HIVE - EJERCICIO 2

Insertando datos

Para insertar datos en las tablas de *Hive* podemos hacerlo de varias formas:

- Cargando los datos mediante sentencias `LOAD DATA`.
- Insertando los datos mediante sentencias `INSERT`.
- Cargando los datos directamente mediante *Sqoop* o alguna herramienta similar.

Cargando datos

Para cargar datos se utiliza la sentencia `LOAD DATA`. Si quisiéramos volver a cargar los datos desde HDFS utilizaremos:

```
LOAD DATA INPATH '/user/iabd/hive/customer'
  overwrite into table customers;
```

Si en cambio vamos a cargar los datos desde un archivo local a nuestro sistema de archivos añadiremos la *keyword* `LOCAL`:

```
LOAD DATA LOCAL INPATH '/home/cloudera/workspace/clients.csv'
  overwrite into table customers;
```

Insertando datos

Aunque podemos insertar datos de forma atómica (es decir, registro a registro mediante `INSERT INTO [TABLE] ... VALUES`), realmente las inserciones que se realizan en *Hive* se hacen a partir de los datos de otras tablas mediante el comando *insert-select* a modo de ETL:

```
INSERT INTO destino
  SELECT col1, col2 FROM fuente;
```

```
INSERT OVERWRITE destino
 SELECT col1, col2 FROM fuente;
```

Mediante la opción `OVERWRITE`, en cada ejecución se vacía la tabla y se vuelve a rellenar. Si no lo indicamos o utilizamos `INTO`, los datos se añadirían a los ya existentes.

Por ejemplo, vamos a crear una nueva tabla de clientes, pero la vamos a almacenar en formato Parquet:

```
CREATE TABLE customersp
(
 Name STRING COMMENT "Nombre",
 Situation STRING COMMENT "Tipo de cliente",
 country STRING
)
STORED AS PARQUET;
```

Y a continuación la cargamos con los datos de la tabla `customers`:

```
INSERT INTO customersp SELECT * FROM customers;
```

Prueba a realizar una consulta que cuente la cantidad de clientes en cada una de las tablas ¿Cuál tarda menos? ¿Por qué?

Modificando datos

Acabamos de añadir datos, y en teoría podemos realizar operaciones `UPDATE` y `DELETE` sobre las filas de una tabla.

HDFS no se diseñó pensando en las modificaciones de archivos, con lo que los cambios resultantes de las inserciones, modificaciones y borrados se almacenan en archivos delta. Por cada transacción, se crea un conjunto de archivo delta que altera la tabla (o partición). Los ficheros delta se fusionan periódicamente con los ficheros base de las tablas mediante *jobs MapReduce* que el metastore ejecuta en *background*.

Para poder modificar o borrar los datos, Hive necesita trabajar en un contexto transaccional, por lo que necesitamos activar las siguientes variables:

```
set hive.support.concurrency=true;

set hive.enforce.bucketing=true;

set hive.exec.dynamic.partition.mode=nonstrict;

set hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
```

Una vez configurado, vamos a crear las tablas con el formato ORC y organizar los datos mediante *buckets*. Por ejemplo, volvemos a crear la tabla de clientes:

```
CREATE TABLE customerstx

(

 Name STRING,

 Situation STRING,

 country STRING

)

CLUSTERED BY (country) INTO 4 BUCKETS

STORED AS ORC

TBLPROPERTIES ('transactional' = 'true');
```

Una vez creada la tabla, la vamos a cargar con los datos de los clientes:

```
INSERT INTO customerstx

  SELECT * FROM customers;
```

Un par de minutos después, con los datos datos ya cargados, ya podemos cambiar lo que consideremos:

```
UPDATE customerstx SET Situation="mejorable" WHERE Situation="regular";
```

Extrayendo datos insertados

Combinando los comandos de HQL y HDFS podemos extraer datos a ficheros locales o remotos:

```
# Añadiendo contenido local

hive -e "use iabd; select * from customers" >> prueba1
```

```
# Sobrescribiendo contenido local

hive -e "use iabd; select * from customers" > prueba2

# Añadiendo contenido HDFS

hive -e "use iabd; select * from customers" | hdfs dfs --appendToFile /tmp/prueba3

# Sobrescribiendo contenido

hive -e "use iabd; select * from customers" | hdfs dfs --put -f /tmp/prueba4
```

Si indicamos la propiedad `set hive.cli.print.header=true` antes de la consulta, también nos mostrará el encabezado de las columnas. Esto puede ser útil si queremos generar un csv con el resultado de una consulta:

```
hive -e 'use iabd; set hive.cli.print.header=true; select * from customers' > fichero.csv
```

Mediante `INSERT LOCAL DIRECTORY` podemos escribir el resultado de una consulta en nuestro sistema de archivos, fuera de HDFS. El problema es que si hay muchos datos creará múltiples ficheros y necesitaremos concatenarlos para tener un único resultado:

```
insert overwrite local directory '/home/iabd/datos'

row format delimited

fields terminated by ','

select * from customers;
```