Exploring HBase 2

Objective: Use the HBase Shell to populate and explore the `hbase:meta` table.

HBase tables: `[some name]` (use your own table name)

In this exercise you will use the HBase Shell to explore the `hbase:meta` table.

Hbase is not a relational database. The operations available for data stored in hbase are the following:

- Get: retrieves a row or a subset of a row
- Put: Add or update a row or a subset of a row
- Scan: retrieve a range of sequential rows
- Delete: remove a row or a subset of a row

Data Definition Language

`ddl` stands for Data Definition Language. So help on creating tables defining column family attributes will be displayed by typing:

```
hbase> help 'ddl'
hbase> help 'get'
hbase> help 'put'
hbase> help 'scan'
hbase> help 'delete'
```

So let’s do a couple of these.

<img src="https://user-images.githubusercontent.com/558905/40613898-7a6c70d6-624e-11e8-9178-7bde851ac7bd.png" align=“left” width=“50” height=“50” title=“ToDo Logo”>
<h4>1. Create the Table</h4>

Enter HBase Shell:

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

Create a table in HBase (again, name it your choice) and specify a column name:

```
hbase(main):007:0> create '[some name]','address'
```

Now check the tables:

```
hbase(main):008:0* list
```

Verify the new table:

```
hbase(main):012:0> describe '[some name]'
DESCRIPTION          ENABLED
'[some name]', {NAME => 'address', DATA_BLOCK_ENCODING =>
'NONE', BLOOMFILTER => 'ROW', REPLICATI true ON_SCOPE => '
0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS =
> '0', TTL => '2147483647', KEEP_DELETED_CELLS => 'false',
 BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE =>
 'true'}
1 row(s) in 0.0750 seconds
```

<img src="https://user-images.githubusercontent.com/558905/40613898-7a6c70d6-624e-11e8-9178-7bde851ac7bd.png" align=“left” width=“50” height=“50” title=“ToDo Logo”>
<h4>2. PUT Something in the Table</h4>

Review the output and notice the column (‘address’), parameter properties such as COMPRESSION, DATA_BLOCK_ENCODING, whether is kept in memory…

Insert data using PUT command :

```
hbase(main):004:0> put '[some name]','row1','address','val
ue1'
0 row(s) in 0.4250 seconds
hbase(main):005:0> put '[some name]','row2','address','val
ue1'
```

```
0 row(s) in 0.0120 seconds
hbase(main):006:0> put '[some name]','row1','address','value2'
0 row(s) in 0.0110 seconds
hbase(main):007:0> put '[some name]','row1','address','value3'
0 row(s) in 0.0070 seconds
```

View the table data using SCAN:

```
hbase(main):008:0> scan '[some name]'
ROW        COLUMN+CELL
row1       column=address:, timestamp=1408499993409, value=value3
row2       column=address:, timestamp=1408499977588, value=value1
2 row(s) in 0.1020 seconds
```

This time view the data using SCAN with versions:

```
hbase(main):011:0> scan '[some name]', { VERSIONS => 3}
ROW        COLUMN+CELL
row1       column=address:, timestamp=1408499993409, value=value3
row2       column=address:, timestamp=1408499977588, value=value1
2 row(s) in 0.0200 seconds
```

Use the `count` command to determine the number of rows in a table:

```
hbase(main):002:0> count '[some name]'
2 row(s) in 0.4020 seconds
=> 2
```

<img src="https://user-images.githubusercontent.com/558905/40613898-7a6c70d6-624e-11e8-9178-7bde851ac7bd.png" align=“left” width=“50” height=“50” title=“ToDo Logo”>
<h4>3. Retrieve data using GET</h4>

```
hbase(main):012:0> get '[some name]','row1'
COLUMN        CELL
address:      timestamp=1408499993409, value=value3
1 row(s) in 0.0530 seconds
```

Use GET command to check on row2:

```
hbase(main):003:0> get '[some name]','row2'
COLUMN        CELL
address:      timestamp=1408499977588, value=value1
1 row(s) in 0.0110 seconds
```

<img src="https://user-

images.githubusercontent.com/558905/40613898-7a6c70d6-624e-11e8-9178-7bde851ac7bd.png" align=“left” width=“50” height=“50” title=“ToDo Logo”>

<h4>4. Now use PUT command on row2</h4>

```
hbase(main):004:0> put '[some name]','row2','address','val
ue2'
0 row(s) in 0.3340 seconds
```

Retrieve the value:

```
hbase(main):005:0> get '[some name]','row2'
```

Test the row counts:

```
hbase(main):006:0> count '[some name]'
2 row(s) in 0.2290 seconds
=> 2
```

<img src="https://user-images.githubusercontent.com/558905/40613898-7a6c70d6-624e-11e8-9178-7bde851ac7bd.png" align=“left” width=“50” height=“50” title=“ToDo Logo”>

<h4>5. Run the delete command</h4>

```
hbase(main):020:0> delete '[some name]','row1','address'
```

```
0 row(s) in 0.1990 seconds
```

We can verify the data using SCAN command:

```
hbase(main):021:0> scan '[some name]'
ROW             COLUMN+CELL
row2            column=address:, timestamp=1409577011732, val
ue=value2
1 row(s) in 0.0750 seconds
```

<img src="https://user-images.githubusercontent.com/558905/40613898-7a6c70d6-624e-11e8-9178-7bde851ac7bd.png" align=“left” width=“50” height=“50” title=“ToDo Logo”>
<h4>6. Disable</h4>

Before we disable the table, let’s run a count on the table:

```
hbase(main):022:0> count '[some name]'
1 row(s) in 0.0320 seconds
=> 1
```

Execute the disable now:

```
hbase(main):007:0> disable '[some name]'
0 row(s) in 1.7060 seconds
```

Do the count command again:

```
hbase(main):008:0> count '[some name]'
```

Now enable the table again:

```
hbase(main):025:0> enable '[some name]'
0 row(s) in 0.8730 seconds
```

Use count to check the table again:

```
hbase(main):026:0> count '[some name]'
```

Check the table again:

```
hbase(main):010:0> list
TABLE
...
[some name]
...
```

<img src="https://user-images.githubusercontent.com/558905/40613898-7a6c70d6-624e-11e8-9178-7bde851ac7bd.png" align=“left” width=“50” height=“50” title=“ToDo Logo”>

<h4>7. Drop</h4>

Now use the DROP command to get rid of the table:

```
hbase(main):011:0> drop '[some name]'
0 row(s) in 0.8290 seconds
```

To verify use the list command again:

```
hbase(main):012:0> list
```

Results

You’re getting the hang of it! Good!

<button type=“button”><a href="https://virtuant.github.io/hadoop-overview-spark-hwx/">Go Back</a></button>

<br>

<br>