Assignment 10.1 – HBASE Basics

TASK1

SOLUTION#1

Question 1.What is NoSQL data base?

A NoSQL database is called 'Not only SQL' database which is used when we are dealing with big data , specially where data being processed is unstructured. A NoSQL database provides performance and scalability , but less functionality when compared to traditional RDBMS. A NoSQL database has following properties

1. There is no fixed schema for the database
2. There is no concept of primary/foreign key
3. Table Joins are not needed
4. Data being stored is unstructured (such as pictures, audio, video files)
5. There is need to run the database over a cluster for scaling reasons
6. No complex transaction support provided
7. No constraint in the query are supported (application needs to provide it)
8. Table scans are much faster than traditional RDBMS, but retrieval of complete individual row is slower than RDBMS

A NoSQL database is used in the following cases

1. When the data is unstructured and varied in types (such as text,logs,pictures etc)
2. When the data is required to be stored with high velocity
3. When the database access pattern mostly revolves around querying for many rows at once (table scan) , but only few fields(columns) of those rows

2.How does data get stored in NoSQl database?

Data gets stored in one of the following format in case of NoSQL database

1. Key/Value pair – e.g. Redis
2. Table format - e.g HBASE, Cassandra
3. Document format – e.g. MongoDB

3.What is a column family in HBase?

A column family in HBASE represents a set of column (fields) in the table that are all required to be accessed/updated together. In case of HBASE, all columns belonging to same column family are stored together in hfile so that they can be accessed together. The columns of a columns family must have same access pattern and size requirements. The column family must be specified at the time of HBase table creation

4.How many maximum number of columns can be added to HBase table?

There is no limit of maximum number of columns in HBase table.

5.Why columns are not defined at the time of table creation in HBase?

A HBase table consists only of column families and it is stored per column family basis on disk. The column within a column family become individual records with in that file due to using column name as part of the key of the record. In this way any column for a given column family can be accessed/updated without providing the column name at the time of table creation.

6.How does data get managed in HBase?

The data is managed using the hfiles in HBase. The hfile contains sorted map of rows for a particular column family of a region. Hfile consists blocks where each block contains many rows. hfile also has an index of blocks and row keys so that a particular block is looked up if access to a particular row is required.
The block contains multiple KeyValue corresponding to each row. KeyValue wraps a byte array and takes offsets and lengths into passed array at where to start interpreting the content as KeyValue.
The KeyValue format inside a byte array is:

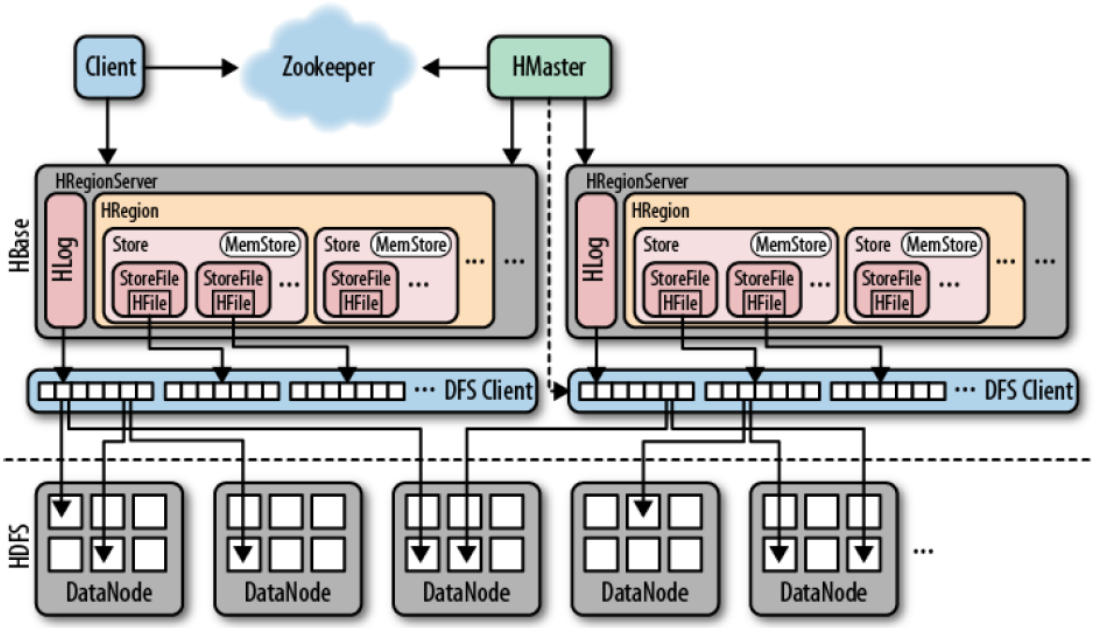
- keylength
- valuelength
- key
- value

The Key is further decomposed as:

- rowlength
- row (i.e., the rowkey)
- columnfamilylength
- columnfamily
- columnqualifier
- timestamp
- keytype (e.g., Put, Delete, DeleteColumn, DeleteFamily)

7.What happens internally when new data gets inserted into HBase table?

Following picture shows the main component of HBase



When client issues an HTable.put(Put) request to the HRegionServer, which hands the details to the matching HRegion instance. The first step is to write the data to the writeahead log (the WAL), represented by the HLog class.The WAL is a standard Hadoop SequenceFile and it stores HLogKey instances. These keys contain a sequential number as well as the actual data and are used to replay not-yet-persisted data after a server crash. Once the data is written to the WAL, it is placed in the MemStore. At the same time, it is checked to see if the MemStore is full and, if so, a flush to disk is requested. The request is served by a separate thread in the HRegionServer, which writes the data to a new HFile located in HDFS. It also saves the last written sequence number so that the system knows what was persisted so far.

Task 2

1. Create an HBase table named 'clicks' with a column family 'hits' such that it should be able to store last 5 values of qualifiers inside 'hits' column family.

2. Add few records in the table and update some of them. Use IP Address as row-key. Scan the table to view if all the previous versions are getting displayed.

Solution#2

```
hbase(main):055:0*
hbase(main):056:0*

# Creating table name 'clicks' and also specifying column family 'hits'. Also
specifying that 5 versions of the row are needed

hbase(main):057:0* create 'clicks',{NAME=>'hits',VERSIONS=>5}
0 row(s) in 2.2450 seconds

=> Hbase::Table - clicks


# Dumping the schema of the table. We see that versions=>5 is set

hbase(main):058:0> describe 'clicks'
Table clicks is ENABLED
clicks
COLUMN FAMILIES DESCRIPTION
{NAME => 'hits', BLOOMFILTER => 'ROW', VERSIONS => '5', IN_MEMORY => 'false',
KEEP_DELETED_CELLS => 'FALSE', DATA_BL
OCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0',
BLOCKCACHE => 'true', BLOCKSIZ
E => '65536', REPLICATION_SCOPE => '0'}
1 row(s) in 0.0230 seconds

# Now we enter 5 records for row id 10.10.10.1 and setting the hits:c value from 1 to
5

hbase(main):059:0> put 'clicks','10.10.10.1','hits:c','1'
0 row(s) in 0.1210 seconds

hbase(main):060:0>  put 'clicks','10.10.10.1','hits:c','2'
0 row(s) in 0.0080 seconds

hbase(main):061:0>  put 'clicks','10.10.10.1','hits:c','3'
0 row(s) in 0.0050 seconds

hbase(main):062:0>  put 'clicks','10.10.10.1','hits:c','4'
0 row(s) in 0.0060 seconds

hbase(main):063:0>  put 'clicks','10.10.10.1','hits:c','5'
0 row(s) in 0.0060 seconds
```

 # Now we enter 5 more records for row id 10.10.10.2 and setting the hits:c value from a to e

```
hbase(main):064:0> put 'clicks','10.10.10.2','hits:c','a'
0 row(s) in 0.0130 seconds

hbase(main):065:0>  put 'clicks','10.10.10.2','hits:c','b'
0 row(s) in 0.0040 seconds

hbase(main):066:0>  put 'clicks','10.10.10.2','hits:c','c'
0 row(s) in 0.0070 seconds

hbase(main):067:0>  put 'clicks','10.10.10.2','hits:c','d'
0 row(s) in 0.0060 seconds

hbase(main):068:0>  put 'clicks','10.10.10.2','hits:c','e'
0 row(s) in 0.0080 seconds

hbase(main):069:0>
```

 # We scan the table and it shows the two records with latest value of column hits:c with value 5 and e

```
hbase(main):070:0* scan 'clicks'
ROW                           COLUMN+CELL
 10.10.10.1                   column=hits:c, timestamp=1533538740960, value=5
 10.10.10.2                   column=hits:c, timestamp=1533538746695, value=e
2 row(s) in 0.0270 seconds
```

 # We dump last 5 records for row id 10.10.10.1 and we see all 5 values

```
hbase(main):071:0> get 'clicks','10.10.10.1',{COLUMNS=>'hits:c',VERSIONS=>5}
COLUMN                        CELL
 hits:c                       timestamp=1533538740960, value=5
 hits:c                       timestamp=1533538740937, value=4
 hits:c                       timestamp=1533538740918, value=3
 hits:c                       timestamp=1533538740898, value=2
 hits:c                       timestamp=1533538740863, value=1
5 row(s) in 0.0360 seconds
```

 # We dump last 2 records for row id 10.10.10.1 and we see last 2 values

```
hbase(main):072:0> get 'clicks','10.10.10.1',{COLUMNS=>'hits:c',VERSIONS=>2}
COLUMN                        CELL
 hits:c                       timestamp=1533538740960, value=5
 hits:c                       timestamp=1533538740937, value=4
2 row(s) in 0.0080 seconds
```

 # We dump last 5 records for row id 10.10.10.2 and we see all 5 values from a to e

```
hbase(main):073:0> get 'clicks','10.10.10.2',{COLUMNS=>'hits:c',VERSIONS=>5}
COLUMN                        CELL
 hits:c                       timestamp=1533538746695, value=e
 hits:c                       timestamp=1533538746673, value=d
 hits:c                       timestamp=1533538746647, value=c
 hits:c                       timestamp=1533538746627, value=b
 hits:c                       timestamp=1533538746598, value=a
5 row(s) in 0.0340 seconds
```

 # We dump last 3 records for row id 10.10.10.2 and we see last 3 values with values c,d and e.

```
hbase(main):074:0> get 'clicks','10.10.10.2',{COLUMNS=>'hits:c',VERSIONS=>3}
COLUMN                          CELL
 hits:c                         timestamp=1533538746695, value=e
 hits:c                         timestamp=1533538746673, value=d
 hits:c                         timestamp=1533538746647, value=c
3 row(s) in 0.0080 seconds

hbase(main):075:0>
```