HBase and Pig Interaction

Objective: Get acquainted with integrating HBase and Pig.

Exercise directory: `~/data/movie`

For this lab, we will be working with some movie data. If you need it the data can be gotten from here. In the `ml-1m.zip` file, you should see three files inside: `movies.dat, ratings.dat, users.dat`.

Let's put the data into HDFS quickly with some data prep.

```
hdfs dfs -mkdir -p /moviedata/ratings
hdfs dfs -mkdir -p /moviedata/movies
hdfs dfs -mkdir -p /moviedata/users
sed 's/::/#/g' /dataset/ml-1m/movies.dat > movies.t
sed 's/::/#/g' /dataset/ml-1m/ratings.dat > ratings.t
sed 's/::/#/g' /dataset/ml-1m/users.dat > users.t
hdfs dfs -put movies.t /moviedata/movies
hdfs dfs -put ratings.t /moviedata/ratings
hdfs dfs -put users.t /moviedata/users
```

Awesome! Let's put aside this data for now, and just get some practice interacting with HBase. We will come back to this data that we just put into HDFS shortly.

Let’s create a table now.

```
hbase(main):001:0> create 'tabletest', 'columnfamily1'
0 row(s) in 1.5990 seconds
=> Hbase::Table - tabletest
```

In the last command, we specify the create command with tabletest as the name of our table with columnfamily1 as a column family name — the command is create `<<table name>>`, `<<column family>>`.

Use the list command to print out some information about your table:

```
hbase(main):002:0> list 'tabletest'
TABLE


tabletest


1 row(s) in 0.0270 seconds

=> ["tabletest"]
```

Naturally we should put some data into this table to get a real feel for it. Let’s just throw some test data into it.

```
hbase(main):004:0> put 'tabletest', 'firstrow', 'columnfam
ily1','testValue1'
0 row(s) in 0.0130 seconds
hbase(main):005:0> put 'tabletest', 'secondrow', 'columnfa
mily1','testValue2'
0 row(s) in 0.0150 seconds
hbase(main):006:0> put 'tabletest', 'thirdrow', 'columnfam
ily1','testValue3'
0 row(s) in 0.0060 seconds
```

This format looks a little different:

- The first part put `tabletest` is just telling HBase which table we want to put the data into.
- `firstrow` denotes where in the table we want to put the data. Remember random read and writes, so we have to specify exactly where to store the data.
- `columnfamily1` refers to which column family we are going to be putting data into
- Finally we put the value of `testvalue`.

It's important to note here that HBase doesn't support multiple columns in a single statement. So if you had a table with a column family with two columns under it, the put statement to add to both columns would look like the following:

```
PUT 'tableName', 'columnFamily:column1', 'data'
```

```
PUT 'tableName', 'columnFamily:column2', 'data'
```

In other words, the : is used to specify the column names.

Let’s query this data and see if it what we expected.

```
hbase(main):007:0> scan 'tabletest'

ROW             COLUMN+CELL

firstrow        column=columnfamily1:, timestamp=1518293802
613, value=testValue1
secondrow       column=columnfamily1:, timestamp=1518293817
887, value=testValue2
thirdrow        column=columnfamily1:, timestamp=1518293828
203, value=testValue3
3 row(s) in 0.0610 seconds
```

Perfect. Notice that when we do a scan it brings up all values in the table. It shows us the ROW and COLUMN and CELL. Notice we just put one value into the column family but there are now two. The timestamp just magically appeared. That is because every row has a timestamp.

Let’s try and get just one row of the table.

```
get 'tabletest', 'thirdrow'
```

```
COLUMN                CELL

columnfamily1:        timestamp=1518293828203, value=testVa
lue3
1 row(s) in 0.0260 seconds
```

This shows just the column and the cell.

A cell is the value that is at the intersection of a row and a column. To find a cell you need to know the column, row, and version (timestamp) to get the correct value. Pretty cool.

It shows us the column family and the cell with the value and the timestamp. The timestamp should match above because that timestamp is created when data is put into the row so it shouldn’t change unless something happens to that row.

Something a little different with HBase is the disable and enable commands. If you want to update the settings of a table or drop a table, you need to disable the table. Then when you are ready to use it again, enable the table. Let’s try it.

First let’s do a describe on the table.

```
describe 'tabletest'

Table tabletest is ENABLED
```

```
tabletest

COLUMN FAMILIES DESCRIPTION
{NAME => 'columnfamily1', DATA_BLOCK_ENCODING => 'NONE', B
LOOMFILTER =>'ROW', REPLICATION_SCOPE => '0', VERSIONS =>
'1'
, COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FORE
VER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536',
 IN_
MEMORY => 'false', BLOCKCACHE => 'true'}


1 row(s) in 0.0490 seconds
```

Notice the table is ENABLED.

```
disable 'tabletest'
describe 'tabletest'
```

Table tabletest is DISABLED

```
tabletest



COLUMN FAMILIES
{NAME => 'columnfamily1', DATA_BLOCK_ENCODING => 'NONE', B
```

```
LOOMFILTER =>'ROW', REPLICATION_SCOPE => '0', VERSIONS =>
'1'
, COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FORE
VER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536',
 IN_
MEMORY => 'false', BLOCKCACHE => 'true'}
1 row(s) in 0.0580 seconds
```

It’s now disabled. Let’s try and put some data into it and see what happens.

```
put 'tabletest', 'fourthrow', 'columnfamily1', 'testValue4
'


ERROR: Failed 1 action: NotServingRegionException: 1 time,
```

Got an error. Let’s enable that table and try and put data into it.

```
hbase(main):013:0> enable 'tabletest'
0 row(s) in 1.2850 seconds

hbase(main):014:0> put 'tabletest', 'fourthrow', 'columnfa
mily1','testValue4'
0 row(s) in 0.0160 seconds

hbase(main):015:0> get 'tabletest', 'fourthrow'
```

```
COLUMN                CELL

columnfamily1:     timestamp=1518295072544, value=testValu
e4
1 role(s) in 0.0240 seconds
```

It worked. Awesome job!! Let’s try and drop the table.

```
drop 'tabletest'

ERROR: Table tabletest is enabled. Disable it first.
```

The table is enabled so it won’t let us drop it. Perfect. Go ahead and drop that table on your own. Once you get done with that, go ahead and type quit and press enter and it’ll get you back to the main command line.

Loading Data From HDFS into HBase using Pig

Let’s get back to that data that we put into HDFS and let’s put it into HBase using Pig. First thing that we need to do is go back into the HBase shell and make three tables.

```
create 'users', 'userdata'

0 row(s) in 1.4860 seconds
=> Hbase::Table - users

```

```
create 'ratings', 'ratingsdata'

0 row(s) in 1.5700 seconds
=> Hbase::Table - ratings

create 'movies', 'moviedata'

0 row(s) in 1.5380 seconds
=> Hbase::Table - movies
```

Great! Let's create a script to load the data using pig. Copy the following code into a file called `loadHbase.pig`.

```
movies = LOAD '/moviedata/movies/movies.t' USING PigStorag
e(',') AS (movieid:int, title:chararray, genres:chararray)
;
STORE movies INTO 'hbase://movies' USING org.apache.pig.ba
ckend.hadoop.hbase.HBaseStorage('moviedata:title moviedata
:genres');
ratings = LOAD '/moviedata/ratings/ratings.t' USING PigSto
rage(',') AS (userid:int, movieid:int, rating:int, tstamp:
chararray);
STORE ratings INTO 'hbase://ratings' USING org.apache.pig.
backend.hadoop.hbase.HBaseStorage('ratingsdata:movieid rat
ingsdata:rating ratingsdata:tstamp');
users = LOAD '/moviedata/users/users.t' USING PigStorage('
,') AS (userid:int, gender:chararray, age:int, occupation:
```

```
int, zipcode:chararray);
STORE users INTO 'hbase://users' USING org.apache.pig.back
end.hadoop.hbase.HBaseStorage('userdata:gender userdata:ag
e userdata:occupation userdata:zipcode');
```

Awesome. The data should now be loaded into HBase. Let’s jump into the HBase shell and do some querying of the data.

```
hbase(main):002:0> scan 'movie'

 988       column=moviedata:title, timestamp=1518381804300
, value=Grace of My Heart (1996)

 989       column=moviedata:genres, timestamp=151838180430
0, value=Drama

 989       column=moviedata:title, timestamp=1518381804300
, value=Schlafes Bruder (Brother of Sleep) (1995)

 99        column=moviedata:genres, timestamp=151838180368
5, value=Documentary

 99        column=moviedata:title, timestamp=1518381803685
```

, value=Heidi Fleiss: Hollywood Madam (1995)

 990   column=moviedata:genres, timestamp=151838180430
0, value=Action|Adventure|Thriller

 990   column=moviedata:title, timestamp=1518381804300
, value=Maximum Risk (1996)

 991   column=moviedata:genres, timestamp=151838180430
0, value=Drama|War

 991   column=moviedata:title, timestamp=1518381804300
, value=Michael Collins (1996)

 992   column=moviedata:genres, timestamp=151838180430
1, value= The (1996)

 992   column=moviedata:title, timestamp=1518381804301
, value=Rich Man's Wife

 993   column=moviedata:genres, timestamp=151838180430

```
1, value=Drama

 993       colun=moviedata:title, timestamp=1518381804301,
 value=Infinity (1996)

 994       column=moviedata:genres, timestamp=151838180430
1, value=Drama

 994       column=moviedata:title, timestamp=1518381804301
, value=Big Night (1996)

 996       column=moviedata:genres, timestamp=151838180430
1, value=Action|Drama|Western

 996       column=moviedata:title, timestamp=1518381804301
, value=Last Man Standing (1996)

 997       column=moviedata:genres, timestamp=151838180430
1, value=Drama|Thriller

 997       column=moviedata:title, timestamp=1518381804301
```

```
, value=Caught (1996)


 998       column=moviedata:genres, timestamp=151838180430
2, value=Action|Crime


 998       column=moviedata:title, timestamp=1518381804302
, value=Set It Off (1996)


 999       column=moviedata:genres, timestamp=151838180430
2, value=Crime

 999       column=moviedata:title, timestamp=1518381804302
, value=2 Days in the Valley (1996)


3883 row(s) in 14.4740 seconds
```

There are a lot of rows there and scan will show you them all. I just put the last couple. Great job! You've successfully loaded some data into HBase using Pig.

Let's do a quick query to see how the data is setup in HBase.

```
get 'movies', 77
```

Are

```
COLUMN                CELL

moviedata:genres      timestamp=1518381803660, value=Docu
mentary

moviedata:title       timestamp=1518381803660, value=Nico
 Icon (1995)


2 row(s) in 0.1390 seconds
```

Results

Great - now you understand about enabling and disabling tables.