**Práctica Hive**

1. **A partir de los datos (CSV) de Padrón de Madrid (**[**https://datos.madrid.es/egob/catalogo/200076-1-padron.csv**](https://datos.madrid.es/egob/catalogo/200076-1-padron.csv)**) llevar a cabo lo siguiente:**
   1. **Crear Base de datos datos\_padron**

DROP DATABASE datos\_padron CASCADE;

CREATE DATABASE IF NOT EXISTS datos\_padron;

USE datos\_padron;

* 1. **Crear tabla padron\_txt con todos los campos del fichero CSV y cargar los datos mediante el comando LOAD DATA LOCAL INPATH. La tabla tendrá formato texto y tendrá como delimitador de campo el carácter ';' y los campos estarán encerrados en comillas dobles '”’ y se deberá omitir la cabecera del fichero de datos al crear la tabla.**

-- Creamos la tabla

DROP TABLE IF EXISTS datos\_padron.padron\_txt;

CREATE TABLE IF NOT EXISTS datos\_padron.padron\_txt(

COD\_DISTRITO INT,

DESC\_DISTRITO STRING,

COD\_DIST\_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' -- Utilizamos SerDe para posteriormente cargar el archivo CSV

WITH SERDEPROPERTIES

(-- Propiedades para que delimite cada campo entre ';' , lea cada campo encerrado entre "" , y lea el salto de linea

"separatorChar" = ";",

"quoteChar" = "\"",

"escapeChar" = "\n"

)

STORED AS TEXTFILE --Tabla tipo txt

TBLPROPERTIES("skip.header.line.count"="1", -- Propieda para tratar los campos con espacio como null

"seriealization.null.format" = ""); -- Propiedad para omitir encabezado

-- Cargar los datos desde local (Ejecutar en la shell con usuario root, en HUE no funciona)

LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/Datasets HIVE/data.csv'

OVERWRITE INTO TABLE datos\_padron.padron\_txt;

--verificamos

select \* from padron\_txt limit 5;

Otra forma de cargar los datos directamente sería crearla como tabla externa y apuntar en location hacia el directiorio donde está el archivo CSV

* 1. **Crear tabla padron\_txt2 que haga trim sobre los datos de algunas columnas con la finalidad de eliminar los espacios en blanco innecesarios. (A través de CTAS -> crea una tabla a partir de la estructura de un select).**

-- Creamos la tabla elminando los espacios en blanco con trim, no es necesario el SerDe puesto que hacemos un select de la tabla anterior con los datos csv cargados

DROP TABLE IF EXISTS padron\_txt2

CREATE TABLE padron\_txt2

AS

SELECT

COD\_DISTRITO,

trim(DESC\_DISTRITO) DESC\_DISTRITO, -- trim(col) elimina espacios

COD\_DIST\_BARRIO,

trim(DESC\_BARRIO) DESC\_BARRIO, -- trim(col) elimina espacios

COD\_BARRIO,

COD\_DIST\_SECCION,

COD\_SECCION,

COD\_EDAD\_INT,

EspanolesHombres,

EspanolesMujeres,

ExtranjerosHombres,

ExtranjerosMujeres

FROM padron\_txt;

-- verificamos

select \* from padron\_txt2 limit 5;

-- TENEMOS QUE CONVERTIR LAS COLUMNAS NECESARIAS A INT POR QUE EL SERDE LAS PASA TODAS A STRING. Creamos tabla padron\_txt3 casteando a mano todas las columnas

-- Supongo que en versiones superiores de hive o con otros softwares como spark no sería necesario

DROP TABLE IF EXISTS padron\_txt3

CREATE TABLE padron\_txt3

STORED AS TEXTFILE --Tabla tipo txt

AS

SELECT

CAST(COD\_DISTRITO AS INT) COD\_DISTRITO,

trim(DESC\_DISTRITO) DESC\_DISTRITO,

CAST(COD\_DIST\_BARRIO AS INT) COD\_DIST\_BARRIO ,

trim(DESC\_BARRIO) DESC\_BARRIO,

CAST(COD\_BARRIO AS INT) COD\_BARRIO,

CAST(COD\_DIST\_SECCION AS INT) COD\_DIST\_SECCION,

CAST(COD\_SECCION AS INT) COD\_SECCION,

CAST(COD\_EDAD\_INT AS INT) COD\_EDAD\_INT,

CAST(EspanolesHombres AS INT) EspanolesHombres,

CAST(EspanolesMujeres AS INT) EspanolesMujeres,

CAST(ExtranjerosHombres AS INT) ExtranjerosHombres,

CAST(ExtranjerosMujeres AS INT) ExtranjerosMujeres

FROM padron\_txt2; -- apuntamos a txt2 para coger la tabla con espacios ya eliminados

* 1. **Investigar y entender la diferencia de incluir la palabra LOCAL en el comando LOAD DATA**

La palabra LOCAL DENTRO DE LOAD DATA INPATH indica que la carga de datos/archivo se hará desde el sistema de archivos local, si no se especifica, la carga de datos/archivo se hará desde HDFS.

1. **¿Qué es CTAS?**
   1. **Crear Tabla (en Hive) padron\_parquet (cuyos datos serán almacenados en el formato columnar parquet) a partir de la tabla padron\_txt mediante un CTAS.**

-- Introducimos el formato columnar parquet en STORED AS. La hacemos referenciando a padron\_txt3 que no tiene espacios y tiene bien delimitadas las columnas

DROP TABLE IF EXISTS padron\_parquet

CREATE TABLE padron\_parquet

STORED AS PARQUET --Tabla tipo parquet

AS

SELECT \*

FROM padron\_txt3; --apuntamos a txt3 para coger los datos limpios (sin espacios y columnas bien delimitadas

* 1. **Crear Tabla (en Hive) padron\_parquet2 (cuyos datos serán almacenados en el formato columnar parquet) a partir de la tabla padron\_txt2 mediante un CTAS.**

-- Realmente no necesario porque con padron\_parquet ya cargamos los datos limpios

DROP TABLE IF EXISTS padron\_parquet2

CREATE TABLE padron\_parquet2

STORED AS PARQUET --Tabla tipo parquet

AS

SELECT \*

FROM padron\_txt3; --utilizo parquet\_txt3 porque tiene las columnas int bien definidas

* 1. **Investigar en que consiste el formato columnar parquet y las ventajas de trabajar con este tipo de formatos.**

**Parquet** es un formato de almacenamiento basado en columnas para Hadoop mientras que **Avro** es un formato basado en filas. Parquet es más utilizado cuando los datos tienen muchas columnas, su uso esta generalizado a trabajar con un subconjunto de estas columnas en lugar de todos los registros completos.

* 1. **Comparar el tamaño de los ficheros de los datos de las tablas padron\_txt (CSV) y padron\_parquet (alojados en hdfs cuya ruta se puede obtener de la propiedad location de cada tabla por ejemplo haciendo SHOW CREATE TABLE).**

Los ficheros almacenados en HDFS de la tabla padron\_txt ocupan 22mb porque es en el que cargamos desde local el archivo csv, los que hacen referencia a la tabla padron\_txt2 (en la que eliminamos los espacios en blanco con la función trim) ocupan 11.6mb y los que hacen referencia a la tabla padron\_parquet ocupan solo 1mb. El formato de columnas parquet tiene mucha más eficiencia a la hora de comprimir.

* 1. **Comparar el tamaño de los ficheros de los datos de las tablas padron\_txt (CSV), padron\_txt2, padron\_parquet y padron\_parquet2 (alojados en hdfs cuya ruta se puede obtener de la propiedad location de cada tabla por ejemplo haciendo SHOW CREATE TABLE).**
* Padron\_txt 22mb
* Padron\_txt2 11.6mb
* Padron\_parquet 931.8kb
* Padron\_parquet2 929.7kb

Utilizamos el visor de archivos de HDFS de la izquierda. Como hemos comentado anteriormente, parquet optimiza mucho más la comprensión y por ello hace que ocupe alrededor de 1mb. La diferencia entre padron\_parquet (931.8kb) y paddron\_parquet2 (929.7kb) estaría en que la segunda hace refencia a la tabla en la que se han eliminado los espacios en blanco con la función trim (padron\_txt2).

1. **Impala**
   1. **¿Qué es impala?**

Es un motor de consultas de bbdd diseñado para optimizar la latencia de las consultas SQL en Hadoop, es una alternativa a Hive para consultas menos pesadas.

Impala ejecuta las consultas directamente en el cluster en lugar de ejecutar un MapReduce para procesar**, cargando los datos en memoria y permaneciendo allí toda la fase de consulta**, lo que le hace ser más rápido y efectivo que Hive.

Impala requiere que la mayor parte de los datos quepan en la memoria principal, servidores con más capacidad de memoria que los de hadoop en mapreduce. Si un nodo cae, la consulta no puede lanzarse y lanza un error. Se recomienda para consultas rápidas y que se puedan reiniciar.

* 1. **¿En qué se diferencia a Hive?**

Hive e Impala se suelen utilizar juntas en proyectos. Las principales diferencias radican en Impala no soporta algunas funcionalidades que Hive sí, aunque poco a poco las van implementando. Algunas de ellas pueden ser:

* El tipo de datos date
* Funciones XML y JSON, algunas funciones de agregación, UDFs
* Sampling (ejecutar querys sobre una muestra de la tabla)
  1. **Comando INVALIDATE METADATA, ¿en qué consiste?**

Marca los metadatos de una o todas las tablas como obsoletos, por lo que la próxima vez que Impala realiza una consulta en una tabla con este comando aplicado, vuelve a cargar los metadatos antes de continuar con la consulta.

Se trata de una operación muy costosa en comparación con la actualización incremental de datos con REFRESH, por lo que tendremos que elegir entre estas dos para actualizar los metadatos.

* 1. **Hacer invalidate metadata en Impala de Base de datos datos\_padron**

INVALIDATE METADATA

* 1. **Calcular el total de EspanolesHombres, EspanolesMujeres, ExtranjerosHombres y ExtranjerosMujeres agrupado por DESC\_DISTRITO y DESC\_BARRIO.**
     1. Llevar a cabo la consulta en **Hive** en las tablas padron\_txt y padron\_parquet. ¿Alguna conclusión?

-- Select de tabla parquet o txt (cambiar nombre en FROM)

SELECT DESC\_DISTRITO,

DESC\_BARRIO,

SUM(espanolesHombres) Tot\_Esp\_H,

SUM(espanolesMujeres) Tot\_Esp\_M,

SUM(extranjerosHombres) Tot\_Ext\_H,

SUM(extranjerosMujeres) Tot\_Ext\_M

FROM padron\_txt

GROUP BY DESC\_DISTRITO, DESC\_BARRIO;

Query padron\_txt -> Time taken: **34.971 seconds**, Fetched: 133 row(s)

Query padron\_parquet -> Time taken: **22.654 seconds**, Fetched: 133 row(s)

Se observa que para el formato de tabla columnar parquet HIVE es más rapido procesando la query. (35segs en padron\_txt y 23 segs en padron\_parquet)

* + 1. **Llevar a cabo la consulta en Impala en las tablas padron\_txt y padron\_parquet. ¿Alguna conclusión?**

Impala no soporta el formato SerDe aplicado en la tabla padron\_txt por lo que esta consulta solo se podría hacer en Hive.

Por otro lado, la tabla padron\_parquet esta creada con CTAs a través de la tabla padron\_txt. Esto implica que el SerDe al ser aplicado ha transformado todas las columnas a tipo STRING, por lo que para poder lanzar esta query en impala habría que pasar dichas columnas a tipo INT **(creadas ya en tablas padron\_txt3 y padron\_parquet2).**

**Si realizamos las dos consultas se nota una mejora de tiempo de ejecución en la query de impala sobre hive**

* + 1. **¿Se percibe alguna diferencia de rendimiento entre Hive e Impala?**

Impala aun **no soporta algunas funcionalidades que Hive sí**, pero es más **rápido procesando querys**, por lo que es recomendable usar Hive salvo que sepas que esa query está bien optimizada en Impala.

* + 1. **Obtener alguna métrica más adicional que nos sean de interés (Sitios con pocos empadronados, que distrito o barrio con más extranjeros españoles, etc.**

-- Valores medios en Extranjeros y Españoles agrupado por distrito y barrio

SELECT DESC\_DISTRITO,

DESC\_BARRIO,

AVG(espanolesHombres) EspanolesHombres,

AVG(espanolesMujeres) EspanolesMujeres,

AVG(extranjerosHombres) ExtranjerosHombres,

AVG(extranjerosMujeres) ExtranjerosMujeres

FROM padron\_parquet2

GROUP BY DESC\_DISTRITO, DESC\_BARRIO;

1. **Particionamiento**
   1. **Crear tabla (Hive) padron\_particionado particionada por los campos DESC\_DISTRITO y DESC\_BARRIO cuyos datos estén en formato parquet.**

-- Creamos tabla padron\_particionado y padron\_particionado\_txt (cambiar formato y nombre)

DROP TABLE IF EXISTS padron\_particionado\_txt;

CREATE TABLE padron\_particionado\_txt --Cambiamos nombre (txt o parquet)

(

COD\_DISTRITO INT,

COD\_DIST\_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 )

STORED AS TEXTFILE; -- Cambiamos formato (PARQUET O TEXTFILE)

* 1. **Insertar datos (en cada partición) dinámicamente (con Hive) en la tabla recién creada a partir de un select de la tabla padron\_parquet2.**

-- seteamos estas propiedades que permite realizar el particionado dinamico

set hive.exec.dynamic.partition = true;

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

set hive.exec.max.dynamic.partitions = 10000;

set hive.exec.max.dynamic.partitions.pernode = 1000;

-- seteamos mas meroia ram (por defecto la configuracion de hadoop trae 1gb)

set mapreduce.map.memory.mb = 2048;

set mapreduce.reduce.memory.mb = 2048;

set mapreduce.map.java.opts=-Xmx1800m

-- Insertamos a traves de los datos de la tabla padron\_txt (formato txt)

--Importante el orden de las columnas

INSERT OVERWRITE TABLE padron\_particionado\_txt -- cambiar nombre tabla

PARTITION (DESC\_DISTRITO, DESC\_BARRIO)

SELECT

COD\_DISTRITO,

COD\_DIST\_BARRIO,

COD\_BARRIO,

COD\_DIST\_SECCION,

COD\_SECCION,

COD\_EDAD\_INT,

EspanolesHombres,

EspanolesMujeres,

ExtranjerosHombres,

ExtranjerosMujeres,

DESC\_DISTRITO,

DESC\_BARRIO

FROM padron\_txt3; -- cambiar tabla a txt o parquet

-- Insertamos a traves de los datos de la tabla padron\_parquet2 (formato parquet)

INSERT OVERWRITE TABLE padron\_particionado\_parquet

PARTITION (DESC\_DISTRITO, DESC\_BARRIO)

SELECT

COD\_DISTRITO,

COD\_DIST\_BARRIO,

COD\_BARRIO,

COD\_DIST\_SECCION,

COD\_SECCION,

COD\_EDAD\_INT,

EspanolesHombres,

EspanolesMujeres,

ExtranjerosHombres,

ExtranjerosMujeres,

DESC\_DISTRITO,

DESC\_BARRIO

FROM padron\_parquet;

-- verificamos las particiones (las vemos en el browser tambien en wharehouse)

SHOW PARTITIONS padron\_particionado\_parquet;

SHOW PARTITIONS padron\_particionado\_txt;

* 1. **Hacer invalidate metadata en Impala de Base de datos padron\_particionado.**

--HACER EN IMPALA

INVALIDATE METADATA;

* 1. **Calcular el total de EspanolesHombres, EspanolesMujeres, ExtranjerosHombres y ExtranjerosMujeres agrupado por DESC\_DISTRITO y DESC\_BARRIO para los distritos CENTRO, LATINA, CHAMARTIN, TETUAN, VICALVARO y BARAJAS.**
     1. **Llevar a cabo la consulta en Hive en las tablas padron\_parquet y padron\_particionado. ¿Alguna conclusión?**

-- Para la tabla padron\_parquet (28 segundos)

-- Para la tabla padron\_parquet\_particionado (15 segundos)

-- Como podemos observar las particiones por distrinto y barrio hacen que sean más rapidas las consultas.

SELECT

DESC\_DISTRITO,

DESC\_BARRIO,

SUM(EspanolesHombres) TotalEsp\_H,

SUM(EspanolesMujeres) TotalEsp\_M,

SUM(ExtranjerosHombres) TotalExt\_H,

SUM(ExtranjerosMujeres) TotalExt\_M

FROM padron\_particionado\_parquet -- cambiar (padron\_parquet o padron\_particionado\_parquet)

-- En lugar de usar = o LIKE y repetir la vairble cogemos IN y ponemos los valores en una lista

WHERE DESC\_DISTRITO IN ('CENTRO','LATINA','CHAMARTIN','TETUAN','VICALVARO','BARAJAS')

GROUP BY DESC\_DISTRITO, DESC\_BARRIO;

* + 1. **Llevar a cabo la consulta en Impala en las tablas padron\_parquet y padron\_particionado. ¿Alguna conclusión?**

-- Para la tabla padron\_parquet (28 segundos)

-- Para la tabla padron\_parquet\_particionado (15 segundos)

-- Como podemos observar las tablas con particiones son mas rápidas querys en Impala son mas rápidas.

SELECT

DESC\_DISTRITO,

DESC\_BARRIO,

SUM(EspanolesHombres) TotalEsp\_H,

SUM(EspanolesMujeres) TotalEsp\_M,

SUM(ExtranjerosHombres) TotalExt\_H,

SUM(ExtranjerosMujeres) TotalExt\_M

FROM padron\_particionado\_parquet -- cambiar (padron\_parquet o padron\_particionado\_parquet)

-- En lugar de usar = o LIKE y repetir la vairble cogemos IN y ponemos los valores en una lista

WHERE DESC\_DISTRITO IN ('CENTRO','LATINA','CHAMARTIN','TETUAN','VICALVARO','BARAJAS')

GROUP BY DESC\_DISTRITO, DESC\_BARRIO;

* 1. **Hacer consultas de agregación (Max, Min, Avg, Count) tal cual el ejemplo anterior con las 3 tablas (padron\_txt, padron\_parquet y padron\_particionado) y comparar rendimientos tanto en Hive como en impala y sacar conclusiones.**

-- Agrupamos en distritos y barrios con más extranjeros (Hombre, mujer y total) en orden ascendente

SELECT DESC\_DISTRINTO,

DESC\_BARRIO,

Max(ExtranjerosHombres) Ext\_H

Max(Extranjeros Mujeres) Ext\_M

Ext\_H + Ext\_M AS Tot\_Ext

FROM padron\_particionado\_parquet – podemos cambiar por padron\_txt, padron\_parquet y padron\_particionado\_parquet

GROUP BY DESC\_DISTRINTO, DESC\_BARRIO

SORTED BY

-- Distritos y Barrios agrupados por Población total de empadronados ordenados de menor a mayor

SELECT DESC\_DISTRITO, DESC\_BARRIO,

(SUM(ExtranjerosHombres) + SUM(ExtranjerosMujeres) + SUM(EspanolesHombres) + SUM(EspanolesMujeres)) AS 'Poblacion'

FROM padron\_parquet2

GROUP BY DESC\_DISTRITO, DESC\_BARRIO

ORDER BY Poblacion ASC

-- Distritos/Barrios del total de extranjeros ordenados de mayor a menor

SELECT DESC\_DISTRITO, DESC\_BARRIO,

(SUM(ExtranjerosHombres) + SUM(ExtranjerosMujeres)) AS Tot\_Ext,

SUM(ExtranjerosHombres) AS Extranjeros\_Hombres,

SUM(ExtranjerosMujeres) AS Extranjeros\_Mujeres

FROM padron\_parquet2

GROUP BY DESC\_DISTRITO, DESC\_BARRIO

ORDER BY Tot\_Ext DESC

--Población Media agrupada por distrito y barrio

SELECT DESC\_DISTRITO, DESC\_BARRIO,

AVG(ExtranjerosHombres) Avg\_Ext\_H,

AVG(ExtranjerosMujeres) Avg\_Ext\_M,

AVG(EspanolesHombres) Avg\_Esp\_H,

AVG(EspanolesMujeres) Avg\_Esp\_M

FROM padron\_parquet2

GROUP BY DESC\_DISTRITO, DESC\_BARRI

En cuanto a las clonclusiones, comparando con esta query observamos:

* Mucha diferencia en tiempos de ejecución de querys entre Hive /Impala (siendo Impala mucho más rápido, pero soportando menos funcionalidades (función SORT BY por ejemplo).
* En cuanto a formatos columnares de tabla (txt, parquet) se aprecian mas rapidez en tiempos de ejecucion de querys en parquet.
* En cuanto a particionado/no particionado, se aprecia más rapidez en tiempos de ejecución de querys en las tablas particionadas.
* Finalmente creo que lo más óptimo para trabajar con Hive/Impala es utilizar las consultas más rápidas y que se puedan volver a lanzar sin problema en Impala, usando el formato de columna parquet y el particionamiento para mejorar el rendimiento.

Información:

* <https://cwiki.apache.org/confluence/collector/pages.action?key=Hive>
* <https://cwiki.apache.org/confluence/display/Hive/GettingStarted#GettingStarted-CreatingHiveTables>
* <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL>
* <https://cwiki.apache.org/confluence/display/Hive/DynamicPartitions>