

5. Problem Statement

Task 1

Answer in your own words with example.

1.What is NoSQL data base?

NoSQL is an approach to databases that represents a shift away from traditional relational database management systems (RDBMS). To define NoSQL, it is helpful to start by describing SQL, which is a query language used by RDBMS. Relational databases rely on tables, columns, rows, or schemas to organize and retrieve data. In contrast, NoSQL databases do not rely on these structures and use more flexible data models. NoSQL can mean “not SQL” or “not only SQL.” As RDBMS have increasingly failed to meet the performance, scalability, and flexibility needs that next-generation, data-intensive applications require, NoSQL databases have been adopted by mainstream enterprises. NoSQL is particularly useful for storing unstructured data, which is growing far more rapidly than structured data and does not fit the relational schemas of RDBMS. Common types of unstructured data include: user and session data; chat, messaging, and log data; time series data such as IoT and device data; and large objects such as video and images.

2.How does data get stored in NoSQL database?

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.

You can have a look at the various categories of NoSQL databases here: [NOSQL Databases](#)

I have personally played around with MongoDB for a while now and it has always impressed me. MongoDB is an document store, where data is stored as Key: Value pairs in JSON format.

For Example-:

```
{
```

```
name: "sid",
phone: 1234567890,
address:
{
    street: "1234 Some_XYZ Pkwy" ,
    Apt: 1001,
    City: "Richardson",
    State: "Texas"
}
```

3.What is a column family in HBase?

A column family defines shared features to all columns that are created within them (think of it almost as a sub-table within your larger table). You will notice that HBase columns are composed of a combination of the column family and column qualifier (or column key): 'family:qualifier'

Where the qualifier can be an arbitrary array of bytes, the column family has to be composed of printable characters. Also, on HDFS, the column family is what is stored in human-readable format, example: '/hbase/table/region/<colfamX>'

Remember - all column families must be created up-front where as columns can be added on the fly. Hence understanding the design of your data access pattern is crucial when first building your HBase instance.

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

HBase currently does not do well with anything above two or three column families so keep the number of column families in your schema low. Currently, flushing and compactions are done on a per Region basis so if one column family is carrying the bulk of the data bringing on flushes, the adjacent families will also be flushed though the amount of data they carry is small. When many column families the flushing and compaction interaction can make for a bunch of needless i/o loading (To be addressed by changing flushing and compaction to work on a per column family basis).

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

Column families are part of the schema of the table. You can add them at runtime with an online schema change. But you wouldn't add them dynamically the way that you can dynamically create new "columns" in an HBase table, if that's what you had in mind.

The reason column families are part of the schema and would require a schema change is that they profoundly impact the way the data is stored, both on disk and in memory. Each column family has its own set of HFiles, and its own set of data structures in memory of the RegionServer. It would be pretty expensive to dynamically create or start using new column families.

Column families are only needed when you need to configure differently various parts of a table (for instance you want some columns to have a TTL and others to not expire), or when you want to control the locality of accesses (things accessed together should better be in the same column family if you want good performance, as the cost of operations grows linearly with the number of column families). So, again, because of those specialized reasons, it doesn't make sense to dynamically add new column families at runtime the way you would add regular "columns" within a family.

6.How does data get managed in HBase?

HBase is a column-oriented database that's an open-source implementation of Google's Big Table storage architecture. It can manage structured and semi-structured data and has some built-in features such as scalability, versioning, compression and garbage collection.

Since it uses write-ahead logging and distributed configuration, it can provide fault-tolerance and quick recovery from individual server failures. HBase built on top of Hadoop / HDFS and the data stored in HBase can be manipulated using Hadoop's MapReduce capabilities.

Let's now take a look at how HBase (a column-oriented database) is different from some other data structures and concepts that we are familiar with Row-Oriented vs. Column-Oriented data stores. As shown below, in a row-oriented data store, a row is a unit of data that is read or written together. In a column-oriented data store, the data in a column is stored together and hence quickly retrieved.

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

When the client gives a command to Write, Instruction is directed to Write Ahead Log.

Once the log entry is done, the data to be written is forwarded to MemStore which is actually the RAM of the data node.

Data in the memstore is sorted in the same manner as data in a HFile.

When the memstore accumulates enough data, the entire sorted set is written to a new HFile in HDFS.

Once writing data is completed, ACK (Acknowledgement) is sent to the client as a confirmation of task completed.

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.

Command-:

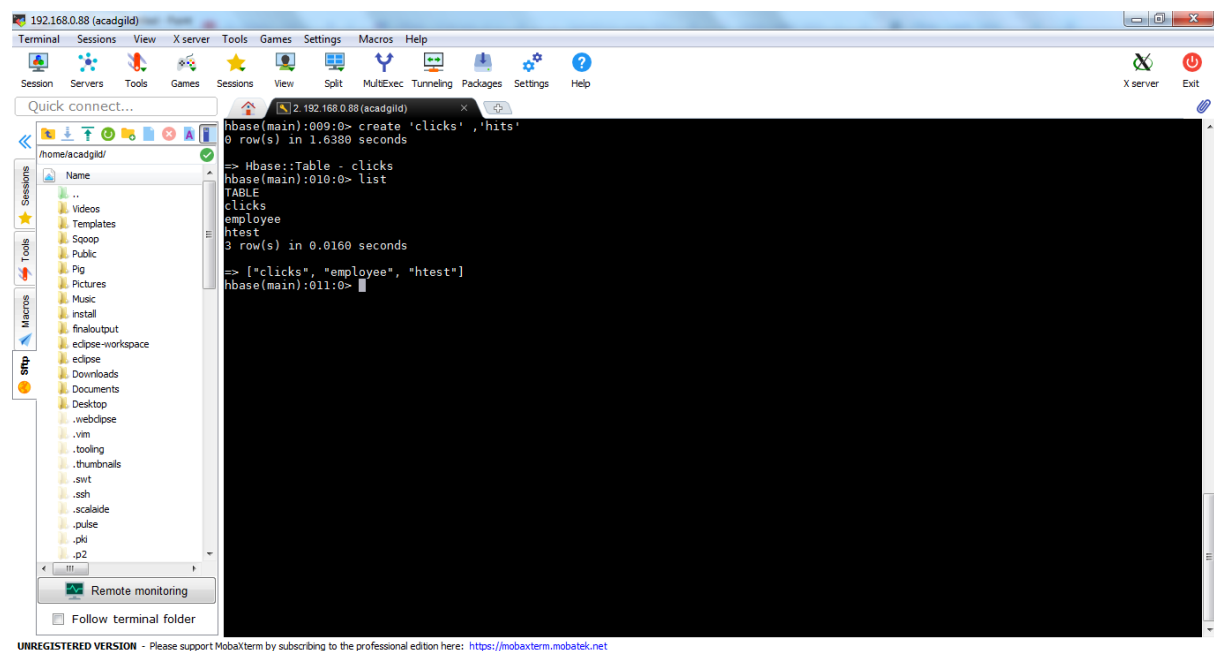
create 'clicks','hits'

→ Command to create table clicks with column family hits

List

-> Command to list tables present in database

Output-:



The screenshot shows a MobaXterm terminal window with the title bar '192.168.0.88 (acadgild)'. The terminal displays the following commands and output:

```
hbase(main):009:0> create 'clicks','hits'
0 row(s) in 1.6380 seconds

=> Hbase::Table - clicks
hbase(main):010:0> list
TABLE
clicks
employee
htest
3 row(s) in 0.0160 seconds

=> ['clicks', 'employee', 'htest']
hbase(main):011:0>
```

On the left side of the terminal, there is a sidebar with a 'Quick connect...' search bar and a list of sessions. The 'Sessions' list includes:

- ...
- Videos
- Templates
- Scoop
- Public
- Pig
- Pictures
- Music
- Install
- finaloutput
- eclipse-workspace
- eclipse
- Downloads
- Documents
- Desktop
- .webclipse
- .vim
- .tooling
- .thumbnails
- .swt
- .ssh
- .scalaide
- .pulse
- .pki
- .p2

At the bottom of the sidebar, there are checkboxes for 'Remote monitoring' and 'Follow terminal folder'.

At the bottom of the terminal window, a message reads: 'UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>'

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.

Command-:

```
put 'clicks','192.168.0.1','hits','1000'
```

```
put 'clicks','192.168.0.2','hits','2000'
```

```
put 'clicks','192.168.0.3','hits','3000'
```

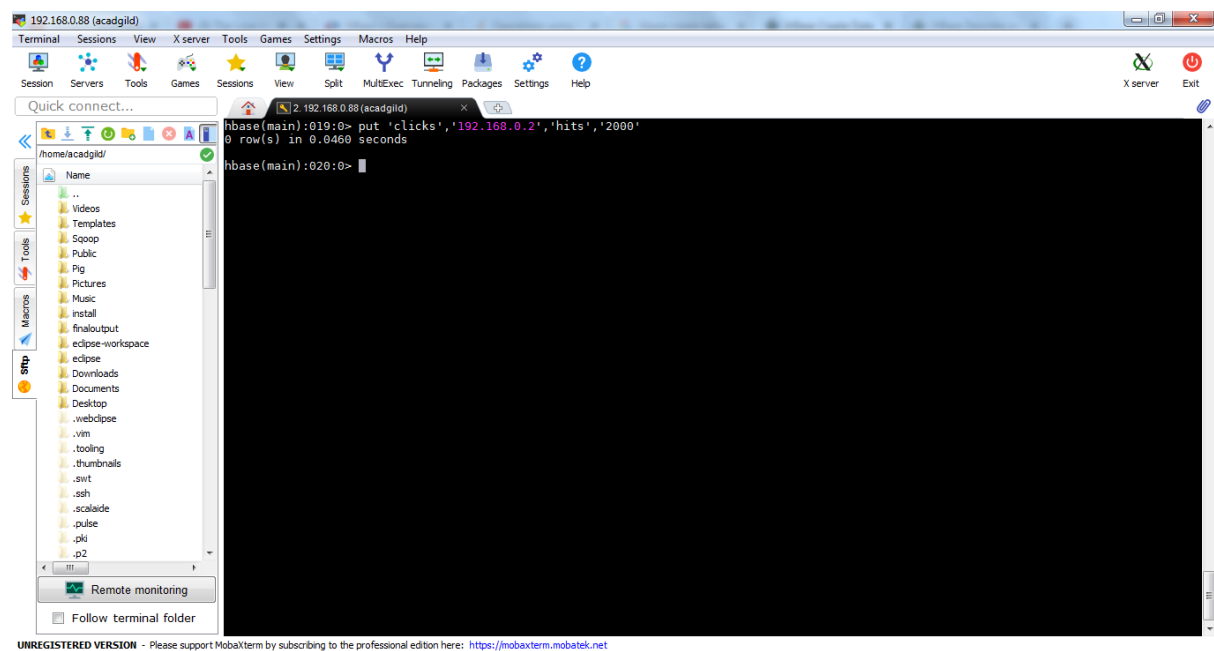
```
put 'clicks','192.168.0.4','hits','4000'
```

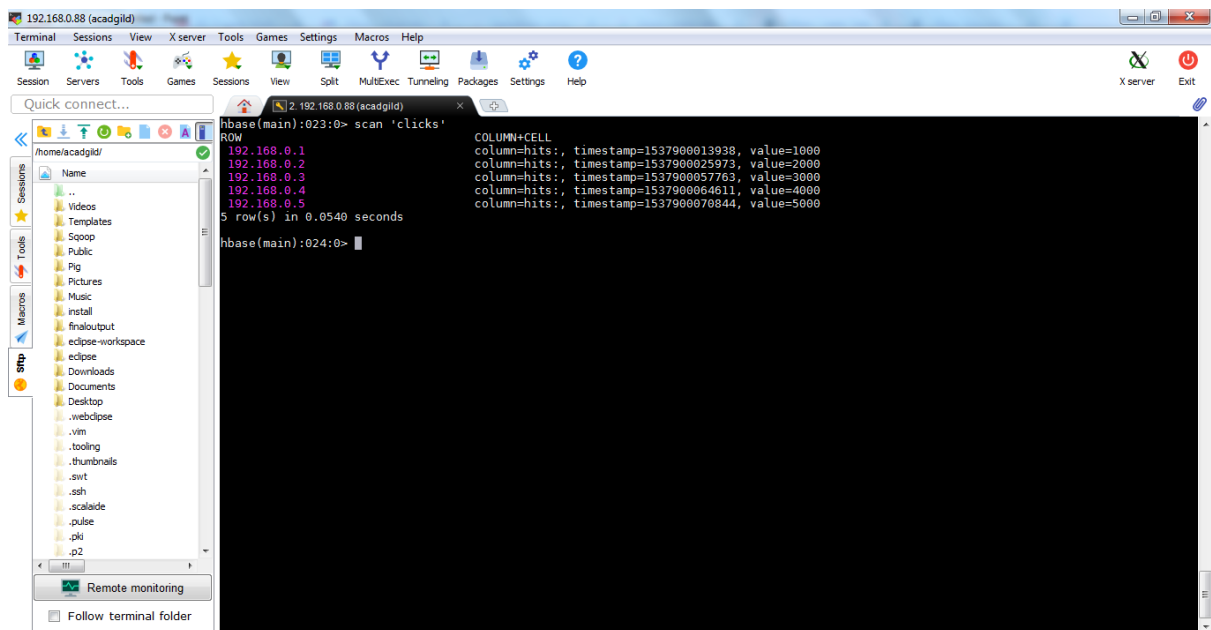
```
put 'clicks','192.168.0.5','hits','5000'
```

->put command is used to insert data into hbase

->scan command is used to get data from database

Output-:





Update few records:-

Command:-

```
put 'clicks','192.168.0.1','hits','6000'
```

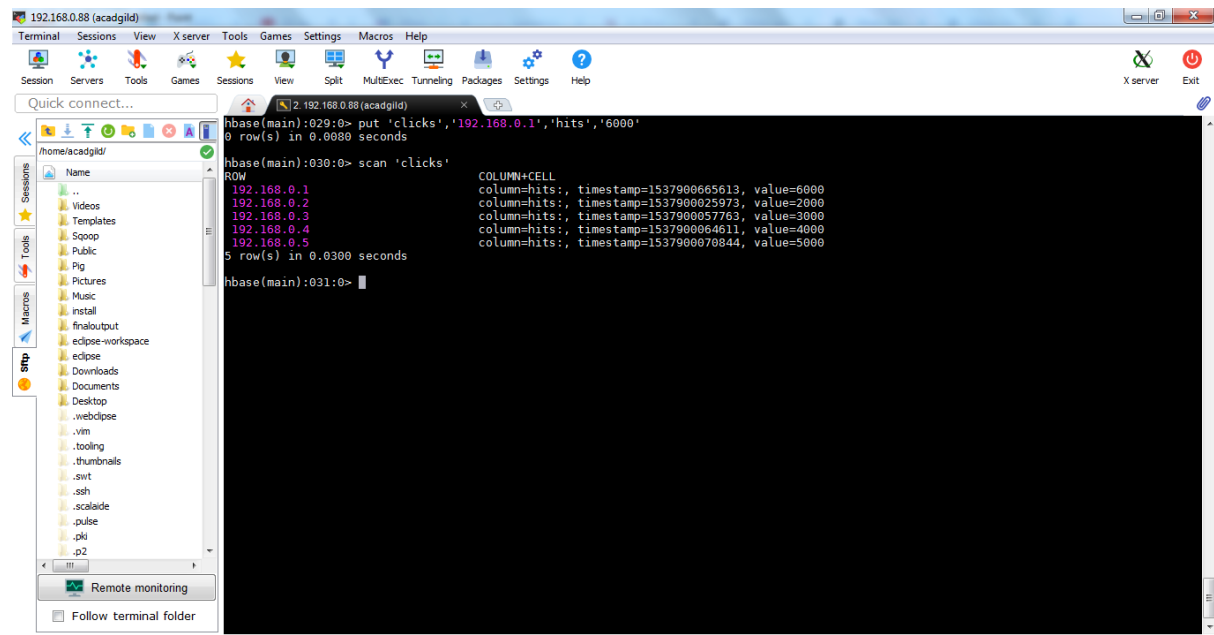
->Updating value in row 1 using row key 192.168.0.1 with value as 6000 and having column family as hits

```
get 'clicks', '192.168.0.1', {COLUMN => 'hits', VERSIONS => 1 ,TIMESTAMP => 1537900013938}
```

```
get 'clicks', '192.168.0.1', {COLUMN => 'hits', VERSIONS => 2 ,TIMESTAMP => 1537900665613}
```


->Get record based on previous and new time stamp and using versions as well, by default hbase maintains upto 3 version

Output:-



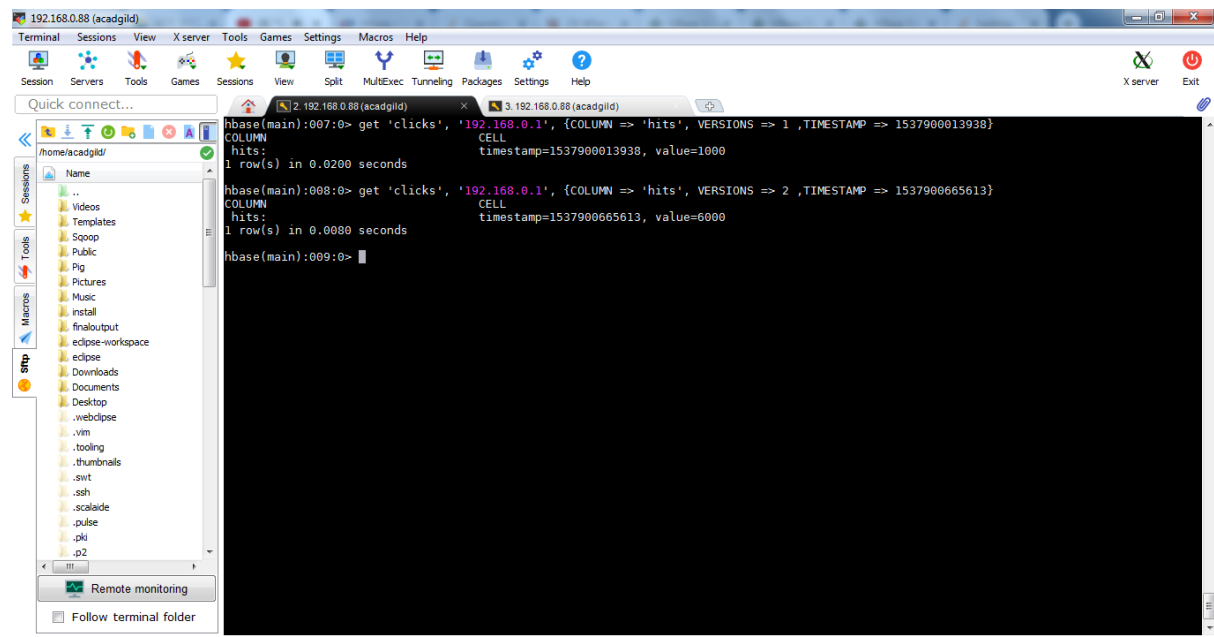
The screenshot shows a MobaXterm terminal window with the title '192.168.0.88 (acadgild)'. The terminal displays the following commands and output:

```
hbase(main):029:0> put 'clicks','192.168.0.1','hits','6000'
0 row(s) in 0.0080 seconds

hbase(main):030:0> scan 'clicks'
ROW                                COLUMN+CELL
192.168.0.1                        column=hits: timestamp=1537900665613, value=6000
192.168.0.2                        column=hits: timestamp=1537900025973, value=2000
192.168.0.3                        column=hits: timestamp=1537900057763, value=3000
192.168.0.4                        column=hits: timestamp=1537900064611, value=4000
192.168.0.5                        column=hits: timestamp=1537900070844, value=5000
5 row(s) in 0.0300 seconds

hbase(main):031:0>
```

At the bottom of the terminal window, there is a message: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>"



The screenshot shows a MobaXterm terminal window with the title '192.168.0.88 (acadgild)'. The terminal displays the following commands and output:

```
hbase(main):007:0> get 'clicks', '192.168.0.1', {COLUMN => 'hits', VERSIONS => 1, TIMESTAMP => 1537900013938}
COLUMN
CELL
hits: timestamp=1537900013938, value=1000
1 row(s) in 0.0200 seconds

hbase(main):008:0> get 'clicks', '192.168.0.1', {COLUMN => 'hits', VERSIONS => 2, TIMESTAMP => 1537900665613}
COLUMN
CELL
hits: timestamp=1537900665613, value=6000
1 row(s) in 0.0080 seconds

hbase(main):009:0>
```

At the bottom of the terminal window, there is a message: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>"