# **ASSIGNMENT 32.1**

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

# **MEMSTORE:**

# MemStore - write buffer where HBase will accumulate all the data in the memory before the permanent write gets starts.

# All the contents will be flushed to disk in the form of HFile .

# Once the MemStore gets filled up it will write to the existing HFile.

# But it will form a new file for every flush.  for each MemStore there will be a column family.

# MemStore size is defined as system-wide property for hbase-site.xml which is called as ***hbase.hregion.memstore.flush.size.***

# **HFILE:**

# HFile - underlying storage format for HBase.

# HFiles belong to a column family and a column family can have multiple HFiles.

# But a single HFile can’t have data for multiple column families.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*2****)* Describe compactions in HBase.**

* [Apache HBase](http://hbase.apache.org/) - distributed data store.
* Which is based on log-structured merge tree.
* Optimal read performance will be used for only one file per store (Column Family).
* Ideally it is not possible during the time of heavy incoming writes.
* HBase will combine HFiles in order to reduce maximum number of disk specially to read.
* And this process is called as compaction.
* Compactions means it will choose some file from single store in a particular region.
* And will combine those.
* All the process have reading the Key pair Values from input files
* Those input will be written out to any of the KeyValues which is not deleted inside time to live (TTL).
* Number of versions will not be violated.
* New combined file which were created will be replaced by input files.
* When any client questions for data then HBase will know about the data from input files which were held by contiguous file on disk .
* one seek will be needed.
* Previously seek for each file is required.

And there are 2 types of compactions and they are:

1.)**Minor compactions -** will combine configurable files into number of smaller HFiles to larger HFile.

2.)**Major compactions -** seeks will combine all HFiles into one a large HFile.

compaction will not cleanup work after a user deletes a record.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

* Hbase - Nosql database as well columnar structure.
* To store data in hbase it will be divide into regions.
* Regions will act as horizontal partitions of the tables.
* Each row inside the table will have a specific start and end key for particular region.
* specify region row key should be must.
* Hbase will provide with random access for a particular record and will modify it.
* It can be done by specifying the row key of that record.
* For deleting all the record markers will be set .
* Than directly deleting the data.
* After major compaction gets over after a period of time the record will be deleted.
* Row key is very important for hbase.
* Mainly will denote the lowest unit of storage.
* Without the row key no operations will be possible on a particular record .

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

HBase will include many filter types and has the ability to group filters and create your own custom filters.

* **KeyOnlyFilter** – have no arguments. Returns the key portion of each key-value pair.
* **FirstKeyOnlyFilter** – have no arguments. Returns the key portion of the first key-value pair.
* **PrefixFilter** – holds single argument, a prefix of a row key. Will returns the key-values which is present inside a row that will start specified row prefix
* **ColumnPrefixFilter** – hold single argument - a column prefix. And will returns those key-values which are present in a column which will starts with specified column prefix.

**MultipleColumnPrefixFilter** – will hold a list of column prefixes. And will returns key-values.

* which are present inside a column that can starts with *any* specified column prefixes.

**ColumnCountGetFilter** – will hold one argument, as a limit and will returns first limit number of the columns into the table.

**PageFilter** – will hold one argument, a page size and will return page size number of rows from the table.

**ColumnPaginationFilter** – will hold two arguments, as a limit and offset.

* And will return limit number of columns after the offset number of columns and will follow this for all the rows.

**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.

**TimeStampsFilter** – will hold a list of timestamps.

* And will return the key-values whose timestamps will match with *any* of the specified timestamps.

**RowFilter** – it will take compare operator and a comparator.

* Will compare each row key with comparator using compare operator.
* if comparison returns true then it will return all key-values in that row.

**FamilyFilter** – will take compare operator as well comparator.

* It will compare each family name with comparator using compare operator
* If comparison returns true, will returns all the key-values in that family.

**QualifierFilter** – will take compare operator and a comparator.

* It will compare each qualifier name with comparator by making use of compare operator.
* if comparison returns true then it will return all key-values in that column.

**ValueFilter** – will take compare operator and a comparator.

* It will compare each value with comparator by making use of compare operator.
* If comparison returns true, then it will return key-value.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

There are 4 primary data model operations and they are

* 1. Get
  2. Put
  3. Scan
  4. Delete

**Get:**

[Get](http://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Get.html) will return attributes for all specified row.

**Put:**

[Put](http://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Put.html) will either add new row to a table when the provided key is new or it can update with a existing rows when a key already exists.

**Scan:**

[Scan](http://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Scan.html) will allow iteration over multiple rows for some specified attributes.

**Delete:**

[Delete](http://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Delete.html) will remove row from a table.

**How can MapReduce be used with HBase?**

* HBase will provides with TableInputFormat to which table scan is provided.
* Will splits rows which results from table scan into regions where those rows reside.
* Map process will pass ImmutableBytesWritable which will contains row key for row .
* As a Result it will contain columns for row.
* Map will process outputs with key/value pair which is based on business logic.
* Reduce will builds its results but will emits row key as an ImmutableBytesWritableand .
* Put command will be used to store results back to HBase.
* All results will be stored in HBase by HBase MapReduce infrastructure. And there will not be any need for Put commands.
* Map reduce will be used to process data which are stored in hbase.
* For data processing special implementation which is called as tableinputformatbase .
* And the subclass will be caleed as tableinputformat.
* Former will implement majority of functionality but will remains abstract.
* Subclass will be a lightweight concrete version of tableinputformat.
* It will be used by many samples as well real mapreduce classes.
* Hbase is provided with tablemapper class.
* will enforces key class 1 as an immutablebyteswritable.
* value class 1 as Result type.

**What is regionserver?**

* HBase Tables are divided horizontally by means of row key range into “Regions.”
* Region will contain all the rows in a table between region’s start key and end key.
* Regions were assigned to nodes in a cluster, called as “Region Servers,”.
* Serve data will be used for reads and writes.
* Region server will serve about 1,000 regions.
* RegionServers are basically software processes called as daemons.
* One can activate store and retrieve data into a HBase Hadoop Database.
* For production environments each RegionServer will be deployed on its own which is dedicated into compute node.
* One can start HBase, and can create a table
* Which is used to store and retrieve data.
* Table will grow beyond the configurable limit.
* HBase system will automatically splits table and will distributes those load to another RegionServer.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_