Práctica Hive

Instalación

1. Verifique que Java esté instalado.

```
    $ java -version
    java version "1.8.0_151"
    Java(TM) SE Runtime Environment (build 1.8.0_151-b12)
    Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, m
ixed mode)
```

2. Reinstale Hadoop borrando la carpeta original y volviendo a descomprimir el instalador.

```
    $ cd /home/mlds/Descargas
    $ rm -rf hadoop-3.1.2
    $ tar -xzvf hadoop-3.1.2.tar.gz
```

3. Descargue Hive desde link
4. Descomprima Hive y cambie al directorio creado:

```
    $ tar -xzvf apache-hive-2.3.5-bin.tar.gz
    $ cd apache-hive-2.3.5-bin
```

5. Desde la misma consola, diríjase a su home(`/home/mlds`), ejecute `nano .bashrc` y añada las siguientes líneas:

```
    export HIVE_HOME=/home/mlds/Descargas/apache-hive-2.3.
5-bin
    export PATH=$PATH:$HIVE_HOME/bin
```

6. Guarde el archivo y ejecute el siguiente comando para que los cambios surtan efecto:

```
    $ source ~/.bashrc
```

7. Verifique que Hadoop esté funcionando correctamente:

```
    $ echo $HADOOP_HOME
    /home/mlds/Descargas/hadoop-3.1.2
    $ hadoop version
    Hadoop 3.1.2
    Source code repository
    ...
```

8. Vamos a crear un archivo de configuración llamado `hive-site.xml`:

```
    $ cp /home/mlds/Descargas/apache-hive-2.3.5-bin/conf/h
ive-default.xml.template /home/mlds/Descargas/apache-hive-
2.3.5-bin/conf/hive-site.xml
```

9. Cree los directorios HDFS `~/tmp` y `~/warehouse` con el siguiente comando:

```
$ hadoop fs -mkdir ~/tmp
$ hadoop fs -mkdir ~/warehouse
```

10. Cambie los permisos de los directorios creados:

```
$ hadoop fs -chmod 733 ~/tmp
$ hadoop fs -chmod 755 ~/warehouse
```

11. Modifique el archivo `hive-site.xml` que se creó en el **paso 8**. Ubique las siguientes 5 propiedades en el XML y modifíquelas:

```
<name>hive.exec.scratchdir</name>
<value>/home/mlds/tmp/hive-${user.name}</value>
...
<name>hive.exec.local.scratchdir</name>
<value>/home/mlds/tmp/${user.name}</value>
...
<name>hive.downloaded.resources.dir</name>
<value>/home/mlds/tmp/${user.name}_resources</value>
...
<name>hive.scratch.dir.permission</name>
<value>733</value>
...
<name>hive.metastore.schema.verification</name>
<value>false</value>
...
<name>hive.metastore.warehouse.dir</name>
```

```
    <value>/home/mlds/warehouse</value>
```

12. Luego, en una consola sitúese en la carpeta de instalación de Hive y ejecute el siguiente comando

```
    $ cd /home/mlds/Descargas/apache-hive-2.3.5-bin
    $ schematool -dbType derby -initSchema
```

13. Ejecute Hive desde la misma consola.

```
    $ hive
    SLF4J: Class path contains multiple SLF4J bindings.
    SLF4J: Found binding in ...
```

14. Pruebe Hive. Liste las bases de datos por defecto.

```
    $ hive> show databases;
    OK
    default
    Time taken: 0.007 seconds, Fetched: 1 row(s)
```

Hive Data Definition Language

Cuando ejecute el comando `hive` es importante que esté situado en la carpeta donde quedó instalado Hive, debido a que el tratará de buscar la carpeta `metastore_db` que se creó en el paso 12 de la sección anterior)

1. Creación de tablas:

```
    hive> CREATE TABLE ratings (rate INT, film STRING);
    hive> CREATE TABLE invites (foo INT, bar STRING) PARTI
TIONED BY (ds STRING);
    OK
    Time taken: 0.263 seconds
```

2. Liste todas las tablas:

```
    hive> SHOW TABLES;
    OK
    invites
    ratings
    Time taken: 0.017 seconds, Fetched: 2 row(s)
    hive> SHOW TABLES 'ra*.';
    OK
    ratings
    Time taken: 0.029 seconds, Fetched: 1 row(s)
```

3. Obtenga la descripción de una tabla:

```
    hive> DESCRIBE ratings;
    OK
    rate                    int

    film                    string

```

```
    Time taken: 0.037 seconds, Fetched: 2 row(s)
```

4. Añada o elimine columnas de una tabla:

```
    hive> ALTER TABLE ratings RENAME TO calificaciones;
    hive> ALTER TABLE calificaciones ADD COLUMNS (new_col
INT);
    hive> ALTER TABLE invites ADD COLUMNS (new_col2 INT CO
MMENT 'dejamos un comentario');
    hive> ALTER TABLE invites REPLACE COLUMNS (foo INT, ba
r STRING, baz INT COMMENT 'baz reemplaza new_col2');
```

5. Verifique la estructura de la tabla:

```
    hive> DESCRIBE invites;
```

Documentación completa de Hive DDL

Hive Data Manipulation Language

1. Puede cargar archivos al warehouse de Hive usando:

```
    hive> CREATE TABLE pokes (foo INT, bar STRING);
    hive> LOAD DATA LOCAL INPATH './examples/files/kv1.txt
' OVERWRITE INTO TABLE pokes;
```

La palabra clave **LOCAL** determinará si busca en el sistema de archivos o si en el sistema de archivos de Hadoop.

2. Recuerde que si se define una partición en una tabla, se debe especificar a qué partición pertenece una serie de datos que se vaya a ingresar a la tabla:

```
    hive> LOAD DATA LOCAL INPATH './examples/files/kv2.txt
' OVERWRITE INTO TABLE invites PARTITION (ds='2008-08-15')
;
    hive> LOAD DATA LOCAL INPATH './examples/files/kv3.txt
' OVERWRITE INTO TABLE invites PARTITION (ds='2008-08-08')
;
```

3. Operaciones SQL

```
    hive> SELECT a.foo FROM invites a WHERE a.ds='2008-08-
15';
    hive> INSERT OVERWRITE DIRECTORY '/home/<usuario>/tmp/
hdfs_out' SELECT a.* FROM invites a WHERE a.ds='2008-08-15
';
```

Esta última operación guarda los registros de invites en la carpeta HDFS `~/tmp/hdfs_out`. Ahora crearemos una tabla events, donde almacenaremos los conteos por valor de la tabla `invites`.

```
    hive> CREATE TABLE events (foo STRING, bar STRING);
```

```
    hive> INSERT OVERWRITE TABLE events SELECT a.bar, coun
t(*) FROM invites a WHERE a.foo > 0 GROUP BY a.bar;
    hive> SELECT * FROM events;
```

Para una documentación completa, consulte Hive DML

Taller

Vamos a cargar el dataset MovieLens. MovieLens es un dataset recolectado por la Universidad de Minnesota, que consta de:

- 100,000 ratings (1-5) de 943 usuarios en 1682 películas.
- Cada usuario ha calificado al menos 20 películas
- Se brinda información básica de las características demográficas de los usuarios(edad, género, ocupación, zip code)

1. Descargue el dataset desde: ML_100K.
2. Descomprima el archivo y dele un vistazo a los datos. Abra el archivo `u.data`.

```
196    242    3    881250949
186    302    3    891717742
22    377    1    878887116
244    51    2    880606923
166    346    1    886397596
298    474    4    884182806
...
```

Este archivo está compuesto por 100K registros donde la primera columna corresponde al id del usuario, la segunda al id de la película, la tercera a la calificación asignada por el usuario y la cuarta columna corresponde a la fecha en tiempo de Unix de la calificación.

3. Desde el interprete de comandos de Hive, cree la tabla que guardará el dataset.

```
hive> CREATE TABLE u_data (
userid INT,
movieid INT,
rating INT,
unixtime STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE;
```

4. Cargue los datos a Hive

```
hive> LOAD DATA LOCAL INPATH '/home/mlds/Descargas/ml-100k/u.data'
        OVERWRITE INTO TABLE u_data;
```

5. Verifique que se hayan cargado los 100K registros

```
hive> SELECT COUNT(*) FROM u_data;
...
OK
```

```
100000
Time taken: 1.367 seconds, Fetched: 1 row(s)
```

6. La fecha en tiempo de Unix no es humanamente legible, por lo que vamos a usar python para encontrar cual fue la fecha exacta en la que se presentó la calificación. Vamos a crear un archivo ue se llame `datetime_mapper.py` y lo guardaremos en la **carpeta de instalación de Hive**.

```
import sys
import datetime
for line in sys.stdin:
 line = line.strip()
 userid, movieid, rating, unixtime = line.split('\t')
 weekday = datetime.datetime.fromtimestamp(float(unixt
ime)).strftime("%Y-%m-%d")
 print '\t'.join([userid, movieid, rating, str(weekday
)])
```

7. Ahora crearemos una nueva tabla en Hive y añadiremos esta función de **python** como parte de una de la creación del nuevo dataset:

```
hive> CREATE TABLE u_data_new (
userid INT,
movieid INT,
rating INT,
weekday STRING)
```

```
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY '\t';
```

Luego añadimos:

```
    hive> add FILE datetime_mapper.py;
```

Finalmente:

```
    hive> INSERT OVERWRITE TABLE u_data_new
    SELECT
    TRANSFORM (userid, movieid, rating, unixtime)
    USING 'python datetime_mapper.py'
    AS (userid, movieid, rating, weekday)
    FROM u_data;
```

8. Verificamos que haya sido cargada de forma correcta:

```
    hive> SELECT weekday, COUNT(*)
    FROM u_data_new
    GROUP BY weekday;
```