**Assignment 32.1**

**Problem Statement:**

**Explain in brief**

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

**● Describe compactions in HBase.**

**● List and explain the logical entities in HBase.**

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

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

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

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

**● What is regionserver?**

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

Ans. The MemStore stores updates in memory as sorted KeyValues, the same as it would be stored in an HFile. There is one MemStore per column family. The updates are sorted per column family.

Data is stored in an HFile which contains sorted key/values. 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.

**● Describe compactions in HBase.**

HBase is the database for [Hadoop ecosystem](https://acadgild.com/big-data/big-data-development), where distributed file system is used at the bottom layer, where the data is actually stored in physical form. Within HBase, Cache and RAM are used as a storage area, which gives speed to the ecosystem.

When the storage area of HBase is all most filled with data, it starts creating compressed files, which occupies less memory.

Here are the various processes involved in Minor Compaction:

* Bigger Hfile are created by combining smaller Hfiles.
* Hfile keeps the deleted file with them.
* Increases space in memory, useful to store more data.
* Merge sorting is used in process.

The other way to go around is major compaction

* Data present per column family in one region is accumulated to 1 Hfile.
* Possibilities for traffic congestion.
* The Major compaction process is also known as Write Amplification Process.
* This process must be scheduled at a minimum bandwidth of network I/O.

**● List and explain the logical entities in HBase.**

The Data Model in HBase is designed to accommodate semi-structured data that could vary in field size, data type and columns. Additionally, the layout of the data model makes it easier to partition the data and distribute it across the cluster. The Data Model in HBase is made of different logical components such as Tables, Rows, Column Families, Columns, Cells and Versions.

*Tables* – The HBase Tables are more like logical collection of rows stored in separate partitions called Regions. As shown above, every Region is then served by exactly one Region Server. The figure above shows a representation of a Table.

*Rows* – A row is one instance of data in a table and is identified by a *rowkey*. Rowkeys are unique in a Table and are always treated as a byte[].

*Column Families* – Data in a row are grouped together as Column Families. Each Column Family has one more Columns and these Columns in a family are stored together in a low level storage file known as HFile. Column Families form the basic unit of physical storage to which certain HBase features like compression are applied. Hence it’s important that proper care be taken when designing Column Families in table. *Columns* – A Column Family is made of one or more columns. A Column is identified by a Column Qualifier that consists of the Column Family name concatenated with the Column name using a colon – example: columnfamily:columnname. There can be multiple Columns within a Column Family and Rows within a table can have varied number of Columns.

*Cell* – A Cell stores data and is essentially a unique combination of *rowkey*, Column Family and the Column (Column Qualifier). The data stored in a Cell is called its value and the data type is always treated as byte[].

*Version* – The data stored in a cell is versioned and versions of data are identified by the timestamp. The number of versions of data retained in a column family is configurable and this value by default is 3.

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

  Every row in an HBase table has a unique identifier called its rowkey (Which is equivalent to Primary key in RDBMS, which would be distinct throughout the table). Every interaction you are going to do in database will start with the RowKey only. If there is no row key defined,it will not be possible to access records in the table.

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

Filters in Hbase Shell and Filter Language was introduced in [Apache HBase](https://acadgild.com/blog/apache-hbase-beginners-guide/) zero.92. It permits you to perform server-side filtering when accessing HBase over Thrift or within the HBase shell.

**Benefits of filters**

When reading information from HBase using Get or Scan operations, you’ll be able to use custom filters to return a set of results to the client. This, however, doesn’t reduce server-side IO, it will only cut back network information measure and reduces the amount of information the client has to process. Filters are typically implemented using the Java API, however, are often used from HBase shell for testing and debugging purposes.

**the command for some list filters are available in HBase**

**FirstKeyOnlyFilter**

This filter doesn’t take any arguments. It returns solely the primary key-value from every row.

**Syntax**

FirstKeyOnlyFilter ()

**KeyOnlyFilter**

This filter doesn’t take any arguments. It returns solely the key part of every key-value.

**Syntax**

KeyOnlyFilter ()

**prefixfilter:**

This filter takes one argument as a prefix of a row key. It returns solely those key-values present in the very row that starts with the specified row prefix

**Syntax**

PrefixFilter (<row\_prefix>)

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

There are four main data model operations performed in HBase and they are:

* Create
* Read (Get and Scan)
* Put(update)
* Delete.

1.Create

HBase data model, columns are grouped into column families, which must be defined during table creation. We should at least have one column family. HBase currently does not do well with above three column families so keep the number of column families in your schema low.

**Syntax : create ‘<table-name>’,’<column-family1>’ ,’<column-family2>’ …….**

**Basic commands and the structure**

**Column** – A single field in table

**Column-family** – is group of columns

**Row-key**-Row-key in hbase is mandatory field which serves as the unique identifier for every record.

2.Put

We can insert the data using PUT command.

**Syntax: put ‘<table-name>’,’row-key’,’columnfamily:columnname’,’value’**

Row-key is mandatory field which serves as the unique identifier for every record.

3.Get

We should use GET command to retrieve single record from HBase table,

**Syntax: get ‘<table-name>’,’<row-key>’,’<column-family>’**

4.Scan

**To get the all the records fromtable we should use scan**

Syntax : scan ‘<table -name>’

4.Delete

### ****Deleting entire record from table****

delete  ‘<table-name>’,’<rowkey>’

### ****Deleting specific column from table****

**delete ‘<table-name>’,’row-key’,’columnfamily:columnname’**

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

HBase is used as a data source and a sink (the destination for the output):

1. HBase provides a TableInputFormat, to which you provided a table scan, that splits the rows resulting from the table scan into the regions in which those rows reside.
2. The map process is passed an ImmutableBytesWritable that contains the row key for a row and a Result that contains the columns for that row.
3. The map process outputs its key/value pair based on its business logic in whatever form makes sense to your application.

**● What is regionserver?**

Region servers serve data for reads and writes. When accessing data, clients communicate with HBase RegionServers directly.

A Region Server runs on an HDFS data node and has the following components:

* WAL: Write Ahead Log is a file on the distributed file system. The WAL is used to store new data that hasn't yet been persisted to permanent storage; it is used for recovery in the case of failure.
* BlockCache: is the read cache. It stores frequently read data in memory. Least Recently Used data is evicted when full.
* MemStore: 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.