## What is the difference between memstore and hfile in HBase?

Ans: **MemStore**: It is the write cache. It stores new data which has not yet been written to disk. It is sorted before writing to disk. There is one MemStore per column family per region. The MemStore stores updates in memory as sorted KeyValues, the same as it would be stored in an HFile. The updates are sorted per column family in memstore. When the MemStore accumulates enough data, the entire sorted set is written to a new HFile in HDFS.

**Hfiles** store the rows as sorted KeyValues on disk. HBase uses multiple HFiles per column family, which contain the actual cells, or KeyValue instances. These files are created over time as KeyValue edits, sorted in the MemStores are flushed as files to disk. When the MemStore accumulates enough data, the entire sorted KeyValue set is written to a new HFile in HDFS. This is a sequential write. It is very fast, as it avoids moving the disk drive head. It also saves the last written sequence number so the system knows what was persisted so far. The highest sequence number is stored as a meta field in each HFile, to reflect where persisting has ended and where to continue. On region startup, the sequence number is read, and the highest is used as the sequence number for new edits.

1. **Describe compactions in HBase.**

Ans: There are 2 types of compaction – major and minor compaction.

**HBase Minor Compaction:** HBase will automatically pick some smaller HFiles and rewrite them into fewer bigger Hfiles. This process is called minor compaction. Minor compaction reduces the number of storage files by rewriting smaller files into fewer but larger ones, performing a merge sort.

**HBase Major Compaction:** Major compaction merges and rewrites all the HFiles in a region to one HFile per column family, and in the process, drops deleted or expired cells. A major compaction makes any data files that were remote, due to server failure or load balancing, local to the region server.

1. **What will happen if we do not create a row key while inserting the data?**

Ans: We will get an error - HBase row key cannot be NULL.

1. **How can filters be applied in HBase and what are the benefits?**

Ans: Filters return a subset of results to the client. While this does not reduce server-side IO, it does reduce network bandwidth and reduces the amount of data the client needs to process. Filters can be used from HBase Shell for testing and debugging purposes. HBase filters take zero or more arguments, in parentheses. Where the argument is a string, it is surrounded by single quotes ('string'). Filters can be combined together with logical operators. Some filters take a combination of comparison operators and comparators.

1. **What are the data model operations in hBase?**

Ans: The four primary data model operations are Get, Put, Scan, and Delete.

1. [Get](http://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Get.html) returns attributes for a specified row. Gets are executed via [HTable.get](http://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/HTable.html" \l "get%28org.apache.hadoop.hbase.client.Get%29" \t "_top)
2. Put either adds new rows to a table (if the key is new) or can update existing rows (if the key already exists). Puts are executed via HTable.put (writeBuffer) or HTable.batch (non-writeBuffer).
3. Scan allows iteration over multiple rows for specified attributes.
4. [Delete](http://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Delete.html) removes a row from a table. Deletes are executed via [HTable.delete](http://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/HTable.html" \l "delete%28org.apache.hadoop.hbase.client.Delete%29" \t "_top) . HBase does not modify data in place, and so deletes are handled by creating new markers called *tombstones*. These tombstones, along with the dead values, are cleaned up on major compactions.
5. **How can MapReduce be used with HBase?**

Ans

* Mapreduce will run where the data is and network traffic is used only for result data. If input data is huge but output data is tiny for this task then we should use mapreduce, otherwise we need to move huge data over network which will be in gb or tb. But mapreduce output will be only in mb and we can write it to Hbase. Also mapreduce gives us parallel task execution ability, which we can have in java. YARN creates map tasks according to our Hbase table's split count. So if we need more map task, then we can split our table.
* We should use Hbase With MapReduce for [Bulk Loading Into Hbase.](https://www.youtube.com/watch?v=0x9WgmDS-Ds)

1. **What is regionserver?**

Ans: Region server: It serves data for read and write. For accessing data, client communicates with region servers directly. Region server are the nodes that consists of region that contains all rows in the table. It decided the size of the regions by following the region size thresholds.

Zookeeper: Zookeeper is the coordinator. ZooKeeper is a distributed coordination service to maintain server state in the cluster. Zookeeper maintains which servers are alive and available, and provides server failure notification. Zookeeper uses consensus to guarantee common shared state.