Name: Shreya Singh	Class/Roll No.: D16AD/55	Grade:

Title of Experiment: To study Hbase shell perform CRUD(create, read, update and delete) operations on table.

Objective of Experiment:

The objective of this experiment is to familiarize students or participants with HBase's shell commands and their practical application to perform CRUD (Create, Read, Update, Delete) operations on a table within an HBase database

Outcome of Experiment:

Thus, we got familiarized to various HBase commands and executed them in the HBase shell

Problem Statement:

In a world where big data is becoming increasingly important, it is essential to have a good grasp of NoSQL databases like HBase. The problem at hand is the need to perform CRUD operations on an HBase table using the HBase shell.

Description / Theory:

HBase can be defined as an open-source, distributed, and scalable NoSQL database system designed for storing and managing vast amounts of data across a distributed cluster of commodity hardware. It is characterized by its ability to handle both structured and semi-structured data, its high scalability, and its seamless integration with the Hadoop ecosystem for processing and analyzing large datasets. HBase is often used in applications that require fast and efficient data storage and retrieval, making it a valuable tool in the world of big data and real-time data processing.

Key Features & Characteristics:

Distributed and Scalable: HBase is designed to be distributed across clusters of commodity hardware, making it highly scalable. As data grows, you can easily add more nodes to your HBase cluster to accommodate increased storage and processing requirements.

Column-Family Data Model: HBase uses a column-family data model, similar to Google Bigtable. Data is organized into tables, which are divided into rows and column families. Each column family can have multiple columns, and you can add columns dynamically without affecting existing data.

Schema-less: HBase is schema-less in the sense that you don't need to define a rigid schema upfront. You can insert data with varying structures into the same table, which makes it suitable for handling semi-structured or unstructured data.

Strong Consistency: HBase provides strong consistency for read and writes operations within a single row. This means that when you read data from HBase, you are guaranteed to see the most recent write to that data.

Automatic Sharding: HBase automatically splits and distributes tables into regions, which are the units of distribution. As data grows, HBase splits regions and moves them to different servers to maintain load balancing.

Highly Available: HBase is designed to be highly available by replicating data across multiple nodes in the cluster. If one node fails, data can still be retrieved from the replicas.

Batch and Real-time Processing: HBase is often used in conjunction with Hadoop's MapReduce for batch processing and Apache Spark or Apache Storm for real-time processing, making it suitable for a wide range of big data applications.

HBase Shell: HBase provides a command-line interface called the HBase shell, which allows users to interact with HBase using simple commands. This is useful for administrative tasks and data manipulation.

Integration with Hadoop: HBase seamlessly integrates with the Hadoop ecosystem, particularly HDFS (Hadoop Distributed File System), which allows for the storage and processing of vast amounts of data.

Community and Ecosystem: HBase is an Apache Software Foundation project with an active community of developers and users. It has a growing ecosystem of tools and libraries for various use cases.

Program & Output:

Performing CRUD Operations in Hbase.

- Enter "hbase shell" in terminal

```
[cloudera@quickstart ~]$ hbase shell
2023-09-14 08:05:59,722 INFO  [main] Configuration.deprecation: hadoop.native.li
b is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.12.0, rUnknown, Thu Jun 29 04:42:07 PDT 2017

hbase(main):001:0>
```

- Create student table as shown below:

```
hbase(main):001:0> create 'student','s_name','rollno'
0 row(s) in 1.4340 seconds

=> Hbase::Table - student
hbase(main):002:0>
```

- To see an already existing table use the following command:

```
hbase(main):005:0* list
TABLE
student
1 row(s) in 0.0260 seconds
```

- Use the following command to see the details of student table:

```
hbase(main):009:0* describe 'student'
Table student is ENABLED
student
COLUMN FAMILIES DESCRIPTION
{NAME => 'rollno', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE
=> '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEE
P_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'tru
e'}
{NAME => 's_name', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE
=> '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEE
P_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'tru
e'}
2 row(s) in 0.1210 seconds
```

- Insert Data in table:

```
hbase(main):010:0> put 'student','1','s_name','Heramb'
0 row(s) in 0.0520 seconds
```

Note: Insert 5-10 rows using same command.

- Update roll numbers for the same

```
hbase(main):015:0> put 'student','5','rollno','5'
0 row(s) in 0.0210 seconds

hbase(main):016:0> put 'student','4','rollno','4'
0 row(s) in 0.0030 seconds

hbase(main):017:0> put 'student','3','rollno','3'
0 row(s) in 0.0040 seconds

hbase(main):024:0* scan 'student'
ROW                           COLUMN+CELL
 1                            column=rollno:, timestamp=1694704898353, value=1
 1                            column=s_name:, timestamp=1694704678247, value=Heramb
 2                            column=rollno:, timestamp=1694704892679, value=2
 2                            column=s_name:, timestamp=1694704799446, value=Ramakant
 3                            column=rollno:, timestamp=1694704887220, value=3
 3                            column=s_name:, timestamp=1694704809099, value=Pawar
 4                            column=rollno:, timestamp=1694704880448, value=4
 4                            column=s_name:, timestamp=1694704846369, value=Shaunak
 5                            column=rollno:, timestamp=1694704871612, value=5
 5                            column=s_name:, timestamp=1694704856665, value=Ajinkya
5 row(s) in 0.0440 seconds

hbase(main):027:0* get 'student','1'
COLUMN                        CELL
 rollno:                      timestamp=1694704898353, value=1
 s_name:                      timestamp=1694704678247, value=Heramb
2 row(s) in 0.0070 seconds
```

- Delete Record

```
hbase(main):028:0> delete 'student','1','s_name'
0 row(s) in 0.0190 seconds

hbase(main):029:0> scan 'student'
ROW                           COLUMN+CELL
 1                            column=rollno:, timestamp=1694704898353, value=1
 2                            column=rollno:, timestamp=1694704892679, value=2
 2                            column=s_name:, timestamp=1694704799446, value=Ramakant
 3                            column=rollno:, timestamp=1694704887220, value=3
 3                            column=s_name:, timestamp=1694704809099, value=Pawar
 4                            column=rollno:, timestamp=1694704880448, value=4
 4                            column=s_name:, timestamp=1694704846369, value=Shaunak
 5                            column=rollno:, timestamp=1694704871612, value=5
 5                            column=s_name:, timestamp=1694704856665, value=Ajinkya
5 row(s) in 0.0240 seconds

hbase(main):030:0> deleteall 'student','2'
0 row(s) in 0.0040 seconds

hbase(main):031:0> scan 'student'
ROW                           COLUMN+CELL
 1                            column=rollno:, timestamp=1694704898353, value=1
 3                            column=rollno:, timestamp=1694704887220, value=3
 3                            column=s_name:, timestamp=1694704809099, value=Pawar
 4                            column=rollno:, timestamp=1694704880448, value=4
 4                            column=s_name:, timestamp=1694704846369, value=Shaunak
 5                            column=rollno:, timestamp=1694704871612, value=5
 5                            column=s_name:, timestamp=1694704856665, value=Ajinkya
4 row(s) in 0.0200 seconds
```

General Commands:

```
hbase(main):035:0> whoami
cloudera (auth:SIMPLE)
    groups: cloudera, default
```

```
hbase(main):037:0> version
1.2.0-cdh5.12.0, rUnknown, Thu Jun 29 04:42:07 PDT 2017
```

```
hbase(main):036:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 3.0000 average load
```

Data Definition Language:

```
hbase(main):038:0> disable 'student'
0 row(s) in 2.2840 seconds
```

```
hbase(main):039:0> is_enabled 'student'
false
0 row(s) in 0.0220 seconds

hbase(main):040:0> enable 'student'
0 row(s) in 1.2890 seconds

hbase(main):041:0> is_disabled 'student'
false
```

```
ıbase(main):044:0> disable 'student'
) row(s) in 2.2500 seconds

ıbase(main):045:0> drop 'student'
) row(s) in 1.2740 seconds

hbase(main):047:0> exists 'student'
Table student does not exist
0 row(s) in 0.0130 seconds
```

Data Manipulation Language:

```
hbase(main):032:0> count 'student'
4 row(s) in 0.0430 seconds
```

Results and Discussions:

In the practical session, we conducted a hands-on exploration of HBase using its shell commands to perform various CRUD (Create, Read, Update, Delete) operations. We also examined general commands and data definition and manipulation commands. Here's a summary of what we did and our observations:

1. **Starting HBase and Accessing the HBase Shell:**
   We initiated the HBase shell to interact with HBase and perform various operations.

2. **CRUD Operations:**

   Create: We used the create command to create an HBase table. This demonstrated how easy it is to set up a table in HBase.

   List: The list command allowed us to view the existing tables in the HBase cluster. This is useful for verifying table creation and management.

   Describe: With the describe command, we obtained detailed information about a specific table, including its schema, column families, and configuration settings.

   Put: The put command was used to insert data into the HBase table. We specified the row key, column family, column qualifier, and cell value to store the data.

   Scan: We employed the scan command to retrieve data from the table. This command is valuable for fetching multiple rows or specific subsets of data.

   Delete: The delete command allowed us to remove specific cells or rows from the table, providing data deletion capabilities.

   Deleteall: We used the deleteall command to delete all versions of a specified cell within a row.

3. **General Commands**:

   whoami: The whoami command helped identify the current HBase user. This is essential for security and auditing purposes.

   version: The version command provided information about the HBase version being used, useful for troubleshooting and compatibility checks.

   status: By running the status command, we obtained an overview of the HBase cluster's status, including the number of servers and regions.

4. **Data Definition Language (DDL) Commands:**

   disable: The disable command allows us to disable a table temporarily. Disabled tables cannot be accessed until they are re-enabled.

   is_enabled: We used the is_enabled command to check whether a table is currently enabled or disabled.

   enable: The enable command re-enables a previously disabled table, making it accessible again.

   is_disabled: The is_disabled command verifies whether a table is currently disabled or not.

   drop: We employed the drop command to permanently delete a table and all its associated data. This is an irreversible operation.

   exists: The exists command checks if a table with a specified name exists in the HBase cluster.

5. **Data Manipulation:**

   count: The count command helped us determine the number of rows in a table, providing a quick way to gauge the size of a table.

In conclusion, this practical session allowed us to gain hands-on experience with HBase's shell commands, covering various aspects of database management, data manipulation, and cluster monitoring. These commands are fundamental to effectively interact with HBase, perform data operations, and maintain the health of an HBase cluster. Understanding these commands is essential for anyone working with HBase in real-world applications, especially in the context of big data and distributed systems.