

Assignment 10

Task 1

1.What is No SQL Database?

- NoSQL, which stand for "Not Only SQL," 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 are mainly used for solving Business problems.
- NoSQL doesn't store NULL values
- Most of the NoSQL databases are stored in Binary format
- NoSQL is an approach to database design that can accommodate a wide variety of data models, including key-value, document, columnar and graph formats.
- NoSQL databases are especially useful for working with large sets of distributed data.
- NoSQL database provides ability to process very large volumes of data and quickly distribute that data across computing clusters
- NoSQL is highly scalable as amount of data increases machines can be increases.
- Example of NoSQL are: Cassandra, Mongo DB, HBase
- Features:-
 - **Generic Data Model :-**
Heterogeneous containers, including sets, maps, and arrays
 - **Dynamic type discovery and conversion :-**
NoSQL analytics systems support runtime type identification and conversion so that custom business logic can be used to dictate analytic treatment of variation.
 - **Non-relational and De-normalised :-**
Data is stored in single tables as compared to joining multiple tables.
 - **Commodity hardware :-**
Adding more of the economical servers allows NoSQL databases to scale to handle more data.
 - **Highly distributable :-**
Distributed databases can store and process a set of information on more than one device.

2.How does data get stored in NoSQL database?

Different NoSQL database use different scheme for storing data. Different schemes are as below:

i.Key-value stores

- Key-value stores, or key-value databases, implement a simple data model that pairs a unique key with an associated value.

- Because this model is simple, it can lead to the development of key-value databases, which are extremely performant and highly scalable for session management and caching in web applications.
- Examples include Aero spike, Berkeley DB, MemcacheDB, Redis and Riak.

ii.Document databases

- Document databases, also called document stores, store semi-structured data and descriptions of that data in document format.
- They allow developers to create and update programs without needing to reference master schema.
- Use of document databases has increased along with use of JavaScript and the JavaScript Object Notation (JSON), a data interchange format that has gained wide currency among web application developers, although XML and other data formats can be used as well.
- Document databases are used for content management and mobile application data handling. Couchbase Server, CouchDB, DocumentDB, MarkLogic and MongoDB are examples of document databases.

iii.Wide-column stores

- Wide-column stores organize data tables as columns instead of as rows.
- Wide-column stores can be found both in SQL and NoSQL databases.
- Wide-column stores can query large data volumes faster than conventional relational databases.
- A wide-column data store can be used for recommendation engines, catalogs, fraud detection and other types of data processing. Google BigTable, Cassandra and HBase are examples of wide-column stores.iv.

iv.Graph stores

- Graph data stores organize data as nodes, which are like records in a relational database and edges, which represent connections between nodes.
- Because the graph system stores the relationship between nodes, it can support richer representations of data relationships.
- Also, unlike relational models reliant on strict schemas, the graph data model can evolve over time and use.
- Graph databases are applied in systems that must map relationships, such as reservation systems or customer relationship management.
- Examples of graph databases include AllegroGraph, IBM Graph, Neo4j and Titan.

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 colon character (:) delimits the column family from the .
- The column family prefix must be composed of *printable* characters.

- The qualifying tail, the column family *qualifier*, can be made of any arbitrary bytes.
- 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 conjured on the fly while the table is up and running.
- Physically, all column family members are stored together on the filesystem.
- Because tunings and storage specifications are done at the column family level, it is advised that all column family members have the same general access pattern and size characteristics

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 atop clusters of commodity hardware
- but usually three column families are recommended (not more than three).

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

- HBase uses query-first” schema design; all possible queries should be identified first, and the schema model designed accordingly.
- By not defining columns at the time of creation it provides flexibility to add columns on need basis. Traditional RDBMS database, all the schemas are defined at the beginning.
- This is a constraint when project requirement changes, we need to rework on whole model.
- By keeping flexibility, any new requirements can be easily done without revisiting models

6.How does data get managed in HBase?

- *Data* in *Hbase* is organized into tables.
- Any characters that are legal in file paths are used to name 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.
- HBase stores data in a table-like format with the ability to store billions of rows with millions of columns.
- Columns can be grouped together in “column families” which allows physical distribution of row values onto different cluster nodes.
- HBase group rows into “regions” which define how table data is split over multiple nodes in a cluster.
- If a region gets too large, it is automatically split to share the load across more servers.

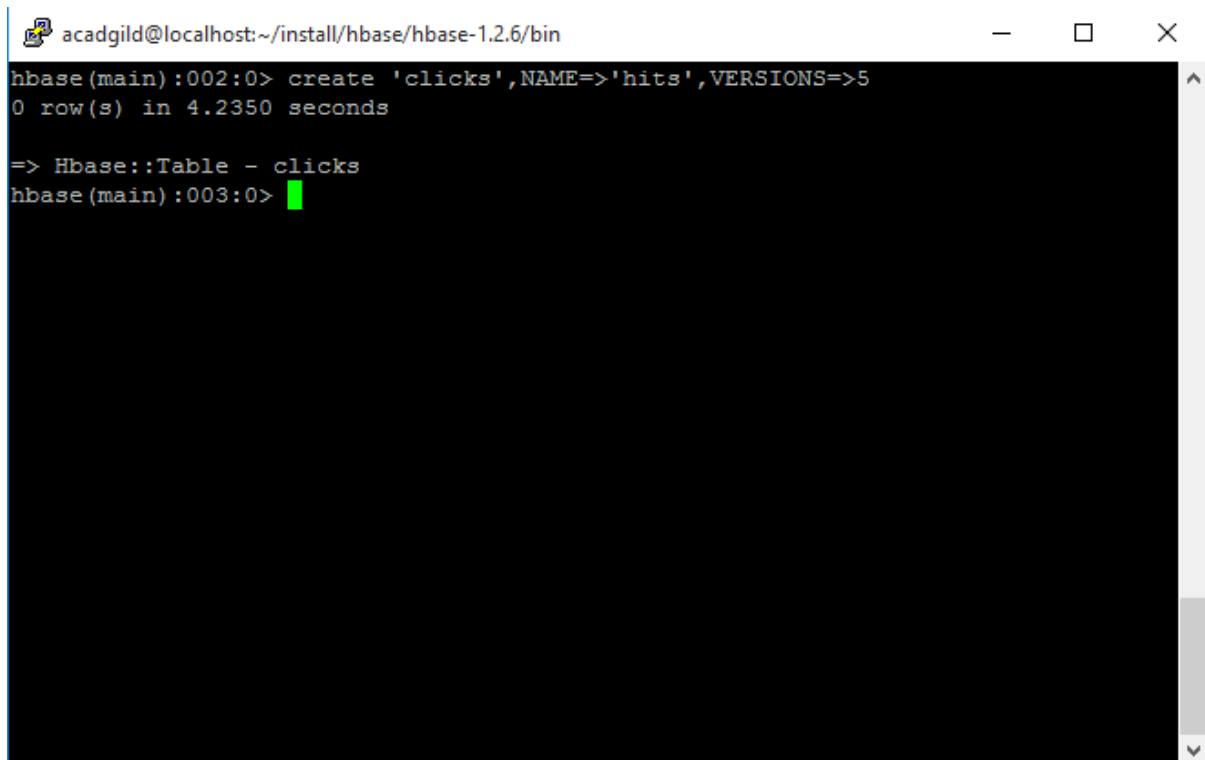
7.What happens internally when new data gets inserted into HBase table?

- When Put is used to store data, it uses the row, the column, and the timestamp.
- The timestamp is unique per version of the cell, and can be generated automatically or specified programmatically by your application, and must be a long integer.
- When a put/insert request is initiated, the value is first written into the WAL and then into the memstore.
- The values in the memstore is stored in the same sorted manner as in the HFile. Once the memstore is full, it is then flushed into a new HFile.
- HFile stores the data in sorted order i.e. the sequential rowkeys will be next to each other.
- HBase periodically performs compactions which will merge multiple HFiles and rewrite's them to a single HFile, this new HFile which is a result of compaction is also sorted.

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 'clicks',NAME=>'hits',VERSIONS=>5



```
acadgild@localhost:~/install/hbase/hbase-1.2.6/bin
hbase(main):002:0> create 'clicks',NAME=>'hits',VERSIONS=>5
0 row(s) in 4.2350 seconds

=> Hbase::Table - clicks
hbase(main):003:0>
```

```
acadmild@localhost:~/install/hbase/hbase-1.2.6/bin
hbase(main):002:0> create 'clicks',NAME=>'hits',VERSIONS=>5
0 row(s) in 4.2350 seconds

=> Hbase::Table - clicks
hbase(main):003:0> list
TABLE
acadmilddb.transactions_hbase
clicks
employee
htest
test
5 row(s) in 0.2540 seconds

=> ["acadmilddb.transactions_hbase", "clicks", "employee", "htest", "test"]
hbase(main):004:0>
```

2.Add few records in the table and update some of them. Use IP Address as row-key.

```
put 'clicks','27.59.2.31','hits:No_Of_Times','12'
put 'clicks','27.59.2.31','hits:No_Of_Times','32'
put 'clicks','27.59.2.31','hits:No_Of_Times','8'
put 'clicks','27.59.2.31','hits:No_Of_Times','44'
put 'clicks','27.59.2.31','hits:No_Of_Times','99'
```

```
acadgild@localhost:~/install/hbase/hbase-1.2.6/bin
hbase(main):004:0> put 'clicks','27.59.2.31','hits:No_Of_Times','12'
0 row(s) in 0.8130 seconds

hbase(main):005:0> put 'clicks','27.59.2.31','hits:No_Of_Times','32'
0 row(s) in 0.0440 seconds

hbase(main):006:0> put 'clicks','27.59.2.31','hits:No_Of_Times','8'
0 row(s) in 0.0330 seconds

hbase(main):007:0> put 'clicks','27.59.2.31','hits:No_Of_Times','44'
0 row(s) in 0.1590 seconds

hbase(main):008:0> put 'clicks','27.59.2.31','hits:No_Of_Times','99'
0 row(s) in 0.0100 seconds

hbase(main):009:0> █
```

Scan the table to view if all the previous versions are getting displayed.

scan 'clicks',{COLUMN=>'hits:No_Of_Times',VERSIONS=>5}

```
acadgild@localhost:~/install/hbase/hbase-1.2.6/bin
hbase(main):009:0> scan 'clicks',{COLUMN=>'hits:No_Of_Times',VERSIONS=>5}
ROW                                COLUMN+CELL
27.59.2.31                         column=hits:No_Of_Times, timestamp=1519540757533, value=99
27.59.2.31                         column=hits:No_Of_Times, timestamp=1519540719006, value=44
27.59.2.31                         column=hits:No_Of_Times, timestamp=1519540712635, value=8
27.59.2.31                         column=hits:No_Of_Times, timestamp=1519540704243, value=32
27.59.2.31                         column=hits:No_Of_Times, timestamp=1519540682956, value=12
1 row(s) in 0.1000 seconds

hbase(main):010:0> █
```