Práctica Hive

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

En consola:

hive

create database datos\_padron

use datos\_padron

### Vamos a importar primero los datos para que aparezcan los campos con comillas, tenemos que definir los campos como string ya que tienen comillas y si intentamos pasarlos a INT peta.

CREATE TABLE padron(

COD\_DISTRITO STRING,

DESC\_DISTRITO STRING,

COD\_DIST\_BARRIO STRING,

DESC\_BARRIO STRING,

COD\_BARRIO STRING,

COD\_DIST\_SECCION STRING,

COD\_SECCION STRING,

COD\_EDAD\_INT STRING,

EspanolesHombres STRING,

EspanolesMujeres STRING,

ExtranjerosHombres STRING,

ExtranjerosMujeres STRING)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ';'

tblproperties("skip.header.line.count"="1")

Cuando insertamos los datos en esta tabla (load data local inpath '/home/cloudera/Desktop/Padron/Padron.csv' into table padron\_txt;) esto es lo que va a resultar.



**### Si la queremos crear a lo pro delimitando el formato correctamente podemos hacerlo con OpenCSV:**

CREATE TABLE padron(

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'

WITH SERDEPROPERTIES (

"separatorChar" = ";",

"quoteChar" = "\""

)

tblproperties("skip.header.line.count"="1")

**Cuando insertemos datos en esta tabla (**load data local inpath '/home/cloudera/Desktop/Padron/Padron.csv' into table padron\_txt;) nos va a quedar lo siguiente:



* 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

load data local inpath '/home/cloudera/Desktop/Padron/Padron.csv' into table padron\_txt;

Comprobamos que no hay valores nulos en las columnas de código ya que son celdas en blanco y no nulos como tal y si intentamos importarlas como INT o hacer un casteo posterior va a fallar.

select cod\_distrito from padron where (length(cod\_distrito)=0 or cod\_distrito == null)

select cod\_dist\_barrio from padron where (length(cod\_distrito)=0 or cod\_distrito == null)

select cod\_barrio from padron where (length(cod\_distrito)=0 or cod\_distrito == null)

select cod\_dist\_seccion from padron where (length(cod\_distrito)=0 or cod\_distrito == null)

select cod\_seccion from padron where (length(cod\_distrito)=0 or cod\_distrito == null)

select cod\_edad\_int from padron where (length(cod\_distrito)=0 or cod\_distrito == null)

Creamos la tabla con todo reformateado para que sea correcto usando expresiones CTAS (Create Table As Select).

create table padron\_bueno as

select cast(cod\_distrito as int) cod\_distrito,

rtrim(desc\_distrito) desc\_distrito,

cast(cod\_dist\_barrio as int) cod\_dist\_barrio,

rtrim(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(CASE WHEN (length(espanoleshombres) = 0) THEN 0 ELSE espanoleshombres END as int) espanoleshombres,

cast(CASE WHEN (length(espanolesmujeres) = 0) THEN 0 ELSE espanolesmujeres END as int) espanolesmujeres,

cast(CASE WHEN (length(extranjeroshombres) = 0) THEN 0 ELSE extranjeroshombres END as int) extranjeroshombres,

cast(CASE WHEN (length(extranjerosmujeres) = 0) THEN 0 ELSE extranjerosmujeres END as int) extranjerosmujeres

from padron

Una manera alternativa para conseguir todo esto de golpe sería utilizar expresiones regulares con un SERDE para declarar el formato de las filas correctamente.

CREATE TABLE padron\_def (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.RegexSerDe'

WITH SERDEPROPERTIES ('input.regex'='"(\\d\*)";"(.\*?)\\s\*";"(\\d\*)";"(.\*?)\\s\*";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)"')

tblproperties("skip.header.line.count" = "1",'serialization.encoding'='utf-8');

* (OPCIONAL) Crear tabla padron\_txt2 que haga **trim** sobre los datos de algunas columnas con la finalidad de eliminar los espacios en blanco innecesarios.

Hecho en el apartado anterior.

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

Local hace que carguemos los datos desde disco de nuestra compu, indicando el directorio, si no lo ponemos cargamos datos desde el sistema de almacenamiento HDFS.

* ¿Qué es CTAS?

Create Table As Select, es lo que hemos hecho ya, pero podemos investigar el crearla en formato parquet.

**Las tablas creadas en Hive se guardan por defecto en su WareHouse, con ruta /user/hive/warehouse.**

* 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.

create table padron\_parquet stored as Parquet as

select \*

from padron

padron -----> 21.5 Mb

padron\_parquet -----> 931.8 Kb

* (OPCIONAL) 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.

create table padron\_bueno\_parquet stored as Parquet as

select \*

from padron\_bueno

padron\_bueno ------> 11.8 Mb, hemos reducido bastante con respecto a Padron quitando espacios.

padron\_bueno\_parquet ------> 916.8 Kb

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

Parquet is a column-oriented binary file format. The parquet is highly efficient for the types of large-scale queries. Parquet is especially good for queries scanning particular columns within a particular table. The Parquet table uses compression Snappy, gzip; currently Snappy by default.

Son archivos en formato binario en contraposición a otros formatos en formato texto.

Contienen una cabecera de metadatos para ayudar a la máquina a la hora de interpretar el contenido.

* 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)

Hecho en apartado anterior.

* (OPCIONAL) 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)

Hecho en apartado anterior.

Para consultar las movidas de cualquier tabla:

show table extended like padron;

* Impala
* ¿Qué es impala?

Es un motor para colsultas SQL masivas que procesa los datos en paralelo sobre el mismo cluster de HDFS en lugar de correr procesos map-reduce. Presencia latencias de milisegundos

* ¿En qué se diferencia a Hive?

En el procesamiento, que no lo hace por Map-Reduce y en la latencia principalmente ya que es un servicio de baja latencia pero que no presenta la misma tolerancia a fallos que Hive. Hay muchas instrucciones y formatos de datos compatibles con Hive que Impala no soporta, está en fase de desarrollo.

* Comando INVALIDATE METADATA, ¿en qué consiste?

Cuando hacemos cambios a los metadatos desde fuera de impala no se actualizan automaticamente por lo tanto hay que actualizar los metadatos cacheados de la tabla, y lo podemos hacer con INVALIDATE METADATA en el caso de que estemos añadiendo una tabla nueva para que cargue la metadata de esa tabla. Si estamos modificando un esquema o añadiendo datos a la tabla haríamos REFRESH <table> y si estamos modificando masivamente una tabla previamente existente podemos añadir el nombre de la tabla a INVALIDATE METADATA <table> para que la actualice la proxima vez que llamemos a la tabla.

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

**Invalidate MEtadata; y resetea la meta de todo, ahora podemos lanzar queries con las nuevas db, detecta incluso db con todas las tablas q haya dentro.**

* Calcular el total de EspanolesHombres, EspanolesMujeres, ExtranjerosHombresy ExtranjerosMujeres agrupado por DESC\_DISTRITO y DESC\_BARRIO.

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

from padron\_bueno

group by desc\_distrito, desc\_barrio

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

21 segundos, hive no parquet.

23 segundos, hive parquet.

Tarda más en parquet, puede que se deba a la descompresión que tiene que hacer de los datos.

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

1 segundos sin parquet.

2 segundos en parquet.

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

Mucha, Hive necesita remitirse a métodos de Map Reduce que ralentizan mucho el proceso y al elaborar queries con datos no masivos sencillamente no merece la pena y es mejor usar Hive.

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

CREATE TABLE padron\_particionado(

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 Parquet

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

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;

set mapreduce.map.memory.mb = 2048;

set mapreduce.reduce.memory.mb = 2048;

set mapreduce.map.java.opts=-Xmx1800m

INSERT INTO TABLE padron\_particionado 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\_bueno\_parquet

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

**invalidate metadata padron\_particionado**

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

select desc\_distrito, desc\_barrio, sum(espanoleshombres), sum(espanolesmujeres), sum(extranjeroshombres), sum(extranjerosmujeres)

from padron\_particionado

where desc\_distrito not in ("CENTRO", "LATINA", "CHAMARTIN", "TETUAN", "VICALVARO", "BARAJAS")

group by desc\_distrito, desc\_barrio

31 segundos

select desc\_distrito, desc\_barrio, sum(espanoleshombres), sum(espanolesmujeres), sum(extranjeroshombres), sum(extranjerosmujeres)

from padron\_def\_parquet

where desc\_distrito not in ("CENTRO", "LATINA", "CHAMARTIN", "TETUAN", "VICALVARO", "BARAJAS")

group by desc\_distrito, desc\_barrio

30 segundos

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

select desc\_distrito, desc\_barrio, sum(espanoleshombres), sum(espanolesmujeres), sum(extranjeroshombres), sum(extranjerosmujeres)

from padron\_def\_parquet

where desc\_distrito not in ("CENTRO", "LATINA", "CHAMARTIN", "TETUAN", "VICALVARO", "BARAJAS")

group by desc\_distrito, desc\_barrio

5 segundos

select desc\_distrito, desc\_barrio, sum(espanoleshombres), sum(espanolesmujeres), sum(extranjeroshombres), sum(extranjerosmujeres)

from padron\_particionado

where desc\_distrito not in ("CENTRO", "LATINA", "CHAMARTIN", "TETUAN", "VICALVARO", "BARAJAS")

group by desc\_distrito, desc\_barrio

4 segundos

* 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.

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/GettingStarted)
* <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL>
* <https://cwiki.apache.org/confluence/display/Hive/DynamicPartitions>

########## El siguiente contenido no forma parte como tal del documento que finaliza aquí, se considera código de prueba.

Lo del Latin:

CREATE TABLE padron\_latin(

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'

WITH SERDEPROPERTIES (

"separatorChar" = ";",

"quoteChar" = "\"",

"serialization.encoding"='ISO-8859-1'

)

tblproperties("skip.header.line.count"="1",

'store.charset'='ISO-8859-1',

'retrieve.charset'='ISO-8859-1')

"([\\d\*)";"(\\w\*\\s{1}\\w\*)\\s\*";"\\d\*";"(\\w\*\\s{1}\\w\*)\\s\*";"\\d\*";"\\d\*";"\\d\*";"\\d\*";"\\d\*";"\\d\*";"\\d\*";"\\d\*](file://\\d*)%252522;%252522(\\w*\\s%2525257B1%2525257D\\w*)\\s*%25252522;%25252522\\d*%25252522;%25252522(\\w*\\s%2525257B1%2525257D\\w*)\\s*%25252522;%25252522\\d*%25252522;%25252522\\d*%25252522;%25252522\\d*%25252522;%25252522\\d*%25252522;%25252522\\d*%25252522;%25252522\\d*%25252522;%25252522\\d*%25252522;%25252522\\d*)"

CREATE EXTERNAL TABLE padron\_def (

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.RegexSerDe'

WITH SERDEPROPERTIES ('input.regex'='"(\\d\*)";"(.\*?)\\s\*";"(\\d\*)";"(.\*?)\\s\*";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)"',

'serialization.encoding'='ISO-8859-1')

tblproperties("skip.header.line.count"="1")

CREATE EXTERNAL TABLE padron\_def (

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.RegexSerDe'

WITH SERDEPROPERTIES ('input.regex'='"(\\d\*)";"(.\*?)\\s\*";"(\\d\*)";"(.\*?)\\s\*";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)"',

'serialization.encoding'='unicode')

tblproperties("skip.header.line.count"="1")

drop table padron\_def

load data local inpath '/home/cloudera/Desktop/Padron/padron\_UTF.csv' into table padron\_def;

##############################################

CÓDIGO ENTERITO PARA HACER TODO DE GOLPE:

CREATE TABLE padron\_def (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.RegexSerDe'

WITH SERDEPROPERTIES ('input.regex'='"(\\d\*)";"(.\*?)\\s\*";"(\\d\*)";"(.\*?)\\s\*";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)";"(\\d\*)"')

tblproperties("skip.header.line.count" = "1",'serialization.encoding'='utf-8');

STORED AS TEXTFILE;

drop table padron\_def

load data local inpath '/home/cloudera/Desktop/Padron/PadronUTF.csv' into table padron\_def;

select \* from padron\_def;

select distinct(desc\_barrio) from padron\_def;

########################################

create table padron\_def\_parquet stored as Parquet as

select

cod\_distrito,

desc\_distrito,

cod\_dist\_barrio,

desc\_barrio,

cod\_barrio,

cod\_dist\_seccion,

cod\_seccion,

cod\_edad\_int,

coalesce(espanoleshombres, 0) espanoleshombres,

coalesce(espanolesmujeres, 0) espanolesmujeres,

coalesce(extranjeroshombres, 0) extranjeroshombres,

coalesce(extranjerosmujeres, 0) extranjerosmujeres

from padron\_def

drop table padron\_def\_parquet

select distinct(desc\_barrio) from padron\_def\_parquet

select \* from padron\_def\_parquet limit 5

######################################

CREATE TABLE padron\_particionado(

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 Parquet

drop table padron\_particionado

SET hive.exec.dynamic.partition = true;

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

set mapreduce.map.memory.mb = 2048;

set mapreduce.reduce.memory.mb = 2048;

set mapreduce.map.java.opts=-Xmx1800m

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

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

INSERT INTO TABLE padron\_particionado 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\_def\_parquet