

Paula Iglesias Reina

Ejercicio 2 - Investigamos el formato columnar parquet.

2.1) ¿Qué es CTAS?

CTAS es una operación paralela que crea una nueva tabla basada en la salida de una instrucción SELECT. CTAS es la forma más sencilla y rápida de crear e insertar datos en una tabla con un solo comando.

Ej: `CREATE TABLE NuevaTabla AS (SELECT * FROM OtraTabla);`

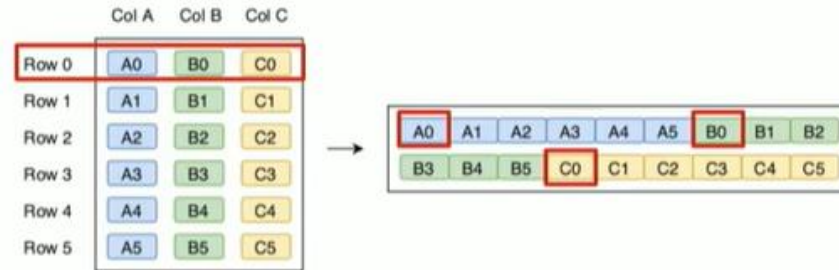
2.4) Opcionalmente también se pueden crear las tablas directamente desde 0 (en lugar de mediante CTAS) en formato parquet igual que lo hicimos para el formato txt incluyendo la sentencia `STORED AS PARQUET`. Es importante para comparaciones posteriores que la tabla `padron_parquet` conserve los espacios innecesarios y la tabla `padron_parquet_2` no los tenga. Dejo a tu elección cómo hacerlo.

Mi elección es con sentencia CTAS

2.5) Investigar en qué consiste el formato columnar parquet y las ventajas de trabajar en este tipo de formatos

Parquet es el formato de almacenamiento en columnas principal en el ecosistema Hadoop. Permite estructuras anidadas. Es muy adecuado para escenarios OLAP (unas pocas operaciones largas que incluyen un subconjunto de todas las columnas), almacenamiento de columnas y escaneo

[\(3\) The Parquet Format and Performance Optimization Opportunities Boudewijn Braams \(Databricks\) - YouTube](#)



- Vertical partitioning
- OLTP ❌, OLAP ✅
 - Free projection pushdown
 - Compression opportunities

Hay un dicho: si HDFS es el estándar de facto para los sistemas de archivos en la era del big data, Parquet es el estándar de facto para los formatos de almacenamiento en la era del big data.

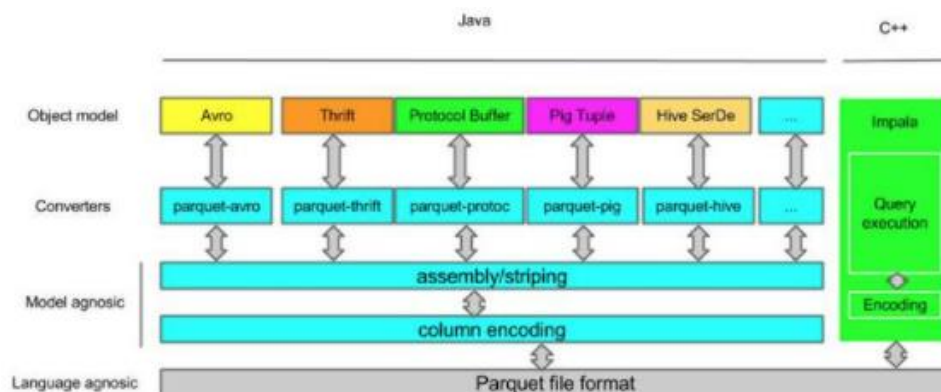
Ventajas

1. Relación de compresión más alta

El almacenamiento de columnas facilita el uso de una codificación y compresión eficientes para cada columna, lo que reduce el espacio en disco.

2. Operaciones IO más pequeñas

Utilice la inserción de mapas y la inserción de predicados para leer solo las columnas requeridas y omitir las columnas que no cumplan con las condiciones, lo que puede reducir el escaneo de datos innecesario, traer mejoras de rendimiento y volverse más obvias cuando hay más campos de tabla.



Ejercicio 3 - Juguemos con Impala

3.1) ¿Qué es Impala?

Es un motor SQL de elevado performance pensada para operar sobre grandes volúmenes de datos

- El motor es MPP: procesamiento masivo en paralelo
- Con latencias de Milisegundos

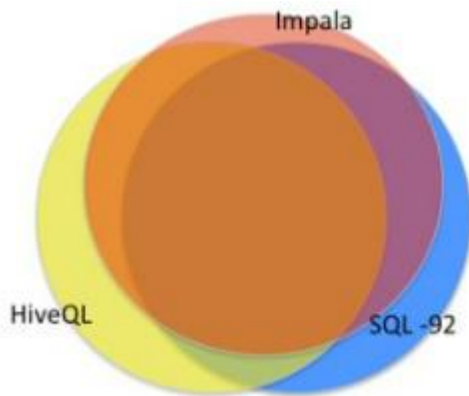
Impala corre sobre clusters Hadoop

- Puede ejecutar Queries sobre HDFS o Hbase

Lee y escribe datos sobre ficheros con tipos de datos típicos de Hadoop

3.2) ¿En qué se diferencia de Hive?

Es en torno a unas 5 veces más rápido que Hive o Pig, aunque a menudo puede ser hasta 20 veces más rápido. Está especialmente optimizado para ejecutar Queries.



Hive fue una de las primeras herramientas que formaron parte del entorno de producción en soluciones basadas en Hadoop, por lo que, a día de hoy, ofrece mucha más funcionalidad.

Hive e Impala son herramientas diferentes, pero ofrecen bastantes similitudes

- Ambas usan variantes de SQL
- Ambas comparten el mismo metastore y data warehouse en el cluster
- En proyectos grandes se suelen usar juntas

Las queries en Impala y Hive operan sobre tablas, igual que en RDBMS (Sistema de gestión de bases de datos relacionales)

- Como sabemos, una tabla Hive no es más que una ruta en el cluster que contiene archivos

La estructura y localización de las tablas se definen cuando se crean

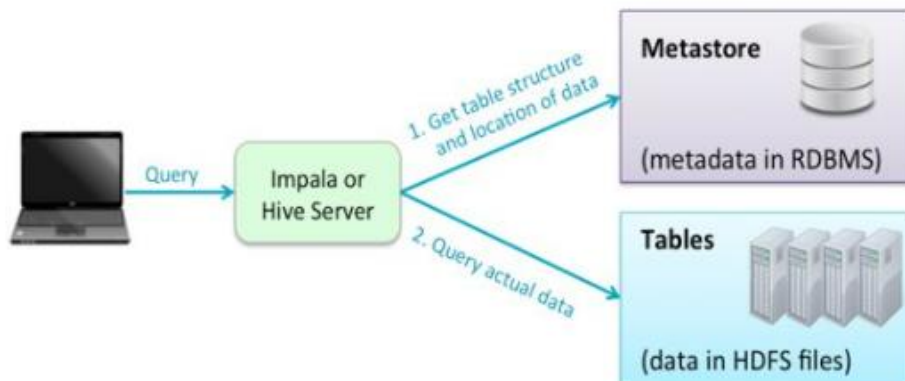
- Dicha información se almacena en el Metastore
 - Almacenado en una RDBMS, por ejemplo MySQL
 - Almacena metadatos de tablas de Hive/Impala

Impala, como Hive, operan sobre los mismos datos

- Tablas en HDFS, y metadatos en el Metastore

Impala: comparativa con Hive

El flujo sería el siguiente:



Funcionalidades que Impala no soporta pero Hive sí

- Ficheros con tipos de datos a medida
- El tipo de datos DATE
- Funciones XML y JSON
- Algunas funciones de agregación como: “covar_pop, covar_samp, corr, percentile, percentile_approx, histogram_numeric, collect_set”
- Sampling (que es el ejecutar queries sobre un subset de una tabla en lugar de sobre toda la tabla)
- Vistas laterales (sobre una columna de una tabla, fila a fila, se le aplica una función (que crea un resultado en forma de vista en ejecución) y sobre ese resultado se aplica otra función, que es lo que se muestra.
- Múltiples cláusulas DISTINCT por query
- UDFs (soportadas a partir de impala 1.2)

Hay más diferencias

Aunque una de las grandes ventajas de Hive e Impala frente a RDBMS es que te permiten definir el esquema de los datos después de que estén en el cluster, no antes

3.3) Comando INVALIDATE METADATA, ¿en qué consiste?

Marca los metadatos de una o todas las tablas como invalidados. La próxima vez que el servicio Impala realice una consulta contra una tabla cuyos metadatos estén invalidados, Impala recargará los metadatos asociados antes de que la consulta continúe. Como esta es una operación muy costosa comparada con la actualización incremental de metadatos realizada por la sentencia REFRESH, cuando sea posible, prefiera REFRESH en lugar de INVALIDAR METADATOS.

INVALIDATE METADATA es necesario cuando se realizan los siguientes cambios fuera de Impala, en Hive y en otro cliente Hive, como SparkSQL:

- Los metadatos de las tablas existentes cambian.
- Se añaden nuevas tablas, e Impala utilizará las tablas.
- Se cambian los privilegios de Sentry de nivel SERVER o DATABASE.

- Los metadatos de los bloques cambian, pero los archivos siguen siendo los mismos (reequilibrio de HDFS).
- Los jars UDF cambian.
- Algunas tablas ya no se consultan y se desea eliminar sus metadatos de las cachés del catálogo y del coordinador para reducir los requisitos de memoria.

Ejercicio 5- Trabajando con tablas en HDFS

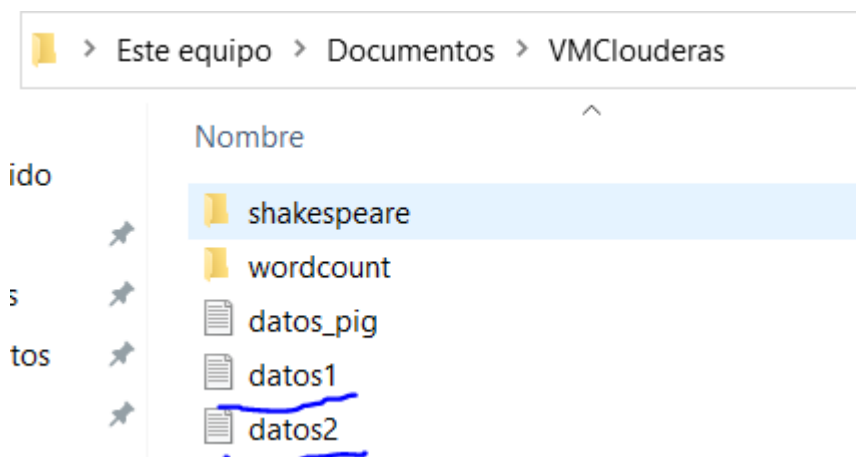
5.1) Crear un documento de texto en el almacenamiento local que contenga una secuencia de números distribuidos en filas y separados por columnas, llámalo datos1 y que sea por ejemplo:

1,2,3

4,5,6

7,8,9

Lo he creado desde Windows en la carpeta compartida con la MV:



Los copiamos a una ruta con permisos para el usuario cloudera (dir /home/cloudera/datos12):

```
cp /mnt/Shared/datos2.txt /home/cloudera/datos12/datos2.txt
```

```
[cloudera@quickstart ~]$ su
Password:
[root@quickstart cloudera]# cp /mnt/Shared/datos1.txt /home/cloudera/datos12/datos1.txt
[root@quickstart cloudera]# cd /home/cloudera/datos12
[root@quickstart datos12]# ls
datos1.txt
-
```

Desde la MV

5.3) Crear un directorio en HDFS con un nombre a placer, por ejemplo, /test. Si estás en una máquina Cloudera tienes que asegurarte de que el servicio HDFS está activo ya que puede no iniciarse al encender la máquina (puedes hacerlo desde el Cloudera Manager). A su vez, en las máquinas Cloudera es posible (dependiendo de si usamos Hive desde consola o desde Hue) que no tengamos permisos para crear directorios en HDFS salvo en el directorio /user/cloudera.

```
hadoop fs -mkdir /user/cloudera/test
```

5.4) Mueve tu fichero datos1 al directorio que has creado en HDFS con un comando desde consola.

```
hadoop fs -put /home/cloudera/datos12/datos1.txt /user/cloudera/test
```

Comprobamos que se ha copiado bien:

```
hadoop fs -ls /user/cloudera/test
```

```
[root@quickstart datos12]# hadoop fs -ls /user/cloudera/test
Found 1 items
-rw-r--r-- 1 root cloudera 21 2021-07-13 02:45 /user/cloudera/test/datos1.txt
```

5.5) Desde Hive, crea una nueva database por ejemplo con el nombre numeros. Crea una tabla que no sea externa y sin argumento location con tres columnas numéricas, campos separados por coma y delimitada por filas. La llamaremos por ejemplo numeros_tbl.

```
CREATE TABLE numeros_tbl
```

```
(x int,
```

```
y int,
```

```
z int)
```

```
ROW FORMAT DELIMITED
```

```
FIELDS TERMINATED BY ','
```

```
LINES TERMINATED BY '\n'
```

The screenshot shows the Hue web interface. At the top right, there's a tab labeled '0s numeros'. The main area displays a SQL query that has been executed successfully. The query is:

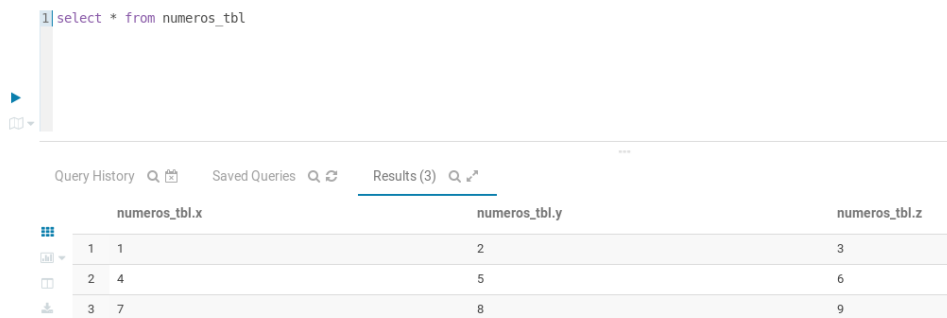
```
1 CREATE TABLE numeros_tbl
2 (x int,
3 y int,
4 z int)
5 ROW FORMAT DELIMITED
6 FIELDS TERMINATED BY ','
7 LINES TERMINATED BY '\n'
```

 Below the query, a green message says 'Success.'. At the bottom, there's a 'Query History' section showing the same query was executed 'a few seconds ago'.

5.6)Carga los datos de nuestro fichero de texto datos1 almacenado en HDFS en la tabla de Hive. Consulta la localización donde estaban anteriormente los datos almacenados. ¿Siguen estando ahí? ¿Dónde están?. Borra la tabla, ¿qué ocurre con los datos almacenados en HDFS?

```
LOAD DATA INPATH '/user/cloudera/test/datos1.txt' INTO TABLE numeros_tbl
```

Vemos que se han guardado correctamente:



```
select * from numeros_tbl
```

	numeros_tbl.x	numeros_tbl.y	numeros_tbl.z
1	1	2	3
2	4	5	6
3	7	8	9

Ahora comprobamos si siguen en la misma localización:

```
[root@quickstart datos12]# hadoop fs -ls /user/cloudera/test
```

No están

Note that after loading the data, the source file will be deleted from the source location, and the file loaded to the [Hive data warehouse location](#) or to the LOCATION specified while creating a table.

Se encuentra almacenado dentro del datawarehouse de hive dentro de la base de datos numeros

```
[root@quickstart datos12]# hadoop fs -ls /user/hive/warehouse/numeros.db
Found 1 items
drwxrwxrwx - cloudera supergroup          0 2021-07-13 03:03 /user/hive/warehouse/numeros.db/numeros_tbl
```

Si borramos la tabla con:

```
drop table numeros_tbl
```

Ya no aparece nada en la misma ubicación de antes:

```
[root@quickstart datos12]# hadoop fs -ls /user/hive/warehouse/numeros.db
Found 1 items
drwxrwxrwx - cloudera supergroup          0 2021-07-13 03:03 /user/hive/warehouse/numeros.db/numeros_tbl
[root@quickstart datos12]#
[root@quickstart datos12]# hadoop fs -ls /user/hive/warehouse/numeros.db
[root@quickstart datos12]#
```

5.7)Vuelve a mover el fichero de texto datos1 desde el almacenamiento local al directorio anterior en HDFS.

```
hadoop fs -put /home/cloudera/datos12/datos1.txt /user/cloudera/test
```

5.8) Desde Hive, crea una tabla externa sin el argumento location. Y carga datos1 (desde HDFS) en ella. ¿A dónde han ido los datos en HDFS? Borra la tabla ¿Qué ocurre con los datos en hdfs?

2. Hive External Table

Hive does not manage the data of the External table.

We create an external table for external use as when we want to use the data outside the Hive.

External tables are stored outside the warehouse directory. They can access data stored in sources such as remote HDFS locations or Azure Storage Volumes.

Whenever we drop the external table, then only the metadata associated with the table will get deleted, the table data remains untouched by Hive.

```
CREATE EXTERNAL TABLE numeros_tbl
(x int,
y int,
z int)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
```

When you create external table with out location , the data will be stored in the hive default location

Así que vuelve a la misma ubicación de antes

```
[root@quickstart datos12]# hadoop fs -ls /user/hive/warehouse/numeros.db
Found 1 items
drwxrwxrwx - cloudera supergroup          0 2021-07-13 03:15 /user/hive/warehouse/numeros.db/numeros_tbl
[root@quickstart datos12]#
```

Tras borrarla y mirar a ver si se ha borrado de la ubicación la tabla sigue apareciendo:

```
[root@quickstart datos12]# hadoop fs -ls /user/hive/warehouse/numeros.db
Found 1 items
drwxrwxrwx - cloudera supergroup          0 2021-07-13 03:15 /user/hive/warehouse/numeros.db/numeros_tbl
[root@quickstart datos12]#
```

Dropping an external table removes all **table**-related metadata. It doesn't **delete** the **external** data.

Para borrarla habria que hacerla tabla interna con un alter table (ALTER TABLE <table-name> SET TBLPROPERTIES('EXTERNAL'='False'); //) y luego ya se podria borrar

5.9) Borra el fichero datos1 del directorio en el que estén. Vuelve a insertarlos en el directorio que creamos inicialmente (/test). Vuelve a crear la tabla numeros desde hive pero ahora de manera externa y con un argumento location que haga referencia al directorio donde los hayas situado en HDFS (/test). No cargues los datos de ninguna manera explícita. Haz una consulta sobre la tabla que acabamos de crear que muestre todos los registros. ¿Tiene algún contenido?

Lo borramos:

```
[cloudera@quickstart ~]$ hadoop fs -rm /user/cloudera/test/datos1.txt
Deleted /user/cloudera/test/datos1.txt
```

Lo volvemos a copiar:

```
hadoop fs -put /home/cloudera/datos12/datos1.txt /user/cloudera/test
```

Creamos la tabla:

```
CREATE EXTERNAL TABLE numeros_tbl
```

```
(x int,
```

```
y int,
```

```
z int)
```

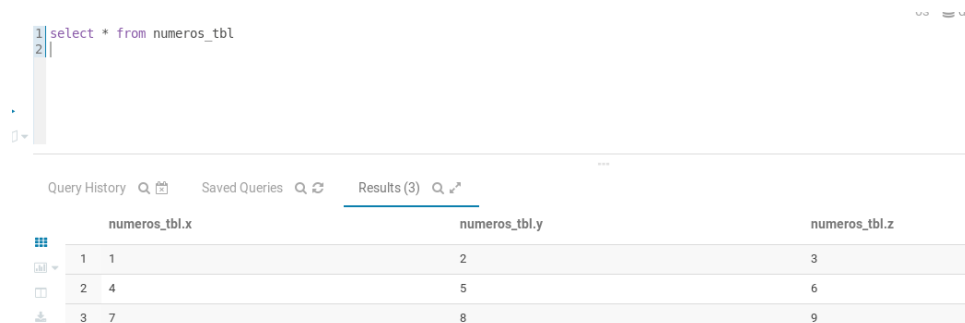
```
ROW FORMAT DELIMITED
```

```
FIELDS TERMINATED BY ','
```

```
LINES TERMINATED BY '\n'
```

```
LOCATION '/user/cloudera/test'
```

Contenido de la tabla:



The screenshot shows a Hive query interface. At the top, a query is entered: `select * from numeros_tbl`. Below the query, the results are displayed in a table with three columns: `numeros_tbl.x`, `numeros_tbl.y`, and `numeros_tbl.z`. The results are as follows:

numeros_tbl.x	numeros_tbl.y	numeros_tbl.z
1	2	3
2	5	6
3	8	9

Vemos si el archivo sigue en la misma ubicación:

```
hadoop fs -ls /user/cloudera/test
```

Si:

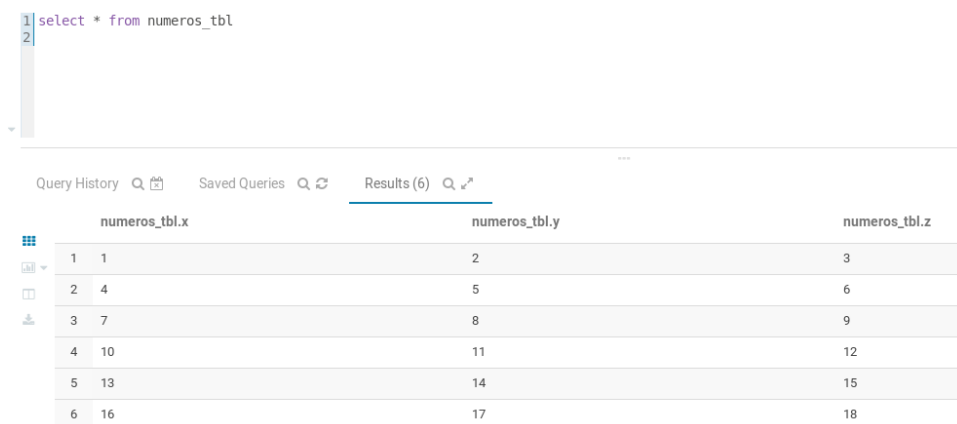
```
[root@quickstart cloudera]# hadoop fs -ls /user/cloudera/test
Found 1 items
-rw-r--r--  1 root cloudera      21 2021-07-14 00:17 /user/cloudera/test/datos1.txt
[root@quickstart cloudera]#
```

5.10) Inserta el fichero de datos creado al principio, "datos2" en el mismo directorio de HDFS que "datos1". Vuelve a hacer la consulta anterior sobre la misma tabla. ¿Qué salida muestra?

```
cp /mnt/Shared/datos2.txt /home/cloudera/datos12/datos2.txt
```

```
hadoop fs -put /home/cloudera/datos12/datos2.txt /user/cloudera/test
```

```
1 select * from numeros_tbl
2
```



	numeros_tbl.x	numeros_tbl.y	numeros_tbl.z
1	1	2	3
2	4	5	6
3	7	8	9
4	10	11	12
5	13	14	15
6	16	17	18

Ha juntado los dos ficheros de texto

5.11) Extrae conclusiones de todos estos anteriores apartados.

CREATE TABLE

- **EXTERNAL** - Para una tabla no gestionada (unmanaged or external), tú defines la tabla y la estructura y se la comunicas a Spark sólo gestiona los metadatos, mientras que tú mismo gestionas los datos en una fuente de datos externa como Cassandra. Si desea ejecutar SQL en esta tabla, puede hacerlo.
- **LOCATION** - Indica donde se encuentran los datos de la tabla
- **ROW FORMAT DELIMITED**
 - **FIELDS TERMINATED BY ','**
 - **LINES TERMINATED BY '\n'**

LOAD DATA INPATH

- Borra el fichero desde el que los datos son exportados