Assignment 10(Hbase Basics)

Problem Statement :

Task 1 Answer in your own words with example.

1.What is NoSQL database?

NoSql stands for not only SQL. This is unlike traditional relational database which builds a data schema even before the database is built. This handles large sets of distributed data which can also accommodate a wide variety of data model including key-value, document, columnar and graph formats. Eg of Nosql databases are MongoDB, Hbase.

2.How does data get stored in NoSQl database?

Nosql stores the data in a nested format rather than table of rows and dedicated columns.

Eg of normal db:

Fruit table

Record num	Fruit	count	Rate of prefernce	Reviewer Id
1	Apple	8	4	3
2	Orange	20	3	2

Reviewer table

Reviewer Id	Name	Gender
1	John	M
2	Sheela	F

Eg of nosql Db:

{

Name of fruit: Banana,

Count:99,

Rate of preference:10,
Reviewer: {
Name:Dan,
Gender: male
}
}

There are various NoSQL Databases. Each one uses a different method to store data. Some might use column store, some document, some graph, etc., Each database has its own unique characteristics.

3.What is a column family in HBase?

Columns in Apache HBase are grouped into *column families*. All column members of a column family have the same prefix. For example, the columns *courses:history* and *courses:math* are both members of the *courses* column family. The column family prefix must be composed of *printable* characters. Column families must be declared up front at schema definition time whereas columns do not need to be defined at schema time but can be created on the fly while the table is up an running.

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

There are no limitation in Hbase regards to column count, there are some on column size though. The restriction on column count would depend on the file format, ORC or Text file . ORC has configurations for number of rows that are grouped together for an index.

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

As there is no limitation on number of column that can be created and there is no consistency between columns of one column family and the other, It is not mandatory to create a columns during table creation. Column family name is the most important one that needs to be mentioned along with Table name while the table is created. Columns name can be added at later point of time using the column family name and row key.

6.How does data get managed in HBase?

Data in Hbase is organized into tables.Tables are further organized into rows that store data. Each row is identified by a unique row key which does not belong to any data type but is stored as a bytearray. Column families are further used to group data in rows. Column families define the physical structure of data so they are defined upfront and their modification is difficult. Each row in a table has same column families. Data in a column family is addressed using a column qualifier. It is not necessary to specify column qualifiers in advance and there is no consistency requirement between rows. No data types are specified for column qualifiers, as such they are just stored as bytearrays. A unique combination of row key, column family and column qualifier forms a cell. Data contained in a

cell is referred to as cell value. There is no concept of data type when referring to cell values and they are stored as bytearrays. Versioning happens to cell values using a timestamp of when the cell was written.

Eg of data stored in Hbase:

RowKey	Column Family	Column Qualifier	version	value
Lavanya	designation	experience	87653529020	7

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

Insert happens when we use the command put ‘TableName’ ,‘RowKey’ , ‘CF:C1’, V1

Everytime when a data is added it is added with timestamp and it maintains the version using it.

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.

Creating table named clicks with column family hits:

---

create 'clicks','hits'

```
hbase(main):001:0> create 'clicks','hits'
0 row(s) in 1.8690 seconds
```

Viewing the created table schema:

Describe ‘clicks’

```
hbase(main):003:0> describe 'clicks'
Table clicks is ENABLED
clicks
COLUMN FAMILIES DESCRIPTION
{NAME => 'hits', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING
=> 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_
SCOPE => '0'}
1 row(s) in 0.3890 seconds
```

Inserting 5 columns under column family hits:

---

put 'clicks', '192.168.0.8', 'hits:c1', 'v1'

put 'clicks', '192.168.0.8', 'hits:c2', 'v2'

put 'clicks', '192.168.0.8', 'hits:c3', 'v3'

put 'clicks', '192.168.0.8', 'hits:c4', 'v4'

put 'clicks', '192.168.0.8', 'hits:c5', 'v5'

```
hbase(main):006:0>
hbase(main):006:0> put 'clicks', '192.168.0.8', 'hits:c2', 'v2'
0 row(s) in 0.1200 seconds

hbase(main):007:0> put 'clicks', '192.168.0.8', 'hits:c3', 'v3'
0 row(s) in 0.0210 seconds

hbase(main):008:0> put 'clicks', '192.168.0.8', 'hits:c4', 'v4'
0 row(s) in 0.0140 seconds

hbase(main):009:0> put 'clicks', '192.168.0.8', 'hits:c5', 'v5'
0 row(s) in 0.0150 seconds
```

Listing the added columns:

Scan 'clicks'

```
hbase(main):010:0> scan 'clicks'
ROW                           COLUMN+CELL
 192.168.0.8                  column=hits:c1, timestamp=1526382957568, value=v1
 192.168.0.8                  column=hits:c2, timestamp=1526382988210, value=v2
 192.168.0.8                  column=hits:c3, timestamp=1526383000188, value=v3
 192.168.0.8                  column=hits:c4, timestamp=1526383010711, value=v4
 192.168.0.8                  column=hits:c5, timestamp=1526383020666, value=v5
1 row(s) in 0.0500 seconds
```

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.

Listing the added columns using get command

```
hbase(main):011:0> get 'clicks','192.168.0.8'
COLUMN                        CELL
 hits:c1                      timestamp=1526382957568, value=v1
 hits:c2                      timestamp=1526382988210, value=v2
 hits:c3                      timestamp=1526383000188, value=v3
 hits:c4                      timestamp=1526383010711, value=v4
 hits:c5                      timestamp=1526383020666, value=v5
5 row(s) in 0.0580 seconds
```

Updating the version from 2 to 3:

```
hbase(main):004:0> alter 'clicks', {NAME => 'hits', VERSIONS => 3}
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 3.0090 seconds
```

Added 2 more columns in the newer version:

```
hbase(main):007:0> put 'clicks', '192.168.0.8', 'hits:c1', 'v1'
0 row(s) in 0.1220 seconds

hbase(main):008:0> put 'clicks', '192.168.0.8', 'hits:c2', 'v2'
0 row(s) in 0.0080 seconds
```

Listing all the columns including previous version:

```
hbase(main):012:0> scan 'clicks', {VERSIONS => 3}
ROW                                   COLUMN+CELL
 192.168.0.8                          column=hits:c1, timestamp=1526384027825, value=v1
 192.168.0.8                          column=hits:c1, timestamp=1526382957568, value=v1
 192.168.0.8                          column=hits:c2, timestamp=1526384040337, value=v2
 192.168.0.8                          column=hits:c2, timestamp=1526382988210, value=v2
 192.168.0.8                          column=hits:c3, timestamp=1526383000188, value=v3
 192.168.0.8                          column=hits:c4, timestamp=1526383010711, value=v4
 192.168.0.8                          column=hits:c5, timestamp=1526383020666, value=v5
1 row(s) in 0.0230 seconds
```