Name : Manav Pahilwani	Class/Roll No. : D16AD/ 37	Grade :

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

Theory:

HBase Shell

HBase contains a shell using which you can communicate with HBase. HBase uses the Hadoop File System to store its data. It will have a master server and region servers. The data storage will be in the form of regions (tables). These regions will be split up and stored in region servers.

The master server manages these region servers and all these tasks take place on HDFS. Given below are some of the commands supported by HBase Shell.

General Commands

- status - Provides the status of HBase, for example, the number of servers.
- version - Provides the version of HBase being used.
- table_help - Provides help for table-reference commands.
- whoami - Provides information about the user.

Data Definition Language

These are the commands that operate on the tables in HBase.

- create - Creates a table.
- list - Lists all the tables in HBase.
- disable - Disables a table.

- is_disabled - Verifies whether a table is disabled.
- enable - Enables a table.
- is_enabled - Verifies whether a table is enabled.
- describe - Provides the description of a table.
- alter - Alters a table.
- exists - Verifies whether a table exists.
- drop - Drops a table from HBase.
- drop_all - Drops the tables matching the 'regex' given in the command.
- Java Admin API - Prior to all the above commands, Java provides an Admin API to achieve DDL functionalities through programming. Under org.apache.hadoop.hbase.client package, HBaseAdmin and HTableDescriptor are the two important classes in this package that provide DDL functionalities.

Data Manipulation Language

- put - Puts a cell value at a specified column in a specified row in a particular table.
- get - Fetches the contents of row or a cell.
- delete - Deletes a cell value in a table.
- deleteall - Deletes all the cells in a given row.
- scan - Scans and returns the table data.
- count - Counts and returns the number of rows in a table.
- truncate - Disables, drops, and recreates a specified table.
- Java client API - Prior to all the above commands, Java provides a client API to achieve DML functionalities, CRUD (Create Retrieve Update Delete) operations and more through programming, under org.apache.hadoop.hbase.client package. HTable Put and Get are the important classes in this package.

Output Screenshots:

```
[cloudera@quickstart ~]$ hbase shell
2023-10-15 06:57:47,676 INFO  [main] Configuration.deprecation: hadoop.native.lib is deprecated.
e
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct  4 11:16:18 PDT 2017

hbase(main):001:0> version
1.2.0-cdh5.13.0, rUnknown, Wed Oct  4 11:16:18 PDT 2017

hbase(main):002:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 2.0000 average load

hbase(main):003:0> whoami
cloudera (auth:SIMPLE)
    groups: cloudera, default
```

Create Table:

```
hbase(main):004:0> create 'customer', 'address','order'
0 row(s) in 2.5970 seconds

=> Hbase::Table - customer
hbase(main):005:0> lis
NameError: undefined local variable or method `lis' for #<Object:0x73fa1cea>

hbase(main):006:0> list
TABLE
customer
1 row(s) in 0.0610 seconds
```

Describe Table:

```
hbase(main):008:0> describe 'customer'
Table customer is ENABLED
customer
COLUMN FAMILIES DESCRIPTION
{NAME => 'address', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => '
NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE
=> 'true'}
{NAME => 'order', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NO
NE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE =>
 'true'}
2 row(s) in 0.3000 seconds
```

Insert values in Table:

```
hbase(main):011:0> delete 'customer', '101', 'address:city'
0 row(s) in 0.1510 seconds

hbase(main):012:0> put 'customer','ABC', 'address:city','Mumbai'
0 row(s) in 0.0580 seconds

hbase(main):013:0> put 'customer','PQR','address:city','Pune'
0 row(s) in 0.0260 seconds

hbase(main):014:0> put 'customer', 'ABC', 'order:no', '101'
0 row(s) in 0.0270 seconds

hbase(main):015:0> put 'customer', 'PQR', 'order:no','102'
0 row(s) in 0.0250 seconds
```

Retrieve Table Data:

```
hbase(main):016:0> get 'customer', 'ABC'
COLUMN                          CELL
 address:city                   timestamp=1697379685093, value=Mumbai
 order:no                       timestamp=1697379884980, value=101
2 row(s) in 0.0810 seconds
```

Scan Table (List all records):

```
hbase(main):017:0> scan 'customer'
ROW                             COLUMN+CELL
 ABC                            column=address:city, timestamp=1697379685093, value=Mumbai
 ABC                            column=order:no, timestamp=1697379884980, value=101
 PQR                            column=address:city, timestamp=1697379703231, value=Pune
 PQR                            column=order:no, timestamp=1697379898507, value=102
2 row(s) in 0.0930 seconds
```

Aggregate Functions (Count Records):

```
hbase(main):018:0> count 'customer'
2 row(s) in 0.0740 seconds

=> 2
```

Delete Cell:

```
hbase(main):020:0> delete 'customer', 'PQR', 'address:city'
0 row(s) in 0.0310 seconds
```

Truncate Table:

```
hbase(main):021:0> truncate 'customer'
Truncating 'customer' table (it may take a while):
 - Disabling table...
 - Truncating table...
0 row(s) in 3.9630 seconds
```

Drop Table:

```
hbase(main):023:0> disable 'customer'
0 row(s) in 2.3600 seconds

hbase(main):024:0> drop 'customer'
0 row(s) in 1.4930 seconds
```

Conclusion:
Successfully, created table in HBase shell, and executed commands on it.