Creación de tablas en formato texto

Crear base de datos

```
Create DATABASE datos_padron;
```

Crear tabla almacenada en formato texto y cargar datos desde fichero local

```
CREATE TABLE IF NOT EXISTS padron_txt(

COD_DISTRITO int,

DESC_DISTRITO STRING,

COD_DISTRITO_BARRIO int,

DESC_BARRIO STRING,

COD_BARRIO int,

COD_DIST_SECCION int,

COD_SECCION int,

COD_EDAD_INT int,

EspanolesHombres int,

EspanolesMujeres int,

ExtranjerosHombres int,

ExtranjerosMujeres int)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES (

   "separatorChar" = ";"

)

STORED AS TEXTFILE

TBLPROPERTIES ("skip.header.line.count"="1");


LOAD DATA LOCAL INPATH
'/media/sf_CarpetaCompartidaFundBigData/Rango_Edades_Seccion_202208.csv' INTO TABLE
datos_padron.padron_txt;
```

Crear tabla padron_txt_2 con CTAS eliminando espacios en blanco de las columnas pertinentes

```
CREATE TABLE IF NOT EXISTS padron_txt_2

row format DELIMITED
```

FIELDS TERMINATED BY ';'

STORED AS TEXTFILE

as SELECT cod_distrito, trim(desc_distrito) as desc_distrito, cod_distrito_barrio, trim(desc_barrio) as desc_barrio, cod_barrio, cod_dist_seccion, cod_seccion, cod_edad_int, espanoleshombres,espanolesmujeres, extranjeroshombres, extranjerosmujeres FROM padron_txt;

Utilidad del comando LOCAL en la sentencia LOAD DATA

Al especificar la opción LOCAL debemos asegurarnos que el archivo que estamos especificando en la ruta está en nuestro sistema de almacenamiento local y no en HDFS o DBFS

Creación de nueva tabla a partir de padron_txt que tengo los valores vacios sustituidos por 0

CREATE TABLE IF NOT EXISTS padron_txt_3

row format DELIMITED

FIELDS TERMINATED BY ';'

STORED AS TEXTFILE

as SELECT cod_distrito, desc_distrito, cod_distrito_barrio, desc_barrio, cod_barrio, cod_dist_seccion, cod_seccion, cod_edad_int, CASE WHEN LENGTH(espanoleshombres) > 0 THEN espanoleshombres ELSE 0 END AS espanoleshombres, CASE WHEN LENGTH(espanolesmujeres) > 0 THEN espanolesmujeres ELSE 0 END AS espanolesmujeres, CASE WHEN LENGTH(extranjeroshombres) > 0 THEN extranjeroshombres ELSE 0 END AS extranjeroshombres, CASE WHEN LENGTH(extranjerosmujeres) > 0 THEN extranjerosmujeres ELSE 0 END AS extranjerosmujeres FROM padron_txt;

Nueva creación de padron_2_txt usando Regex SerDe y creación de padron_txt_4 a partir de padron_2_txt sustituyendo nulos

CREATE TABLE IF NOT EXISTS padron_txt_2(

COD_DISTRITO int,

DESC_DISTRITO STRING,

COD_DISTRITO_BARRIO int,

DESC_BARRIO STRING,

COD_BARRIO int,

COD_DIST_SECCION int,

COD_SECCION int,

COD_EDAD_INT int,

```
EspanolesHombres int,

EspanolesMujeres int,

ExtranjerosHombres int,

ExtranjerosMujeres int)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'

WITH SERDEPROPERTIES (

   "input.regex" = '"(\\d+)";"(\\w+(?:[ \\-]+\\w+)*)\\s*";"(\\d+)";"(\\w+(?:[ \\-
]+\\w+)*)\\s*";"(\\d+)";"(\\d+)";"(\\d+)";"(\\d+)";"(\\d*)";"(\\d*)";"(\\d*)";"(\\d*)"'

)

STORED AS TEXTFILE

TBLPROPERTIES ("skip.header.line.count"="1");

LOAD DATA LOCAL INPATH
'/media/sf_CarpetaCompartidaFundBigData/Rango_Edades_Seccion_202208.csv' INTO TABLE
datos_padron.padron_txt_2;


CREATE TABLE IF NOT EXISTS padron_txt_4

row format DELIMITED

FIELDS TERMINATED BY ';'

STORED AS TEXTFILE

as SELECT cod_distrito, desc_distrito, cod_distrito_barrio, desc_barrio, cod_barrio, cod_dist_seccion,
cod_seccion, cod_edad_int, nvl(espanoleshombres, 0) AS espanoleshombres, nvl(espanolesmujeres,
0) AS espanolesmujeres, nvl(extranjeroshombres,0) AS extranjeroshombres, nvl(extranjerosmujeres,
0) AS extranjerosmujeres FROM padron_txt_2;
```

Investigación del formato columnar Parquet

¿Qué es CTAS?

Es la creación de una tabla a partir de una sentencia select a otra tabla o vista ya esxistente

Creación de tabla padron_parquet, almacenada en formato parquet, a partir de padron_txt_3

```
CREATE TABLE IF NOT EXISTS padron_parquet

row format DELIMITED

FIELDS TERMINATED BY ','

STORED AS PARQUET
```

as SELECT * FROM padron_txt_3;

Creación de tabla padron_parquet_2, a partir de padron_txt_4

CREATE TABLE IF NOT EXISTS padron_parquet_2

row format DELIMITED

FIELDS TERMINATED BY ','

STORED AS PARQUET

as SELECT * FROM padron_txt_4;

Investigar formato columnar parquet y ventajas

https://www.databricks.com/glossary/what-is-parquet

- Más rapido de hacer queris filtrando por columna al estar almacenado en formato columnar en vez de en formato fila
- Menos espacio al estar comprimido
- El schema se almacena junto a los datos

Comparar el tamaño de los ficheros de las tablas padron_txt_3(con valores vacíos sustituidos y espacios innecesarios), padron_txt_4(con valores vacíos sustituidos y sin espacios innecesarios), padron_parquet(Creada a partir de padron_txt_3), padron_parquet_2(creada a partir de padron_txt_4)

Padron_txt_3: 16.2 MB

Padron_txt_4: 11.9 MB

Padron_parquet: 857.2KB

Padron_parquet_2: 842.3 KB

Jugando con Impala

Qué es Impala

Es la alternativa a Hive, creada por cloudera cuyo principal objetivo es superar la lentitud de las consulatas de hive, destaca por su alto rendimiento y su baja latencia.

Diferencias con Hive

Es mas rapido que hive a la hora de realizar consultas, ya que no utiliza MapReduce y almacena en memoria. Tiene mayor rendimiento que hive pero está pensado para queries cortas en grandes conjuntos de datos. No es tolerante a datos.

Comando INVALIDATE METADATA

Marca los metadatos de la tabla o base de datos como viejos y la próxima vez que se realice una operación sobre ellos se recargan. Operación bastante costosa, preferible usar refresh si lo unico que se quiere es recargar los metadatos.

Casos de uso de INVALIDATE METADATA:

- Metadata of existing tables changes.
- New tables are added, and Impala will use the tables.
- The SERVER or DATABASE level Ranger privileges are changed.
- Block metadata changes, but the files remain the same (HDFS rebalance).
- UDF jars change.
- Some tables are no longer queried, and you want to remove their metadata from the catalog and coordinator caches to reduce memory requirements.

Usar INVALIDATE METADATA sobre datos_padron

INVALIDATE METADATA datos_padron.[table_name]

Calcular el total de espanoleshombres, espanolesmujeres, extranjeroshombres y extranjerosmujeres agrupados por desc_ditrito y desc_barrio

```
select desc_distrito, desc_barrio, sum(espanoleshombres), sum(espanolesmujeres),
sum(extranjeroshombres), sum(extranjerosmujeres)

from padron_parquet_2

group by desc_barrio, desc_distrito
```

Llevar a cabo la query anterior usando hive sobre padron_txt_4 y sobre padron_parquet_2

Padron_parquet_2: tiempo ejecución 24.57s

Padron_txt_4: tiempo de ejecución 32.26s

Aunque los tiempos de ejecución puedan variar para la misma query, a grandes rasgos es un poco más veloz la ejecución sobre la tabla almacenada en parquet

Llevar a cabo la query sobre las tablas mencionadas anteriormente usando Impala

Padron_txt_4: 1.15s

Padron_parquet_2: <1s

Al igual que en hive, la ejecución sobre la tabla almacenada en parquet es más rapida que sobre la tabla almacenada texto

Diferencia de rendimiento entre Hive e Impala en la query anterirmente ejecutada

Claramente la ejecución usando Impala es 20 veces más rápida que usando hive

Tablas particionadas

Crear tabla (hive) padron_particionado particionando por desc_distrito y desc_barrio en formato parquet

```
CREATE TABLE IF NOT EXISTS padron_particionado(

COD_DISTRITO int,

COD_DISTRITO_BARRIO int,

COD_BARRIO int,

COD_DIST_SECCION int,

COD_SECCION int,

COD_EDAD_INT int,

EspanolesHombres int,

EspanolesMujeres int,

ExtranjerosHombres int,

ExtranjerosMujeres int)

Partitioned by (DESC_DISTRITO STRING, DESC_BARRIO STRING)

ROW FORMAT delimited

FIELDS TERMINATED BY ','

STORED AS PARQUET;
```

Insertar datos (en cada particion) dinámicamente (Hive) a partir de un select de la tabla padron_parquet_2

```
SET hive.exec.dynamic.partition = true;

SET hive.exec.dynamic.partition.mode = non-strict;

INSERT INTO padron_particionado PARTITION(desc_barrio, desc_distrito) SELECT * FROM padron_parquet_2;
```

En caso de error relacionado con el numero de particiones:

set hive.exec.max.dynamic.partitions=500000;

set hive.exec.max.dynamic.partitions.pernode=500000;

Invalidate Metadata de padron_particionado desde Impala

INVALIDATE METADATA datos_padron.padron_particionado

Calcular el total de espanoleshombres, espanolesmujeres, extranjeroshombres y extranjerosmujeres agrupados por desc_ditrito y desc_barrio para los distritos CENTR, LATINA, CHAMARTIN, TETUAN, VICALVARO y BARAJAS

select desc_distrito, desc_barrio, sum(espanoleshombres), sum(espanolesmujeres), sum(extranjeroshombres), sum(extranjerosmujeres)

from padron_parquet

where desc_distrito in ('CENTRO','LATINA','CHAMARTIN','TETUAN','VICALVARO','BARAJAS')

group by desc_barrio, desc_distrito

Llevar a cabo la query anterion en hive para las tablas padron_parquet_2 y padron_particionado

Padron_parquet: 24.21-30.47s

Padron_particionado: 24.20s

El rendimiento es similar, pero parece que es más rápida cuando se ejecuta sobre padron_parquet

Llevar a cabo la query en Impala para las tablas anteriormente mencionadas

Padron_parquet: 0.5-0.8s

Padron_particionado: 0.5-0.9s

El rendimiento es muy similar, pero parece que es más rápida cuando se ejecuta sobre padron_parquet

Hacer consultas de agregacion con las tablas padron_txt_4, padron_parquet_2 y padron_particionado y comparar rendimientos tanto en Hive como en Impala

select desc_distrito, desc_barrio, count(espanoleshombres), min(espanolesmujeres), max(extranjeroshombres), avg(extranjerosmujeres)

from padron_particionado

where desc_distrito in ('CENTRO','LATINA','CHAMARTIN','TETUAN','VICALVARO','BARAJAS')

group by desc_barrio, desc_distrito;

Comparativa	Padron_txt_4	Padron_parquet_2	Padron_particinado
Hive	25.33	24.33	24.68
Impala	0.9	<0.5	<0.5

Impala sigue siendo mucho más rápido que Hiva a la hora de ejecutar querys. El hecho de tener una tabla particionada no consigue una mejora de rendimiento con respecto a una tabla no particionada almacenada en el mismo formato, al menos para este tipo de queries, es más, es incluso un poco peor pero la diferencia es casi inapreciable para este dataset.

Tablas HDFS

Crear documento de texto datos1.txt

Vim datos1.txt

1,2,3

4,5,6

7,8,9

Crear documento de texto datos2.txt

Vim datos2.txt

12,3,45

6,78,9

1,23,4

Crear directorio en HDFS

hdfs dfs –mkdir /test

Mueve tu fichero datos1 al directorio hdfs que has creado

hdfs dfs -put /media/sf_CarpetaCompartidaFundBigData/datos1.txt /test/datos1.txt

Desde Hive crear base de datos (numeros) y crear tabla que no sea externa para albergar los datos de los ficheros anteriormente creados

```
create database numeros;

CREATE table IF NOT EXISTS numeros_tbl (n1 INT, n2 INT, n3 INT)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ',';
```

Carga los datos desde el fichero datos1 almacenado en HDFS. Consulta la localización donde estaban anteriormente los datos ¿Siguen ahí?¿Dónde están? Borra la tabla ¿Qué ha ocurrido con los datos almacenados en HDFS?

LOAD DATA INPATH '/test/datos1.txt' INTO TABLE numeros.numeros_tbl;

No, el fichero de texto original ya no se encuentra en la localización de origen.

Ahora se encuentra en la carpeta de HDFS correspondiente a la tabla números (/user/hive/warehouse/numeros.db/numeros_tbl/datos1.txt)

Drop table numeros_table;

Han desaparecido completamente, tanto del directorio original como del directorio de la tabla, que tampoco existe ya al haberla borrado.

Vuelve a mover el fichero al directorio anterior de HDFS

hdfs dfs -put /media/sf_CarpetaCompartidaFundBigData/datos1.txt /test/datos1.txt

Desde hive, crea una tabla externa sin location y carga los datos en ella. ¿A dónde han ido a parar los datos en HDFS? Borra la tabla. ¿Qué ocurre con los datos de HDFS?

CREATE external table IF NOT EXISTS numeros_tbl (n1 INT, n2 INT, n3 INT)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ',';

Ahora se encuentra en la carpeta de HDFS correspondiente a la tabla números (/user/hive/warehouse/numeros.db/numeros_tbl/datos1.txt)

DROP table numeros_tbl;

Los datos se han quedado en la carpeta de hdfs correspondiente a la tabla de hive donde estaban;

Borra el fichero del directorio en el que está. Vuelve a insertarlos en el directorio original (/test). Vuelve a crear la tabla de manera externa, pero con el parámetro location apuntando al directorio en el que están. No cargues los datos de manera específica y haz una consulta sobre la tabla recién creada. ¿Tiene algún contenido?

CREATE external table IF NOT EXISTS numeros_tbl (n1 INT, n2 INT, n3 INT)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LOCATION '/test';

Tiene los contenidos correspondientes al fichero datos1

Inserta el fichero datos2 en el mismo directorio que datos1. Vuelve a hacer la consulta anterior sobre la tabla. ¿Qué salida muestra la tabla?

Muestra los datos combinados de los ficheros datos1 y datos2

Extrae conclusiones de todos los apartados anteriores

La diferencia principal entre la creación de una tabla interna y de una externa es, qué al ser interna, Hive gestiona completamente la tabla, es decir que si borramos la tabla también se borra de nuestro directorio HDFS, mientras que si es externa no se eliminan los datos de HDFS si eliminamos la tabla desde Hive.

En lo referente a incluir o no el parámetro location en la creación de la tabla, si lo incluimos, la tabla toma como directorio de almacenamiento de los datos el directorio especificado, haciendo la carga automáticamente la carga de los datos que en ese directorio se alojan, incluso cuando metemos un nuevo fichero en el directorio. Intuyo que los ficheros de dicho directorio tienen que respetar las columnas y separadores especificados en la creación de la tabla.

Un poco de Spark

La realización de estos ejercicios está en el documento adjunto padron.scala