Assignment Submission- Session 10

Task 1

1.What is NoSQL data base?

Ans: NoSQL stands for Not only SQL. It is an approach to design database that can accommodate a wide variety of data models, including key-value, document, columnar and graph formats. It is an alternative to traditional relational databases in which data is placed in tables and data schema is carefully designed before the database is built. NoSQL databases are especially useful for working with large sets of distributed data and store structured, semi-structured or unstructured data.

2.How does data get stored in NoSQL database?

Ans: Data is stored in NoSQL DB based on the type. NoSQL database are of following types:

Key-value database:

Key-value database are the simplest NoSQL database to use. Every item in the database is stored as a "key" together with its value. The value is a blob that the data store just stores, without caring or knowing what's inside; it's the responsibility of the application to understand what was stored. Since key-value stores always use primary-key access, they generally have great performance and are highly scalable for session management and caching in web applications. The key-value database uses a hash table to store unique keys and pointers with respect to each data value it stores. There are no column type relations in the database; hence, its implementation is easy. Key-value databases give great performance and can be very easily scaled as per business needs.

Document store NoSQL database:

Document store NoSQL databases are similar to key-value databases in that there's a key and a value. Data is stored as a value. Its associated key is the unique identifier for that value. The difference is that, in a document database, the value contains structured or semi-structured data. This structured/semi-structured value is referred to as a document and can be in XML, JSON or BSON format.

Column store NoSQL database:

In column-oriented NoSQL databases, data is stored in cells grouped in columns of data rather than as rows of data. Columns are logically grouped into column families. Column families can contain a virtually unlimited number of columns that can be created at runtime or while defining the schema. Read and write is done using columns rather than rows. Column families are groups of similar data that is usually accessed together.

3.What is a column family in HBase?

Ans: HBase tables are organized by column, rather than by row. The columns are organized in groups called column families. A column family defines shared features to all columns that are created within them, we can think of it as a sub-table in a larger table. When creating a HBase table, we must define the column families before inserting any data. Each column family, can contain a very large number of columns and columns can be added on the fly.

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

Ans: HBase currently doesn't do well with more than 2 or 3 column families, for best performance, we shall try to create schema with 1 column family only.

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

Ans: Columns do not need to be defined at schema time but can be added on the fly while the table is up and running. This gives us the flexibility to add columns as and when required for any specific row key, it is not necessary that under each row key, the column family have the same columns.

6.How does data get managed in HBase?

Ans: In Hbase, data is stored using row key, column family and columns. The row key is unique and is bytearray type and contains column family(kind of sub table). the column family is consistent in all row key, however you can use columns in column family as per requirement. For a particular row key, if under column family cf1, there are columns cf1:c1, cf1:c2.
We may have cf1:c1, cf1:c2, cf1:c3 or cf1:c1 in second row key as required.

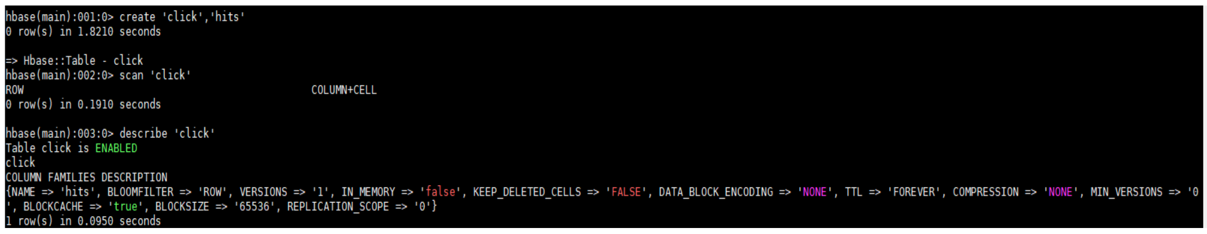
7.What happens internally when new data gets inserted into HBase table?

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.

Added table using:

create 'click','hits'



```
hbase(main):001:0> create 'click','hits'
0 row(s) in 1.8210 seconds

=> Hbase::Table - click
hbase(main):002:0> scan 'click'
ROW                                         COLUMN+CELL
0 row(s) in 0.1910 seconds

hbase(main):003:0> describe 'click'
Table click is ENABLED
click
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 0.0950 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.

Added data in 2 columns using below commands:

put 'click', 'ipAddress1', 'hits:pc', 'lenovo'

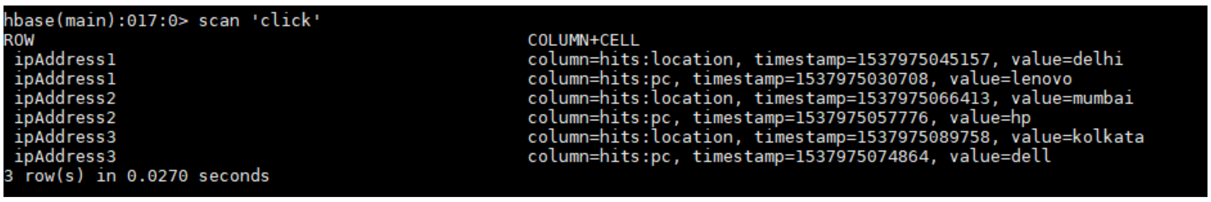
put 'click', 'ipAddress1', 'hits:location', 'delhi'

put 'click', 'ipAddress2', 'hits:pc', 'hp'

put 'click', 'ipAddress2', 'hits:location', 'mumbai'

put 'click', 'ipAddress3', 'hits:pc', 'dell'

put 'click', 'ipAddress3', 'hits:location', 'kolkata'



```
hbase(main):017:0> scan 'click'
ROW                                         COLUMN+CELL
 ipAddress1                                 column=hits:location, timestamp=1537975045157, value=delhi
 ipAddress1                                 column=hits:pc, timestamp=1537975030708, value=lenovo
 ipAddress2                                 column=hits:location, timestamp=1537975066413, value=mumbai
 ipAddress2                                 column=hits:pc, timestamp=1537975057776, value=hp
 ipAddress3                                 column=hits:location, timestamp=1537975089758, value=kolkata
 ipAddress3                                 column=hits:pc, timestamp=1537975074864, value=dell
3 row(s) in 0.0270 seconds
```

Altering table property using:

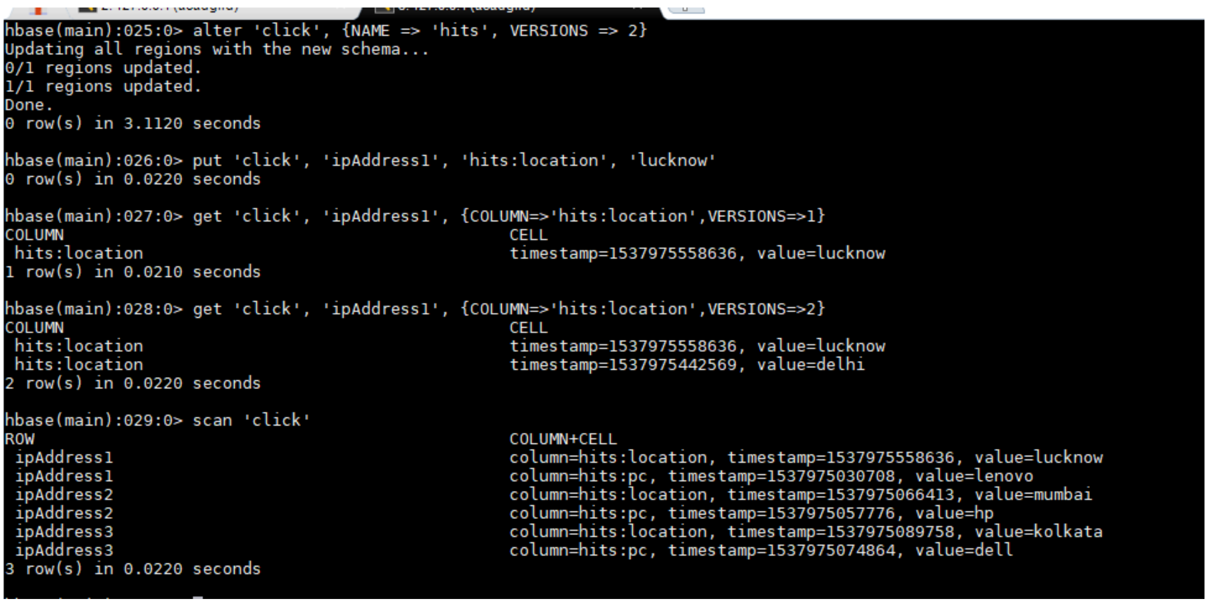
alter 'click', {NAME => 'hits', VERSIONS => 2}

Updating record using:

put 'click', 'ipAddress1', 'hits:location', 'lucknow'

Checking update through version:

get 'click', 'ipAddress1', {COLUMN=>'hits:location',VERSIONS=>2}



```
hbase(main):025:0> alter 'click', {NAME => 'hits', VERSIONS => 2}
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 3.1120 seconds

hbase(main):026:0> put 'click', 'ipAddress1', 'hits:location', 'lucknow'
0 row(s) in 0.0220 seconds

hbase(main):027:0> get 'click', 'ipAddress1', {COLUMN=>'hits:location',VERSIONS=>1}
COLUMN                                             CELL
 hits:location                                     timestamp=1537975558636, value=lucknow
1 row(s) in 0.0210 seconds

hbase(main):028:0> get 'click', 'ipAddress1', {COLUMN=>'hits:location',VERSIONS=>2}
COLUMN                                             CELL
 hits:location                                     timestamp=1537975558636, value=lucknow
 hits:location                                     timestamp=1537975442569, value=delhi
2 row(s) in 0.0220 seconds

hbase(main):029:0> scan 'click'
ROW                                                COLUMN+CELL
 ipAddress1                                        column=hits:location, timestamp=1537975558636, value=lucknow
 ipAddress1                                        column=hits:pc, timestamp=1537975030708, value=lenovo
 ipAddress2                                        column=hits:location, timestamp=1537975066413, value=mumbai
 ipAddress2                                        column=hits:pc, timestamp=1537975057776, value=hp
 ipAddress3                                        column=hits:location, timestamp=1537975089758, value=kolkata
 ipAddress3                                        column=hits:pc, timestamp=1537975074864, value=dell
3 row(s) in 0.0220 seconds
```