Workshop 6

Apache HBase

General Instructions:

The purpose of this workshop is to get you started with Hadoop HBase. Here you will learn how to use Apache HBase to insert, read and update records.

Online resources:

https://hbase.apache.org/

Apache HBase

Prerequisites

Before starting using HBase you need to ensure that all daemons (processes) are running.

- Open a new Terminal and enter this command:

```
$ sudo service --status-all | grep -i 'HBase'
```

This command should display something similar to this screenshot

```
HBase master daemon is running                          [  OK  ]
hbase-regionserver is running
HBase rest daemon is running                            [  OK  ]
HBase Solr Indexer is running                           [  OK  ]
HBase thrift daemon is running                          [  OK  ]
```

- If it is not the case please start all HBase daemons using these commands:

```
$  sudo service hbase-master start
$  sudo service hbase-regionserver start
$  sudo service hbase-solr-indexer start
$  sudo service hbase-thrift start
```

- Check again that all daemons are running

Exercise 1: HBase hands-on

In this exercise you will create a new table, enter and manipulate some data using HBase command line shell.

1. Open a new Terminal window
2. Start a new HBase session:

```
$  hbase shell
```

3. Create a new table 'students' with one Column Family 'data'

```
$ create 'students', 'data'
```

4. Print tables list to check

```
$ list
```

5. Add a few row to the the 'students' table

```
$ put 'students', 'row1', 'data:name', 'michel'
$ put 'students', 'row2', 'data:age', '45'
$ put 'students', 'row3', 'data:city', 'paris'
```

6. Show all records in the 'students' table

```
$ scan 'students'
```

7. How many rows in the 'students' table

```
$ count 'students'
```

8. Add a new column 'age' to 'michel'

```
$
```

9. Change 'age' in 'row2' to '56'

```
$
```

10. List all rows for the 'age' column

```
$
```

11. Modify 'students' table to enable data versioning on 'data' column family and set it to 5

```
$
```

12. Modify 'students' table to enable data versioning on 'city' column and set it to 3

```
$
```

13. Change the value of 'city' in 'row3' to 'London'

```
$
```

14. Print all rows in 'students' table with all data versions of 'city'

```
$
```

15. Without adding a new data version to 'city', replace 'London' in 'row3' / 'city' by 'Roma'

```
$
```

16. Add a new Column Family to 'students' with versioning enabled and set it to 3

```
$
```

17. Print 'students' metadata and schema

```
$
```

Apache Hadoop Tools and Apache HBase Interoperability

Exercise 2: Populate HBase table using Pig

The goal of this exercise is to familiarize you with data integration techniques between Pig and HBase. To be able to write data into HBase using Pig you need to use the *HBaseStorage()* class provided by Pig.

Online resource:

https://pig.apache.org/docs/r0.15.0/api/org/apache/pig/backend/hadoop/hbase/HBaseStorage.html

1. From HBase, create a new table 'stations' with one Column Family 'stations_infos'

2. Open a new Pig session and load the **Stations_2018.csv** file into a Pig relation

   ```
   grunt>
   ```

3. Remove the CSV file header

   ```
   grunt>
   ```

4. Write a STORE statement that uses **HBaseStorage()** to write data to HBase

   ```
   grunt>
   ```

5. From HBase, print rows count for table 'stations'

   $

6. Print the row for the stations with code = 6100

   $

Exercise 3: Hive Integration

Reasons to use Hive on HBase is that a lot of data sitting in HBase due to its usage in a real-time environment, but never used for analysis as there are less connectivity tools to HBase directly.

In this exercise you will create a new Hive table which use storage handler mechanism to create HBase tables from Hive. The ***HBaseStorageHandler()*** class allows Hive DDL (*Data Definition Language*) for managing table definitions in both Hive metastore and HBase's catalog simultaneously and consistently.

Online resource:

https://cwiki.apache.org/confluence/display/Hive/HBaseIntegration

1. From Hive, create a new managed table '**stations_hbase**' and define:
   a. schema to read **Stations_2018.csv** file
   b. using HBase **Serde** **HBaseStorageHandler()** to store the data into HBase
   c. HBase **table name**: **stations_hive**, **Column Family**: '**data**'

$

2. Write a HiveQL query to populate '**stations_hbase**' with rows from '**stations**' (table created in Hive workshop)

   $

3. From Hive print the rows count from '**stations_hbase**' table

   $

4. Write a query to retrieve the row from station's code = **6100**

   $

5. From HBase print the rows count from '**stations_hive**' table

   $

6. Retrieve the row for the station's code = **6100**

   $