by the application. As the combined key is pre-fixed by the rowkey, it enables the application to define the desired sort order. It also allows logical grouping of cells and make sure that all cells with the same rowkey are co-located on the same server.

**10) Explain deletion in Hbase? Mention what are the three types of tombstone markers in Hbase?**

When you delete the cell in Hbase, the data is not actually deleted but a tombstone marker is set, making the deleted cells invisible.  Hbase deleted are actually removed during compactions.

Three types of tombstone markers are there:

* Version delete marker: For deletion, it marks a single version of a column
* Column delete marker: For deletion, it marks all the versions of a column
* Family delete marker: For deletion, it marks of all column for a column family

**11) Explain how does Hbase actually delete a row?**

In Hbase, whatever you write will be stored from RAM to disk, these disk writes are immutable barring compaction. During deletion process in Hbase, major compaction process delete marker while minor compactions don’t. In normal deletes, it results in a delete tombstone marker- these delete data they represent are removed during compaction.

Also, if you delete data and add more data, but with an earlier timestamp than the tombstone timestamp, further **Gets** may be masked by the delete/tombstone marker and hence you will not receive the inserted value until after the major compaction.

**12) Explain what happens if you alter the block size of a column family on an already occupied database?**

When you alter the block size of the column family, the new data occupies the new block size while the old data remains within the old block size. During data compaction, old data will take the new block size.  New files as they are flushed, have a new block size whereas existing data will continue to be read correctly. All data should be transformed to the new block size, after the next major compaction.

**13) Mention the difference between Hbase and Relational Database?**

|  |  |
| --- | --- |
| **Hbase** | **Relational Database** |
| * It is schema-less * It is a column-oriented data store * It is used to store de-normalized data * It contains sparsely populated tables * Automated partitioning is done in Hbase | * It is a schema based database * It is a row-oriented data store * It is used to store normalized data * It contains thin tables * There is no such provision or built-in support for partitioning |

**14) What is HBaseFsck class?**

There is a tool name called back is available in HBase, which is implemented by the HBaseFsck class. It offers several command-line switches that influence its behavior.

**15) What are the main key structures of HBase?**

Row key and Column key are the two most important key structures using in HBase

**16) Discuss how you can use filters in Apache HBase**

Filters In HBase Shell. It was introduced in Apache HBase 0.92 which helps you to conduct server-side filtering for accessing HBase over HBase shell or thrift.

**17) HBase support syntax structure like SQL yes or No?**

No, unfortunately, SQL support for HBase is not available currently. However, by using Apache Phoenix, we can retrieve data from HBase through SQL queries.

**18) What is the meaning of compaction in HBase?**

At the time of heavy incoming writes, it is impossible to achieve optimal performance by having one file per store. HBase helps you to combines all these HFiles to reduce the number of disk seeds for every read. This process is known as for as Compaction in HBase.

**19) How will you implement joins in HBase?**

HBase, not support joins directly but uses MapReduce jobs join queries can be implemented by retrieving data with the help of different HBase tables.

**20) Explain JMX concerning HBSE**

Java Management Extensions or JMX is an export status of Java applications is the standard for them.

**21) What is the use of MasterServer?**

Master sever helps you to assign a region to the region server as well. It also helps you to handle the load balancing we use the MasterServer.

**22) Define the Term Thrift**

Apache Thrift is written in C++. It provides schema compilers for various programming languages like C++, Perl, PHP, Python, Ruby, and more.

**23) Why use HColumnDescriptor class?**

The detail regarding column family such as compression settings, Number of versions, are stored .in HColumnDescriptor.

**24) What is a cell in HBase?**

A cell in HBase is the smallest unit of an HBase table. It helps you to holds a piece of data in the form of a tuple {row, column, version}

**25) What is a Bloom filter?**

HBase supports Bloom Filter helps you to improve the overall throughput of the cluster. An HBase Bloom Filter is a space-efficient mechanism to test whether a HFile includes certain row or row-col cell.

**26) Tell me about the types of HBase Operations?**

Ans. Two types of HBase Operations are:

* Read Operation
* Write Operation

**27) What is the use of HBase HMaster?**

Main responsibilities of a master are:

1. Coordinating the region servers
2. Admin functions

**30) Name the filter which accepts the page size as the parameter in HBase**

A filter named PageFilter accepts the page size as the parameter.

**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.

**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.

**Compare HBase vs HDFS.**

***a. Built on:***

* HBase

Basically,  it is built on top of the HDFS.

* **HDFS**

Whereas, for storing large files, it is suitable.  
***b. lookups***

* HBase

For larger tables, basically, it offers fast lookups.

* HDFS

However, HDFS does not offer fast lookups.  
***c. Latency***

* HBase

It provides low latency access.

* HDFS

And, it provides high latency batch processing.

**25. Compare HBase vs RDBMS.**

Below given is the feature wise comparison of [**HBase vs RDBMS**](https://data-flair.training/blogs/hbase-vs-rdbms/)**:**

***a. Structure***

* HBase

Structure of HBase is schema-less.

* [**RDBMS**](https://data-flair.training/blogs/sql-rdbms/)

Well, RDBMS is governed by its schema only.  
***b. Scalability***

* HBase

Basically, for wide tables, it is built and also it is horizontally scalable.

* RDBMS

But, RDBMS is thin and built for small tables also it is Hard to scale.  
***c. Transaction***

* HBase

No transactions possible in HBase.

* RDBMS

RDBMS is transactional.

**26. Any 3 Features of HBase.**

Some [**features of HBase**](https://data-flair.training/blogs/features-of-hbase/) are:

***a. Consistency***  
For high-speed requirements, we can use it due to its consistent reads and writes.

***b. Atomic read and write***  
At the time of one read or write process, different processes are prevented from performing any read or write operations that’s why it is named as Atomic read and write.

***c. Sharding***  
HBase offers automatic and manual splitting of regions into smaller subregions when it reaches a threshold size.

**30. Explain HBase Meta Table?**

Ans. A special HBase Catalog table is META table. Mainly, it holds the location of the regions in the cluster.

* Also, it keeps a list of all regions in the system.
* .META. table’s structure is :  
  Key: region start key, region id  
  Values: RegionServer

**What is NoSQL?**

Ans. “NoSQL” database means the database isn’t an RDBMS which supports SQL as its primary access language. HBase is a type of NoSQL Database.  
However, we can say, HBase is really more a “Data Store” since it lacks many of the features we find in an RDBMS. For example, secondary indexes, triggers, typed columns, and advanced query languages, etc.

**4. Explain about the data model operations in HBase.**

**Ans.** Data model operations are:  
**Put Method** – It helps to store data in HBase.  
**Get Method** – In HBase, it helps to retrieve data which is stored.  
**Delete Method**– It helps to delete the data from HBase tables.  
**Scan Method** –This operation helps to iterate over the data with larger key ranges or the entire table.

**7. Is it possible to iterate through the rows of HBase table in reverse order?**

Ans. No.

**9. Suppose that your data is stored in collections, for instance, some binary data, message data or metadata is all keyed on the same value. Will you use HBase for this?**

Ans. Definitely yes. Since, whenever key-based access to data is required for storing and retrieving, it is ideal to use HBase.  
[**Let’s revise HBase Admin API**](https://data-flair.training/blogs/hbase-admin-api/)

**10. Assume that an HBase table Student is disabled. So,  how to access the student table once it is disabled, by using Scan command?**

Ans. With the help of Scan command, any HBase table that is disabled cannot be accessed.

**12. Explain the various table design approaches in HBase.**

Ans. There are two [**HBase table**](https://data-flair.training/blogs/commands-in-hbase/) design approaches we can use. Tall-Narrow and Flat-Wide.  
The major difference between flat-wide and tall-narrow approach is when there are less number of rows and a large number of columns we use a tall-narrow approach or when there is less number of columns and a large number of rows, we use flat-wide approach.

**13. What is the best practice on deciding the number of column families for HBase table?**

Ans. Since every column family in HBase is stored as a single file, it is ideal not to exceed the number of columns families per HBase table by 15. Hence, to read and merge multiple files, we will need a large number of columns families.

**14. How will you implement joins in HBase?**

Ans. By using MapReduce jobs join queries, HBase does support joins. Basically, it helps to retrieve data from various HBase tables.

**15. Define MapReduce.**

Ans. To solve the problem of processing in excess of terabytes of data in a scalable way, we use [**MapReduce**](https://data-flair.training/blogs/hadoop-mapreduce-tutorial/) as a process.

**19. What is the use of tools command?**

Ans. To list the HBase surgery tools we use this command.

**20. What is the use of shutdown command?**

Ans. In order to shut down the cluster, we use it.

**24. How to delete the table with the HBase shell?**

Ans.  In order to delete a table, first, disable it then delete it.

**25. What is the use of InputFormat in MapReduce process?**

Ans. In[**MapReduce**](https://data-flair.training/blogs/hadoop-mapreduce-performance-tuning-best-practices/) process, InputFormat the input data. Afterward, it returns a RecordReader instance which defines the classes of the key and value objects and provides a next() method which we use to iterate over each input record.

**27. Which type of data HBase can store?**

Ans. Any type of data that we can convert into the bytes, [**HBase**](https://hbase.apache.org/) can store.

**28. Features of HBase.**

Ans. Some best [**features of HBase**](https://data-flair.training/blogs/features-of-hbase/):

1. Consistency
2. Atomic read and write
3. Sharding
4. High Availability
5. Client API
6. Scalability
7. Hadoop/HDFS integration
8. Distributed storage
9. Data Replication
10. Failover support and load sharing
11. API support
12. MapReduce support
13. Back up support
14. Sorted row keys
15. Real-time processing
16. faster lookups
17. Type of Data
18. Schema-less
19. High throughput
20. Easy to use Java API for client access
21. Thrift gateway and a REST-ful Web service

**29. What is Client API?**

Ans. In order to perform CRUD operations on HBase tables, we use[**Java**](https://data-flair.training/blogs/java-tutorial/)**client API** for HBase. Because of Java Native API of HBase, it offers programmatic access to **DML** (Data Manipulation Language).

[**Learn more about HBase Client API**](https://data-flair.training/blogs/hbase-client-api/)

**30. Explain MemStore?**

Ans. On defining**MemStore**, updates in memory(sorted KeyValues) are stored in the MemStore. And, also the Data which consists of sorted key/values is stored in an HFile.

**When do we need to disable a table in Hbase?**

In Hbase a table is disabled to allow it to be modified or change its settings. .When a table is disabled it cannot be accessed through the scan command.

**Give a command to check if a table is disabled.**

Hbase > is\_disabled “table name”

**What does the following table do?**

hbase > disable\_all 'p.\*'

The command will disable all the table starting with the letter p

**What is the lower bound of versions in Hbase?**

The lower bound of versions indicates the minimum number of versions to be stored in HBase for a column. For example If the value is set to 3 then three latest version will be maintained and the older ones will be removed.

**What is TTL (Time to live) in Hbase?**

TTL is a data retention technique using which the version of a cell can be preserved till a specific time period.Once that timestamp is reached the specific version will be removed.

**Does Hbase support table joins?**

Hbase does not support table jons. But using a mapreduce job we can specify join queries to retrieve data from multiple Hbase tables.

**What is a rowkey in Hbase?**

Each row in Hbase is identified by a unique byte of array called row key.

**What are the two ways in which you can access data from Hbase?**

The data in Hbase can be accessed in two ways.

* Using the rowkey and table scan for a range of row key values.
* Using mapreduce in a batch manner.

**What are the two types of table design approach in Hbase?**

They are − (i) Short and Wide (ii) Tall and Thin

**In which scenario should we consider creating a short and wide Hbase table?**

The short and wide table design is considered when there is

* There is a small number of columns
* There is a large number of rows

**In Which scenario should we consider a Tall-thin table design?**

The tall and thin table design is considered when there is

* There is a large number of columns
* There is a small number of rows

**Give a command to store 4 versions in a table rather than the default 3.**

hbase > alter 'tablename', {NAME => 'ColFamily', VERSIONS => 4}

**What does the following command do?**

hbase > alter 'tablename', {NAME => 'colFamily', METHOD => 'delete'}

This command deletes the column family form the table.

**Give the commands to add a new column family “(newcolfamily”) to a table (“tablename”) which has a existing column family(“oldcolfamily”).**

Hbase > disable ‘tablename’

Hbase > alter ‘tablename’ {NAME => ‘oldcolfamily’,NAME=>’newcolfamily’}

Habse > enable ‘tablename’

**What is the Hbase shell command to only 10 records form a table?**

scan 'tablename', {LIMIT=>10,

STARTROW=>"start\_row",

STOPROW=>"stop\_row"}

**What does the following command do?**

major\_compact 'tablename'

Run a major compaction on the table.

**How does Hbase support Bulk data loading?**

There are two main steps to do a data bulk load in Hbase.

* Generate Hbase data file(StoreFile) using a custom mapreduce job) from the data source. The StoreFile is created in Hbase internal format which can be efficiently loaded.
* The prepared file is imported using another tool like comletebulkload to import data into a running cluster. Each file gets loaded to one specific region.

**How does Hbase provide high availability?**

Hbase uses a feature called region replication. In this feature for each region of a table, there will be multiple replicas that are opened in different RegionServers. The Load Balancer ensures that the region replicas are not co-hosted in the same region servers.

**What are the different Block Caches in Hbase?**

HBase provides two different BlockCache implementations: the default on-heap LruBlockCache and the BucketCache, which is (usually) off-heap.

**How does WAL help when a RegionServer crashes?**

The Write Ahead Log (WAL) records all changes to data in HBase, to file-based storage. if a RegionServer crashes or becomes unavailable before the MemStore is flushed, the WAL ensures that the changes to the data can be replayed.

**Why MultiWAL is needed?**

With a single WAL per RegionServer, the RegionServer must write to the WAL serially, because HDFS files must be sequential. This causes the WAL to be a performance bottleneck.

**In Hbase what is log splitting?**

When a region is edited, the edits in the WAL file which belong to that region need to be replayed. Therefore, edits in the WAL file must be grouped by region so that particular sets can be replayed to regenerate the data in a particular region. The process of grouping the WAL edits by region is called log splitting.

**How can you disable WAL? What is the benefit?**

WAL can be disabled to improve performance bottleneck.

This is done by calling the Hbase client field Mutation.writeToWAL(false).

**When do we do manula Region splitting?**

The manual region splitting is done we have an unexpected hotspot in your table because of many clients querying the same table.

**What is a Hbase Store?**

A Habse Store hosts a MemStore and 0 or more StoreFiles (HFiles). A Store corresponds to a column family for a table for a given region.

**Which file in Hbase is designed after the SSTable file of BigTable?**

The HFile in Habse which stores the Actual data(not metadata) is designed after the SSTable file of BigTable.

**Why do we pre-create empty regions?**

Tables in HBase are initially created with one region by default. Then for bulk imports, all clients will write to the same region until it is large enough to split and become distributed across the cluster. So empty regions are created to make this process faster.

**What is hotspotting in Hbase?**

Hotspotting is asituation when a large amount of client traffic is directed at one node, or only a few nodes, of a cluster. This traffic may represent reads, writes, or other operations. This traffic overwhelms the single machine responsible for hosting that region, causing performance degradation and potentially leading to region unavailability.

**What are the approaches to avoid hotspotting?**

Hotspotting can be avoided or minimized by distributing the rowkeys across multiple regions. The different techniques to do this is salting and Hashing.

**Why should we try to minimize the row name and column name sizes in Hbase?**

In Hbase values are always freighted with their coordinates; as a cell value passes through the system, it’ll be accompanied by its row, column name, and timestamp. If the rows and column names are large, especially compared to the size of the cell value, then indices that are kept on HBase storefiles (StoreFile (HFile)) to facilitate random access may end up occupying large chunks of the HBase allotted RAM than the data itself because the cell value coordinates are large.

**What is the scope of a rowkey in Habse?**

Rowkeys are scoped to ColumnFamilies. The same rowkey could exist in each ColumnFamily that exists in a table without collision.

**What is the information stored in hbase:meta table?**

The Hbase:meta tables stores details of region in the system in the following format.

info:regioninfo (serialized HRegionInfo instance for this region)

info:server (server:port of the RegionServer containing this region)

info:serverstartcode (start-time of the RegionServer process containing this region)

**What is a Namespace in Hbase?**

A Namespace is a logical grouping of tables . It is similar to a database object in a Relational database system.

**How do we get the complete list of columns that exist in a column Family?**

The complete list of columns in a column family can be obtained only querying all the rows for that column family.

**When the records are fetched form a Hbase tables, in which order are the sorted?**

The records fetched form Hbase are always sorted in the order of rowkey-> column Family-> column qualifier-> tiestamp.

##### **What exactly do you know about the Hbase and what exactly do you find different in it as compare to others platforms in its class?**

It is one of the best available Database Management systems which are based on Hadoop. As compared to others, it is actually not a relational DBMS and it cannot be considered when it comes to any structured query language. All the clusters are generally managed by a master node in this approach and this is exactly what that makes it simply the best.

##### **3. What would be the best reasons to prefer Hbase as the DBMS according to you?**

One of the best things about Hbase is it is scalable in all the aspects and modules. The users can simply make sure of catering a very large number of tables in a short time period. In addition to this, it has a vast support available for all the CRUD operations. It is capable to store more data and can manage the same simply. Also, the stores are column oriented and there are a very large number of rows and column available that enable users to keep the pace up all the time.

##### **5. Tell a few scenarios when you will consider HBase?**

When there is a need to shift an entire database, this approach is generally opted. In addition to this, during the data operations which are large to handle, Hbase can be consider. Moreover, when there are a lot of features such as inner joins and transactions maintenance need to be used frequently, the Hbase can be considered easily.

##### **6.** **How can you say that the Hbase is capable to offer high availability?**

There is a special feature known as region replication. There are several replicas available that define the entire region in a table. It is the load balancer in the Hbase which simply make sure that the replicas are not hosted again and again in the servers with similar regions. This is exactly what that makes sure of the high availability of Hbase all the time.

##### **9. Can you directly delete a call from the HBase?**

No, it is not possible in most of the cases. When the users actually do so, the cells get invisible and remain present in the server in the form of a tombstone marker. They are generally removed by the compactions periods. The direct deleting doesn’t work in most of the cases.

##### **10. What is the significance of Data management according to you?**

Generally, organizations have to work with bulk data. When the same is structured or managed, it is easy to utilize or to deploy it for any task. Of course, it cut down the overall time period required to accomplish a task if it is well-managed. The users are always free to keep up the pace simply with the structured or the properly managed data. There are a lot of other reasons too that matters and always let the users assure error-free outcomes.

## 11. What do you know about the set of tables in the Hbase?

They consist of a long series of the rows and columns. It seems quite similar to that of a traditional database.  There is one element in every table and the same is called as the primary key. The columns generally denote an attribute of the concerned objects.

##### **12. Can you tell any one basic condition to be fulfilled when it comes to getting the best out from the Hbase?**

The users must make sure that there are enough nodes and clusters so that Hbase can perform its task reliably and easily. With more nodes, more efficiency can simply be assured by the users.

##### **13. Is Hbase an OS independent approach?**

Yes, it is totally independent of the operating system and the users are free to consider it on Windows, Linux, Unix etc. the only basic requirement is it should have a Java support installed on it.

## 14. Where the compression feature can be applied in the HBase?

It is generally done when the users need to use any of the feature related to the physical storage assessment.  There are no complex restrictions that need to be fulfilled for this.

##### 15. You might have used a relational database, can you tell some of the major differences you noticed in it as compared to HBase?

Well, the first difference is Hbase is not based on schema whereas relation database is. The automated partitioning can easily be done in Hbase while relational database lack this feature. There are more tables in Hbase than in the relational database. Also, it is a row-oriented data store while Hbase is a column-oriented data store.

##### 16. How can you make sure of logical grouping of cells in the HBase?

This can be assured by paying attention to the Row key. The users are free to make sure that all the cells with similar row key can be located to each other and have the presence on a similar server. If the need of defining is realized, the Row key can be considered.

##### 17. Tell something about the procedure of deleting a row in HBase?

The best part about the Hbase is everything written on the RAM gets stored automatically on the Disk. There are some barring compaction remains present with the same. These compactions can be categorized into two parts and they are major and minor. The major compaction can easily delete the files while there is a restriction on the minor ones for the same.

##### 18. What do you know about an Hfile and with whom it is actually related to in an HBase?

It is basically a defined storage format for the HBase and generally, it is related to a column family. There is no strict upper limit on them in the column families. The users can easily deploy an Hfile for storing data that belong to different families.

###### 19. Is it possible for the users to alter the column family’s block size?

Yes, it is possible. Generally, when this is done by the users, the fresh version of data simply occupies the new block size without affecting the old data. The entire old data consume the new one during the compaction.

##### **28. Why Hbase is a schema-less database?**

This is because the users need not to worry about defining the data prior of the time. You only need to define the column family name and nothing else. This makes the Hbase a schema-less database.

##### **29. What is the procedure to write data in the Hbase?**

During any modification or change in the data, it is first sent to a commit log which is also known as WAL. It is after this the data is stored in the memory. In case the data exceed beyond the defined limit, the same is transferred to the disk as an Hfile. The users are free to discard the commit logs and can proceed with the stored data.

##### **30. Define TTL in Hbase?**

It is basically a technique that is useful when it comes to data retention. It is possible for the users to preserve the version of a cell for a defined time period. The same get deleted automatically upon the completion of such a time.

**What do you understand by CAP theorem and which features of CAP theorem does HBase follow?**

CAP stands for Consistency, Availability and Partition Tolerance.

* Consistency –At a given point of time, all nodes in a cluster will be able to see the same set of data.
* Availability- Every request generates a response, regardless of whether it is a success or a failure.
* Partition Tolerance – System continues to work even if there is failure of part of the system or intermittent message loss.
* HBase is a column oriented databases providing features like partition tolerance and consistency.

**Name few other popular column oriented databases like HBase.**

CouchDB, MongoDB, Cassandra

**10) Suppose that your data is stored in collections, for instance some binary data, message data or metadata is all keyed on the same value. Will you use HBase for this?**

Yes, it is ideal to use HBase whenever key based access to data is required for storing and retrieving.

**11)  Assume that an HBase table Student is disabled. Can you tell me how will I access the student table using Scan command once it is disabled?**

Any HBase table that is disabled cannot be accessed using Scan command.

**14) Which one would you recommend for HBase table design approach – tall-narrow or flat wide?**

There are several factors to be considered when deciding between flat-wide (millions of columns and limited keys) and tall-narrow (millions of keys with limited columns), however, a tall-narrow approach is often recommended because of the following reasons –

* Under extreme scenarios, a flat-wide approach might end up with a single row per region, resulting in poor performance and scalability.
* Table scans are often efficient over multiple reads. Considering that only a subset of the row data will be required, tall-narrow table design approach will provide better performance over flat-wide approach.

## ****HBase Interview Questions for Experienced****

1) How will you design the HBase Schema for Twitter data?

2) You want to fetch data from HBase to create a REST API. Which is the best way to read HBase data using a Spark Job or a Java program?

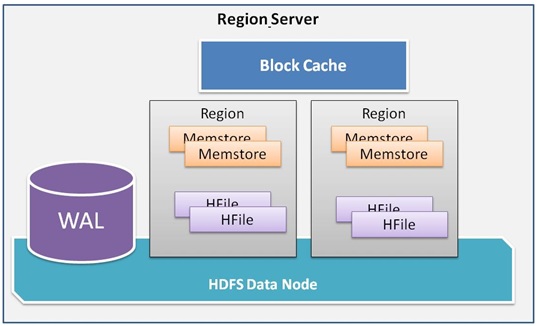
3) Design a HBase table for many to many relationship between two entities, for example employee and department.

4) Explain an example that demonstrates good de-normalization in HBase with consistency.

5) Should your HBase and MapReduce cluster be the same or they should be run on separate clusters?

#### What is BlockCache?

**Answer:** HBase BlockCache is another data storage used in HBase. It is used to keep the most used data in JVM heap. The main purpose of such data storage is to provide access to data from HFiles to avoid disk reading. Each column family in HBase has its own BlockCache. Similarly, each Block in BlockCache represents the unit of data whereas an Hfile is a sequence of blocks with an index over those blocks.



## **HBase Advanced InterviewQuestions and Answers**

#### How HBase handles the write failure?

**Answer:** Failure is common in a large distributed system like HBase. However, we can safeguard against data failure using HBase Write Ahead Log (WAL). Every server which belongs to the HBase cluster maintains a WAL to record the changes that happen in HBase data. Unless a new entry in WAL is written against each write, it will not be considered as successful. Furthermore, HBase is supported by the Hadoop Distributed Filesystem (HDFS). Hence, if HBase goes down, the unflushed data from the MemStore will be recovered by using the WAL.

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

**Answer:**

1.MemStore: It is the first place to check if there is any pending modification in the system.  
2.BlockCache: This is to verify whether the block has been recently accessed.  
3.HFiles:Relevant HFiles on disk. .

#### What is the difference between Cassandra and HBase?

**Answer:**  
Both HBase and Cassandra have [distributed NoSQL database](https://www.educba.com/nosql-interview-questions/) for Big Data from [the Hadoop ecosystem](https://www.educba.com/hadoop-ecosystem/). Both built for different use cases.  
The HBase has kind of master-slave architecture with several components like Zookeeper, Namenode, HBase Master(Hmaster), and Data Nodes etc. [Cassandra treats all nodes](https://www.educba.com/cassandra-interview-questions/) as masters which means all nodes are equal and perform all functions.  
HBase is optimized for reads, write is only happening to the master node and has strong consistency for reading after write. Cassandra has excellent single-row read performance if eventual consistency is selected.  
Hbase does not natively support secondary indexes, Cassandra supports secondary indexes on column families where the column name is known.  
Initially, Hbase is created in Google and they named it BigTable. Even now APIs of Bigtable and HBase is compatible. Origin of Cassandra is from a paper for DynamoDB which is NoSQL database from AWS. Let us move to the next HBase Interview Questions.

#### 5. What is Compaction? Explain different types of it.

**Answer:**  
HBase stores all the received operations into its memstore memory area. When the memory buffer is full, it is flushed to disk. Because this can create many small files in HDFS, from time to time, HBase can select files to be compacted together into a bigger one. A compaction is called Minor when [HBase elects only some of the HFiles](https://www.educba.com/what-is-hbase/) to be compacted but not all. In a Major compaction, all the files are elected to be compacted together. A major compaction works like a minor one except that the delete markers can be removed after they are applied to all the related cells and all extra versions of the same cell will also be dropped.

#### How HBase version data?

**Answer:**  
When a piece of data is inserted/updated/deleted HBase will create a new version for that column.Actual deletion is happening only while compaction. If a particular cell exceeded a number of versions allowed, extra versions will be dropped during compaction

#### What is a difference between getting and Scan?

**Answer:**  
Get will return an only single row from [Hbase table based on row key](https://www.educba.com/hbase-create-table/) given. Scan command returns set of rows depending upon given search condition. Usually get is faster than scan. So should prefer to use that if possible.

#### 8. What happens when deleting a row?

**Answer:**  
At the time of deletion, command data is not physically deleted from file system instead make invisible by setting a marker. Physical deletion happens during a compaction  
Column, Version, and Family Delete Markers are three different types of markers which mark deletion of a Column, Version of Column and Column Family respectively.

#### What are Hlog and HFile?

**Answer:**  
HLog is the write-ahead log file, also known as WAL and HFile is the real data storage file. Data is first written to the write-ahead log file and also written in MemStore.Once MemStore is full, the contents of the MemStore are flushed to the disk into HFiles.