

Task 1

Answer in your own words with example.

1. What is NoSQL data base?

Term NoSQL stands for Not Only SQL. NoSQL is a database design that can store variety of data models such as document, key-value, graphical and columnar format. It is an alternative to traditional DB's where data is stored in tables and data schema is designed before a DB is created. NoSQL databases are very much useful for working with large sets of distributed data.

Advantages of NoSQL

- More Scalable
- Superior performance
- Simplicity of design
- It can handle huge volumes of structured, semi-structured, and unstructured data
- Object-oriented programming which is easy to use and flexible
- Scaling to cluster of machines in its architecture instead of expensive, monolithic architecture

Examples: HBase, MongoDB, Cassandra

2. How does data get stored in NoSQL database?

In NoSql Db's data can be stored as Key-value, Document, graph, object, tabular, columnar formats.

In Key-Value model, data is represented as a collection of key-value pairs and is the simplest data model in nosql.

Eg: Oracle NoSql, Dynamo

In document data model the value of each key is a document. This model can contain key – values, key-arrays and nested documents.

Eg: Mongo Db, IBM Domino

In Graph model, data relations are represented in graph model with elements interconnected with some relations between them. Eg: road maps, network topology data

Eg: Neo4J

In columnar model data is stored as rows of transactions, for fast retrieval.

Eg: AWS, HBase

3. What is a column family in HBase?

In HBase data is organized by column, rather than by row in relational model. The columns are organized in groups called column families. When creating a HBase table column families has to be defined at schema definition time before inserting any data. Column families should not be changed often, nor

should there be too many of them, so it is important to think carefully about what column families will be useful for our particular data. Each column family, however, can contain a very large number of columns. Columns are named using the format family:qualifier. All members of a column family will have same prefix.

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

There is no hard limit to number of columns in HBase, we can have more than 1 million columns but Hbase will have performance issues if the number of column families is more than two or three.

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

Column families are specified when a table is created. They should be carefully designed before a table is created since it would be either impossible or difficult to change them later. All columns in a column family are stored and sorted together in the same HFile. Columns can be defined at any time since they are mapped with a column family.

6. How does data get managed in HBase?

HDFS provides scalable and replicated store layer for HBASE. It guarantees that data is never lost by writing changes across a configurable number of physical servers. The data is stored in HFiles, which are ordered immutable key/value maps. Internally, the Hfiles are sequences of blocks with a block index stored at the end. The block index is loaded when Hfile is opened and kept in memory. The default block size is 64KB but it can be changed since it is configurable. HBASE API can be used to access specific values and also scan range of values given a start and end key.

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

When data is updated it is first written to a commit log, called write-ahead log and then it is stored in in-memory memstore. When the data in memory exceeds a given maximum value, it is flushed as an HFile to disk and after that the commit logs are discarded up to the last flushed modification. The system can continue to serve readers and writers without blocking them while it is flushing the memstore to disk. This is done by rolling the memstore in memory where the new empty one is taking the updates and the old full one is transferred into an HFile. At the same time, no sorting or other special processing has to be performed since the data in the memstores is already sorted by keys matching what HFiles represent on disk.

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.***

Create a table using command – **create 'clicks', 'hits'**

Describe command shows all details about the table. By default the number of versions in 1. If any new versions are added, the older ones gets overwritten. To store last 5 values, make number of versions to 5 using command – **alter 'clicks' {NAME => 'hits', VERSIONS => 5}.**

```
MyHadoop 2.6.1.1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

hbase(main):001:0> create 'clicks', 'hits'
0 row(s) in 4.2550 seconds

=> Hbase::Table - clicks
hbase(main):002:0>
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 0.9490 seconds

hbase(main):004:0>
hbase(main):005:0> alter 'clicks', { VERSIONS => 5}
Unknown argument ignored: VERSIONS
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 4.4950 seconds

hbase(main):006:0> alter 'clicks', { VERSIONS => '5'}
Unknown argument ignored: VERSIONS
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 3.9340 seconds

hbase(main):007:0> alter 'clicks', { NAME => 'hits', VERSIONS => '5'}
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 4.1310 seconds

hbase(main):008:0> describe 'clicks'
```

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 a few url names as records using put command. Scan command shows the records added to table.

```
hbase(main):008:0> describe 'clicks'
Table clicks is ENABLED
clicks
COLUMN FAMILIES DESCRIPTION
{NAME => 'hits', BLOOMFILTER => 'ROW', VERSIONS => '5', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
1 row(s) in 0.2420 seconds

hbase(main):009:0>
hbase(main):010:0> put 'clicks', '198.102.0.1', 'hits:url', 'amazon.com'
0 row(s) in 0.7540 seconds

hbase(main):011:0> put 'clicks', '198.102.1.2', 'hits:url', 'flipkart.com'
0 row(s) in 0.0280 seconds

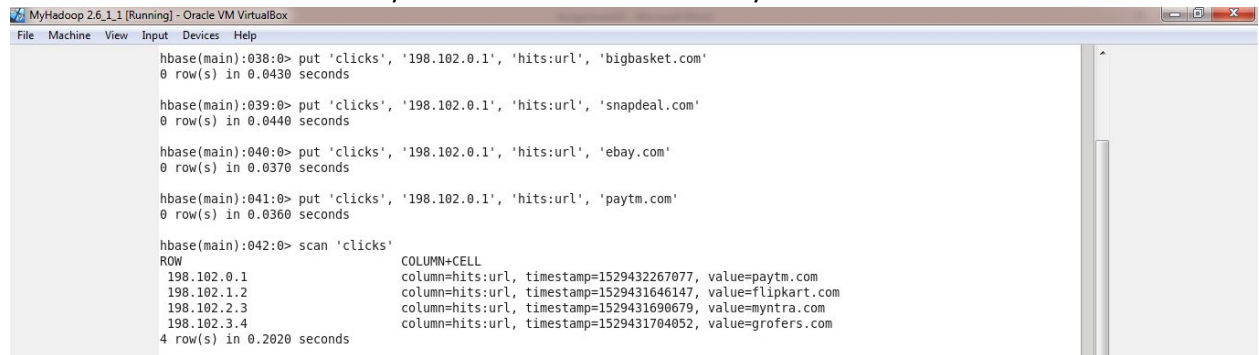
hbase(main):024:0> put 'clicks', '198.102.2.3', 'hits:url', 'myntna.com'
0 row(s) in 0.0960 seconds

hbase(main):025:0> put 'clicks', '198.102.3.4', 'hits:url', 'grofers.com'
0 row(s) in 0.0250 seconds

hbase(main):026:0> scan 'clicks'
ROW COLUMN+CELL
198.102.0.1 column=hits:url, timestamp=1529431286858, value=amazon.com
198.102.1.2 column=hits:url, timestamp=1529431646147, value=flipkart.com
198.102.2.3 column=hits:url, timestamp=1529431690679, value=myntna.com
198.102.3.4 column=hits:url, timestamp=1529431704052, value=grofers.com
4 row(s) in 0.1680 seconds

hbase(main):027:0>
```

Insert 4 more records to row-key 198.102.0.1. Now this row key has 5 records.



```
hbase(main):038:0> put 'clicks', '198.102.0.1', 'hits:url', 'bigbasket.com'
0 row(s) in 0.0430 seconds

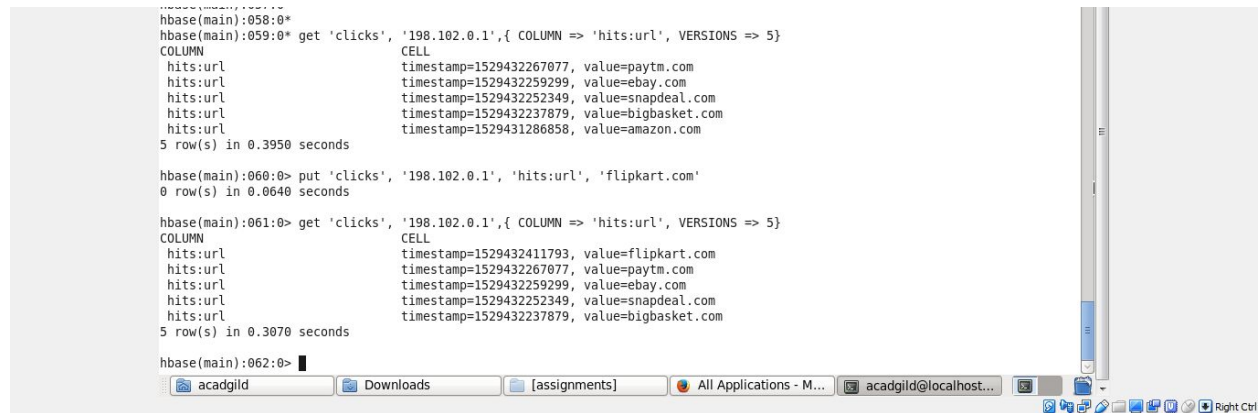
hbase(main):039:0> put 'clicks', '198.102.0.1', 'hits:url', 'snapdeal.com'
0 row(s) in 0.0440 seconds

hbase(main):040:0> put 'clicks', '198.102.0.1', 'hits:url', 'ebay.com'
0 row(s) in 0.0370 seconds

hbase(main):041:0> put 'clicks', '198.102.0.1', 'hits:url', 'paytm.com'
0 row(s) in 0.0360 seconds

hbase(main):042:0> scan 'clicks'
ROW                                COLUMN+CELL
198.102.0.1                        column=hits:url, timestamp=1529432267077, value=paytm.com
198.102.1.2                        column=hits:url, timestamp=1529431646147, value=flipkart.com
198.102.2.3                        column=hits:url, timestamp=1529431690679, value=myntara.com
198.102.3.4                        column=hits:url, timestamp=1529431704052, value=grofers.com
4 row(s) in 0.2020 seconds
```

When new records are added, the older one amazon.com entry got removed and the latest one got added.



```
hbase(main):058:0*
hbase(main):059:0> get 'clicks', '198.102.0.1',{ COLUMN => 'hits:url', VERSIONS => 5}
COLUMN                                CELL
hits:url                              timestamp=1529432267077, value=paytm.com
hits:url                              timestamp=1529432259299, value=ebay.com
hits:url                              timestamp=1529432252349, value=snapdeal.com
hits:url                              timestamp=1529432237879, value=bigbasket.com
hits:url                              timestamp=1529431286858, value=amazon.com
5 row(s) in 0.3950 seconds

hbase(main):060:0> put 'clicks', '198.102.0.1', 'hits:url', 'flipkart.com'
0 row(s) in 0.0640 seconds

hbase(main):061:0> get 'clicks', '198.102.0.1',{ COLUMN => 'hits:url', VERSIONS => 5}
COLUMN                                CELL
hits:url                              timestamp=1529432411793, value=flipkart.com
hits:url                              timestamp=1529432267077, value=paytm.com
hits:url                              timestamp=1529432259299, value=ebay.com
hits:url                              timestamp=1529432252349, value=snapdeal.com
hits:url                              timestamp=1529432237879, value=bigbasket.com
5 row(s) in 0.3070 seconds

hbase(main):062:0>
```