
BIG DATA

HIVE



EDUARD LARA

1. INTRODUCCION

HIVE <http://hive.apache.org/index.html>

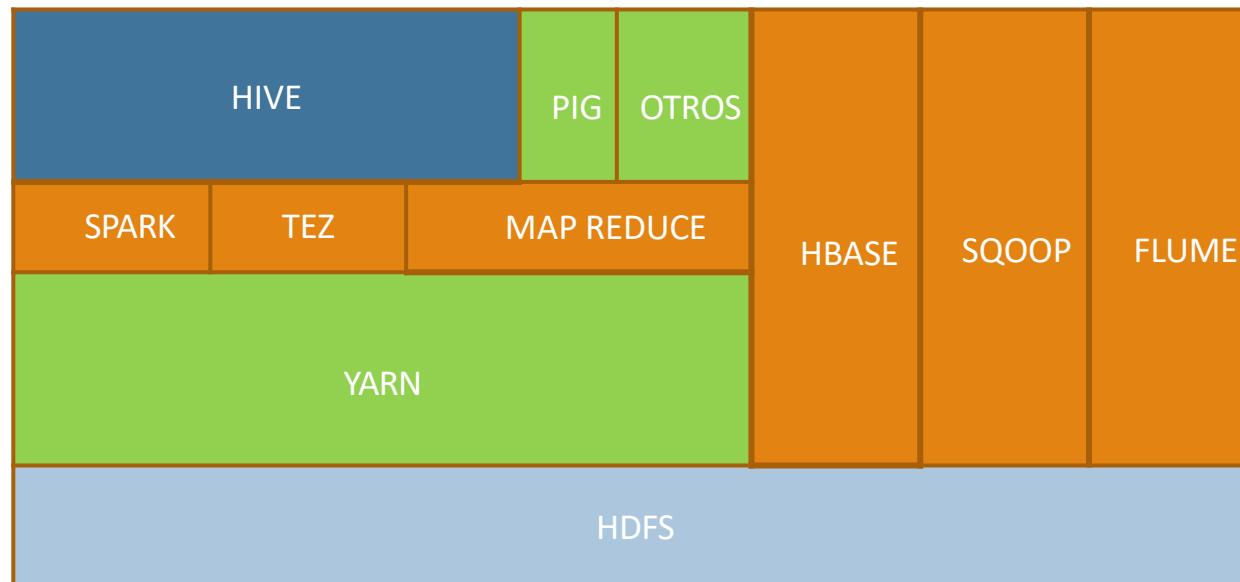
- ❖ Es un producto que forma parte del ecosistema de Hadoop Apache y que normalmente se encuentra en casi todas las instalaciones de Hadoop Big Data.
- ❖ Hasta ahora hemos utilizado varios ejemplos de programas (Java, la Shell de Linux) que nos obligaban entre comillas a crear programas MapReduce.
- ❖ HIVE es otra manera de encarar esta filosofía de trabajo permite acceder a Hadoop-HDFS como si estuviéramos haciéndolo a una bd relacional
- ❖ HIVE no es una base de datos relacional ni convierte HDFS en una base de datos relacional

1. INTRODUCCION

- ❖ Crea una especie de frontal relacional que permite acceder a HDFS de forma muy sencilla sin necesidad de conocer Java, Map Reduce u otras tecnologías
- ❖ El programador no tiene que programar en Java o MapReduce. Hive usa comandos muy parecidos a SQL para recuperar valores, llamado HiveSQL
- ❖ HiveSQL permite simular SQL para acceder a HDFS
- ❖ Cuando tengo que hacer determinadas operaciones analíticas, esto simplifica enormemente la curva de aprendizaje y la gestión con Hadoop.

2. HIVE DENTRO ECOSISTEMA HADOOP

- ❖ Se han representado los productos Hadoop más importantes
- ❖ HDFS + Yarn con MapReduce es la base de Hadoop
- ❖ Yarn soporta distintos motores para trabajar contra HDFS, donde MapReduce ha sido el más utilizado



2. HIVE DENTRO ECOSISTEMA HADOOP

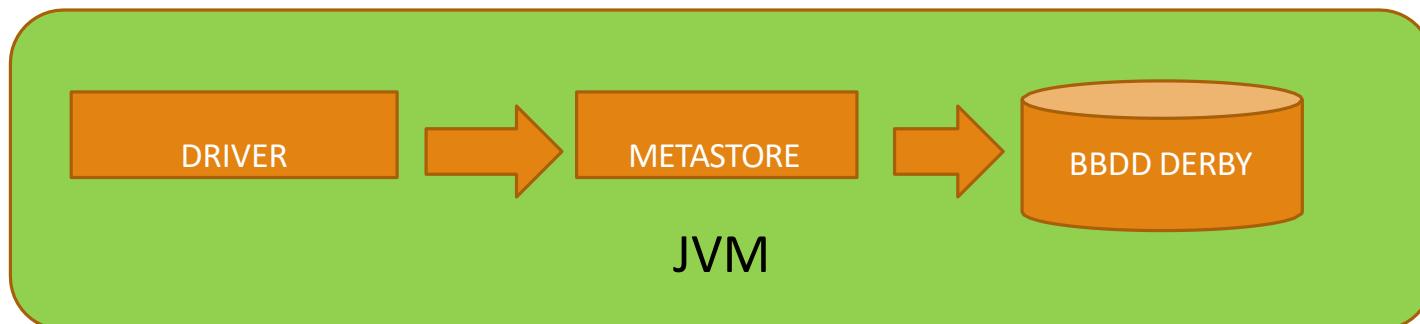
- ❖ Tez o Spark son otros motores Hadoop, que permiten ejecutar programas de Real-Time, in-memory, con memoria de alto procesamiento muy lejos del proceso batch de MapReduce
- ❖ Aunque en principio Hive estaba diseñado para el procesamiento por lotes (MapReduce) ahora se integra con frameworks de tiempo real: Tez y Spark
- ❖ Pig es un lenguaje de scripting.
- ❖ Scoop para poder acceder a base de datos
- ❖ Hay innumerables productos HBASE, FLUME

3. FUNCIONAMIENTO HIVE

- ❖ Hive para trabajar y crear sus objetos utiliza por debajo una base de datos relacional
- ❖ Cuando se crea una tabla en Hive, no se crea en HDFS, no existe realmente ningún contenido.
- ❖ Hive provee un frontal SQL con comandos parecidos y estructura muy parecida a una base de datos relacional
- ❖ Un usuario puede consultar una tabla con una select, pero en realidad por debajo Hive lo convierte a un programa MapReduce para atacar a los ficheros HDFS
- ❖ Toda esa información de metadatos simulados (tablas, vistas) se guarda en una bd Derby que viene empotrada con Hive. Lo suyo seria poner bd mejor: mysql, mariadb

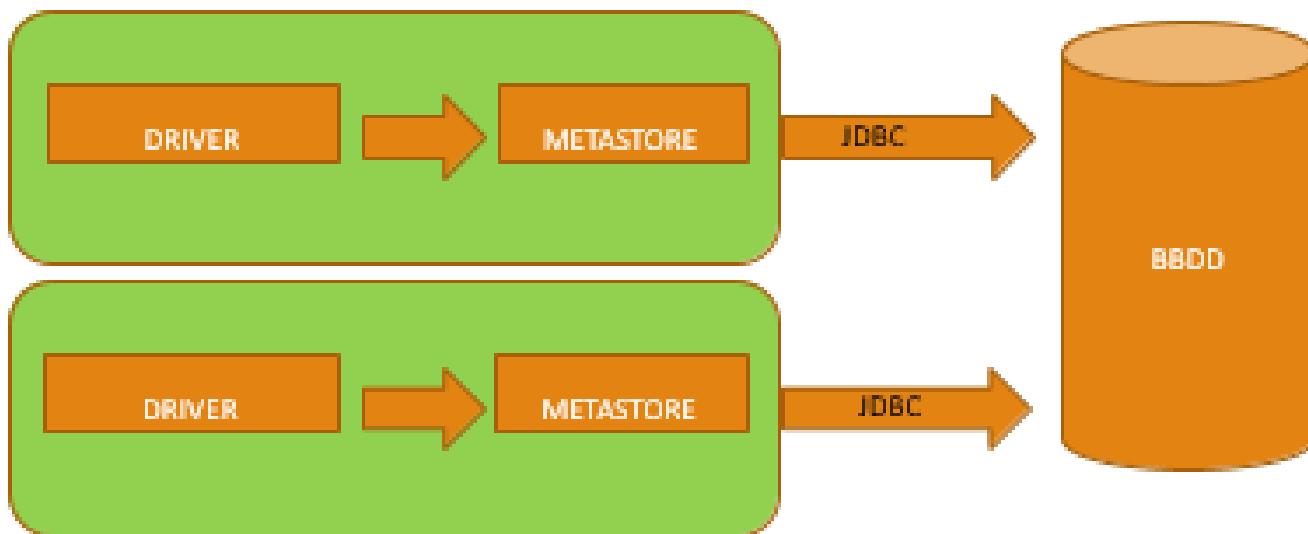
3. MODOS FUNCIONAMIENTO HIVE

- ❖ Hay un servicio que es el **MetaStore**, que es un repositorio central para los metadatos de Hive. Puede trabajar de varias maneras:
- ❖ **MetaStore Modo Embebido:** Está ligado a la base de datos, donde se gestiona los metadatos de Hive
- ❖ Se tiene una sola máquina virtual Java o motor embebido donde todo se está ejecutando la JVM.
- ❖ El problema es que solo la puede utilizar un usuario para hacer pruebas. En la vida real se utiliza muy poco.



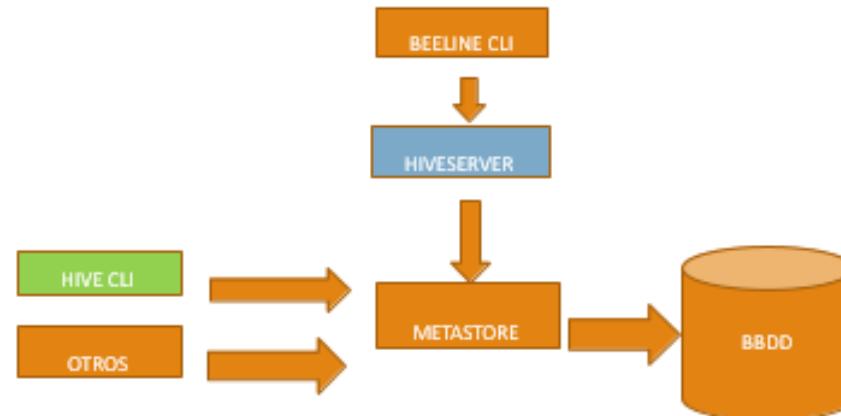
3. MODOS FUNCIONAMIENTO HIVE

- ❖ **Mestatore Modo Local.** Más potente que la anterior.
- ❖ La JVM ejecuta el programa más la parte del servicio de MetaStore y luego vía JDBC atacamos a la base de datos donde tenemos los metadatos
- ❖ Nos permite tener múltiples usuarios lo que pasa que por defecto todo está en la misma máquina.



3. MODOS FUNCIONAMIENTO HIVE

- ❖ **Mestatore Modo Remoto:** Manera más habitual de trabajar.
- ❖ El MetaStore puede ser accedido por el HiveServer o también con cliente Hive CLI en la forma
- ❖ El Hive Server puede ser accedido mediante el cliente Beeline CLI del propio Hive.
- ❖ El HiveServer accede al servicio de MetaStore que a su vez se encarga de la gestión de los metadatos.



4. TIPOS DATOS HIVE

Tipos de datos básicos dentro de Hive

- Son muy similares a los de otros lenguajes (enteros, float, double, string, etc)

TYPE	DESCRIPTION	EXAMPLE
TINYINT	8-bit signed integer	1
SMALLINT	16-bit signed integer	1
INT	32-bit signed integer	1
BIGINT	64-bit signed integer	1
FLOAT	32-bit single precision floating point number	1.0
DOUBLE	64-bit double precision floating point number	1.0
BOOLEAN	true/false value	TRUE
STRING	Character string	'a','a'
TIMESTRAMP	Timestamp with nanosecond precision	'2012-01-02 03:04:05.123456789'

4. TIPOS DATOS HIVE

Tipos complejos

TYPE	DESCRIPTION	EXAMPLE
ARRAY	Colección ordenada de campos. Los campos deben ser del mismo tipo	array(1, 2)
MAP	Colección no ordenada de pares clave-valor. Las claves deben ser primitivas y los valores de cualquier tipo.	map('a', 1, 'b', 2)
STRUCT	Colección de campos con nombre. Los campos pueden ser de distinto tipo.	struct('a', 1, 1.0)

Los pasos a realizar con Hive son: primero descargar el software, después configurar la bd para los metadatos y finalmente unirla a HDFS.

5. INSTALACION HIVE

Paso 1. En la pagina oficial de hive <http://hive.apache.org> encontramos distinta información del producto, una wiki, tutoriales.

En la url de descarga <https://dlcdn.apache.org/hive> vemos una lista de diferentes versiones. Descargaremos el binario tar.gz de la ultima versión, la 3.1.3

Index of /hive

<u>Name</u>	<u>Last modified</u>
Parent Directory	
hive-1.2.2/	2022-06-1
hive-2.3.9/	2022-06-1
hive-3.1.2/	2022-06-1
hive-3.1.3/	2022-06-1
hive-4.0.0-alpha-1/	2022-06-1
hive-4.0.0-alpha-2/	2022-11-1
hive-standalone-metastore-3.0.0/	2022-06-1
hive-storage-2.7.3/	2022-06-1
hive-storage-2.8.1/	2022-06-1
stable-2/	2022-06-1
KEYS	2022-10-2

Releases may be downloaded from Apache mirrors. [Download a release now](#)

recent releases are available, but are not guaranteed to be stable. For stable releases, see the [stable directory](#).

News

- 14 August 2023: release 4.0.0-beta-1 available
 - This release works with Hadoop 3.3.1
 - You can look at the complete [JIRA change log for this release](#).
- 16 November 2022: release 4.0.0-alpha-2 available
 - This release works with Hadoop 3.3.1
 - You can look at the complete [JIRA change log for this release](#).
- 08 April 2022: release 3.1.3 available
 - This release works with Hadoop 3.x.y
 - You can look at the complete [JIRA change log for this release](#).
- 30 March 2022: release 4.0.0-alpha-1 available
 - This release works with Hadoop 3.x.y

<https://dlcdn.apache.org/hive/hive-3.1.3/>

/hive-3.1.3

<u>Last modified</u>	<u>Size</u>	<u>Description</u>
-	-	-
2022-04-08 17:42	312M	-
asc	2022-04-08 17:42	488
sha256	2022-04-08 17:42	95
-	-	-
asc	2022-04-08 17:42	488
sha256	2022-04-08 17:42	95

5. INSTALACION HIVE

Paso 2. Descomprimimos el fichero tar.gz

tar xvf apache-hive.3.1.3.bin.tar.gz

```
hadoop@nodo1:~/Downloads$ ls -l
total 1958232
-rw-rw-r-- 1 hadoop hadoop 504941532 mar  5 13:02 access_log
drwxrwxr-x 10 hadoop hadoop     4096 mar 15 13:02 apache-hive-3.1.3-bin
-rw-rw-r-- 1 hadoop hadoop 326940667 mar 15 12:55 apache-hive-3.1.3-bin.tar.gz
-rw-rw-r-- 1 hadoop hadoop 264075431 mar 12 08:37 cite75_99.txt
-rw-rw-r-- 1 hadoop hadoop 414624228 ene  4 12:09 hadoop-2.10.2.tar.gz
-rw-rw-r-- 1 hadoop hadoop 492368219 feb 23 19:10 hadoop-3.2.4.tar.gz
-rw-rw-r-- 1 hadoop hadoop     1490 mar 12 08:56 'MyJob$MapClass.class'
-rw-rw-r-- 1 hadoop hadoop     1815 mar 12 08:56 'MyJob$Reduce.class'
-rw-rw-r-- 1 hadoop hadoop     2028 mar 12 08:56 MyJob.class
-rw-rw-r-- 1 hadoop hadoop     3876 mar 12 08:58 MyJob.jar
-rw-rw-r-- 1 hadoop hadoop    2293 mar 12 08:45 MyJob.java
-rw-rw-r-- 1 hadoop hadoop  2226045 feb 24 22:59 quijote.txt
hadoop@nodo1:~/Downloads$ tar xvf apache-hive-3.1.3-bin.tar.gz
```

Paso 3. Renombramos el directorio resultado a hive únicamente

```
hadoop@nodo1:~/Downloads$ mv apache-hive-3.1.3-bin hive
hadoop@nodo1:~/Downloads$
```

5. INSTALACION HIVE

Paso 4. Movemos el directorio hive a /opt/hadoop. Dentro del directorio /opt/hadoop vamos a tener un directorio llamado hive

```
hadoop@nodo1:~/Downloads$ mv hive /opt/hadoop
hadoop@nodo1:~/Downloads$
```

NOTA: Aunque en la vida real es posible disponer de un usuario especial para trabajar con hive, usaremos el mismo usuario hadoop

Paso 5. Es importante agregar en el fichero .bashrc la variable de entorno HIVE_HOME. Además añadimos en el PATH el directorio bin de Hive

```
GNU nano 6.2                                .bashrc *
[ -r /home/hadoop/.config/byobu/prompt ] && . /home/hadoop/.config/byobu/prompt

export HADOOP_HOME=/opt/hadoop
export HIVE_HOME=/opt/hadoop/hive
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$HIVE_HOME/bin
```

5. INSTALACION HIVE

Paso 6. En /opt/hadoop/hive encontramos una serie de directorios: una serie de ejemplos, paquetes binarios, librerías, drivers JDBC, etc

```
hadoop@nodo1:/opt/hadoop/hive$ ls -l
total 80
drwxrwxr-x 3 hadoop hadoop 4096 mar 15 13:02 bin
drwxrwxr-x 2 hadoop hadoop 4096 mar 15 13:02 binary-package-licenses
drwxrwxr-x 2 hadoop hadoop 4096 mar 15 13:02 conf
drwxrwxr-x 4 hadoop hadoop 4096 mar 15 13:02 examples
drwxrwxr-x 7 hadoop hadoop 4096 mar 15 13:02 hcatalog
drwxrwxr-x 2 hadoop hadoop 4096 mar 15 13:02 jdbc
drwxrwxr-x 4 hadoop hadoop 20480 mar 15 13:02 lib
-rw-r--r-- 1 hadoop hadoop 20798 mar 28 2022 LICENSE
-rw-r--r-- 1 hadoop hadoop 230 mar 28 2022 NOTICE
-rw-r--r-- 1 hadoop hadoop 540 mar 28 2022 RELEASE_NOTES.txt
drwxrwxr-x 4 hadoop hadoop 4096 mar 15 13:02 scripts
hadoop@nodo1:/opt/hadoop/hive$ █
```

5. INSTALACION HIVE

Paso 7. Dentro del bin tenemos los binarios más importantes:

- hive → es la herramienta cliente en modo local
- beeline → es la otra herramienta cliente en modo comando
- hiveserver2 → para ejecutar el servidor de hive
- schematool → nos permite trabajar contra la base de datos de metadatos o repositorio MetaStore

```
hadoop@nodo1:/opt/hadoop/hive/bin$ ls -l
total 44
-rwxr-xr-x 1 hadoop hadoop  881 oct 24  2019 beeline
drwxrwxr-x 3 hadoop hadoop 4096 mar 15 13:02 ext
-rwxr-xr-x 1 hadoop hadoop 10158 mar 28  2022 hive
-rwxr-xr-x 1 hadoop hadoop 2085 feb 27  2022 hive-config.sh
-rwxr-xr-x 1 hadoop hadoop  885 oct 24  2019 hiveserver2
-rwxr-xr-x 1 hadoop hadoop  880 oct 24  2019 hqlsql
-rwxr-xr-x 1 hadoop hadoop 3064 oct 24  2019 init-hive-dfs.sh
-rwxr-xr-x 1 hadoop hadoop  832 oct 24  2019 metatool
-rwxr-xr-x 1 hadoop hadoop  884 oct 24  2019 schematool
hadoop@nodo1:/opt/hadoop/hive/bin$ █
```

5. INSTALACION HIVE

Paso 8. Vamos al directorio de Hive y aquí nos encontramos con que hay un montón de ficheros pero todos terminan en template. Algunos de estos los tenemos que convertir a ficheros predefinidos.

```
hadoop@nodo1:/opt/hadoop/hive/conf$ ls -l
total 332
-rw-r--r-- 1 hadoop hadoop 1596 oct 24 2019 beeline-log4j2.properties.template
-rw-r--r-- 1 hadoop hadoop 300727 abr  3 2022 hive-default.xml.template
-rw-r--r-- 1 hadoop hadoop 2365 oct 24 2019 hive-env.sh.template
-rw-r--r-- 1 hadoop hadoop 2274 oct 24 2019 hive-exec-log4j2.properties.template
-rw-r--r-- 1 hadoop hadoop 3086 sep  4 2020 hive-log4j2.properties.template
-rw-r--r-- 1 hadoop hadoop 2060 oct 24 2019 ivysettings.xml
-rw-r--r-- 1 hadoop hadoop 3558 sep  4 2020 llap-cli-log4j2.properties.template
-rw-r--r-- 1 hadoop hadoop 6937 mar 28 2022 llap-daemon-log4j2.properties.template
-rw-r--r-- 1 hadoop hadoop 2662 oct 24 2019 parquet-logging.properties
hadoop@nodo1:/opt/hadoop/hive/conf$
```

Paso 9. Hacemos una copia del fichero `hive-default.xml.template` y lo llamaremos `hive-site.xml`. Este es el fichero primario de configuración de hive. Le indicaremos una serie de propiedades muy parecidas a las de Hadoop

```
hadoop@nodo1:/opt/hadoop/hive/conf$ cp hive-default.xml.template hive-site.xml
hadoop@nodo1:/opt/hadoop/hive/conf$
```

5. INSTALACION HIVE

Paso 10. Hacemos una copia de `hive-env.sh.template` en `hive-env.sh`. Este fichero se invoca por los distintos ejecutables de Hive que hemos visto en el directorio bin que se ejecuta para encontrar las variables de entorno.

```
hadoop@nodo1:/opt/hadoop/hive/conf$ cp hive-env.sh.template hive-env.sh
hadoop@nodo1:/opt/hadoop/hive/conf$
```

Paso 11. También se debe de copiar los dos ficheros de log, quitándole la terminación template.

```
hadoop@nodo1:/opt/hadoop/hive/conf$ cp hive-exec-log4j2.properties.template hive-exec-log4j2.properties
hadoop@nodo1:/opt/hadoop/hive/conf$ cp hive-log4j2.properties.template hive-log4j2.properties
hadoop@nodo1:/opt/hadoop/hive/conf$ █
```

Paso 12. Por último copiamos el fichero beeline, quitando template

```
hadoop@nodo1:/opt/hadoop/hive/conf$ cp beeline-log4j2.properties.template beeline-log4j2.properties
hadoop@nodo1:/opt/hadoop/hive/conf$ █
```

5. INSTALACION HIVE

Paso 13. En resumen, a todos se les cambia el nombre quitándole el template, menos al fichero default que le hemos llamado hive-site

```
hadoop@nodo1:/opt/hadoop/hive/conf$ ls -l
total 644
-rw-r--r-- 1 hadoop hadoop 1596 mar 15 19:36 beeline-log4j2.properties
-rw-r--r-- 1 hadoop hadoop 1596 oct 24 2019 beeline-log4j2.properties.template
-rw-r--r-- 1 hadoop hadoop 300727 abr 3 2022 hive-default.xml.template
-rw-r--r-- 1 hadoop hadoop 2365 mar 15 19:17 hive-env.sh
-rw-r--r-- 1 hadoop hadoop 2365 oct 24 2019 hive-env.sh.template
-rw-r--r-- 1 hadoop hadoop 2274 mar 15 19:43 hive-exec-log4j2.properties
-rw-r--r-- 1 hadoop hadoop 2274 oct 24 2019 hive-exec-log4j2.properties.template
-rw-r--r-- 1 hadoop hadoop 3086 mar 15 19:28 hive-log4j2.properties
-rw-r--r-- 1 hadoop hadoop 3086 sep 4 2020 hive-log4j2.properties.template
-rw-r--r-- 1 hadoop hadoop 300727 mar 15 19:04 hive-site.xml
-rw-r--r-- 1 hadoop hadoop 2060 oct 24 2019 ivysettings.xml
-rw-r--r-- 1 hadoop hadoop 3558 sep 4 2020 llap-cli-log4j2.properties.template
-rw-r--r-- 1 hadoop hadoop 6937 mar 28 2022 llap-daemon-log4j2.properties.template
-rw-r--r-- 1 hadoop hadoop 2662 oct 24 2019 parquet-logging.properties
hadoop@nodo1:/opt/hadoop/hive/conf$
```

5. INSTALACION HIVE

Paso 14. Tenemos que entrar en `hive-env.sh` y definir dos variables

`export HADOOP_HOME=/opt/hadoop`

`export HIVE_CONF_DIR=/opt/hadoop/hive/conf`

```
GNU nano 6.2                                     hive-env.sh

# The heap size of the jvm stared by hive shell script can be
#
# export HADOOP_HEAPSIZE=1024
export HADOOP_HOME=/opt/hadoop
export HIVE_CONF_DIR=/opt/hadoop/hive/conf
```

5. INSTALACION HIVE

Paso 15. Creamos dos directorios en HDFS que Hive usará cuando se creen tablas y objetos. Hive ofrece un frontal SQL pero por debajo realmente ataca HDFS. Creamos el directorio /tmp dentro de HDFS, uno donde por defecto va a trabajar Hive.

hdfs dfs -mkdir /tmp

hdfs dfs -chmod g+w /tmp

hdfs dfs -chmod 777 /tmp → damos permisos a este directorio para que Hive pueda trabajar con él.

```
hadoop@nodo1:~$ hdfs dfs -mkdir /tmp
hadoop@nodo1:~$ hdfs dfs -chmod g+w /tmp
hadoop@nodo1:~$ hdfs dfs -ls /
Found 3 items
drwxr-xr-x  - hadoop supergroup          0 2023-03-12 08:39 /practicas
drwxr-xr-x  - hadoop supergroup          0 2023-03-09 16:04 /prueba
drwxrwxr-x  - hadoop supergroup          0 2023-03-17 22:41 /tmp
hadoop@nodo1:~$ █
```

Nota: Trabajaremos con el usuario hadoop. Si tuviéramos otros usuarios para trabajar con hive, tendríamos que hacer que pertenecieran también al grupo de hadoop para no tener problemas₂₁

5. INSTALACION HIVE

Paso 16. Creamos el directorio /user/hive/warehouse

hdfs dfs -mkdir -p /user/hive/warehouse

hdfs dfs -chmod g+w /user/hive/warehouse

hdfs dfs -chmod 777 /user/hive/warehouse

Debe ser exactamente ese nombre. Por defecto Hive escribe en ese sitio concreto de HDFS

```
hadoop@nodo1:~$ hdfs dfs -ls /
Found 4 items
drwxr-xr-x    - hadoop supergroup          0 2023-03-12 08:39 /practicas
drwxr-xr-x    - hadoop supergroup          0 2023-03-09 16:04 /prueba
drwxrwxr-x    - hadoop supergroup          0 2023-03-17 22:41 /tmp
drwxr-xr-x    - hadoop supergroup          0 2023-03-17 22:45 /user
hadoop@nodo1:~$
```

5. INSTALACION HIVE

Paso 17. Podemos ver los directorios a través de la web HDFS, menú Browse the file system

Comprobamos que los directorios /tmp y /user/hive/warehouse tienen los permisos adecuados para trabajar bien.

Browse Directory

/

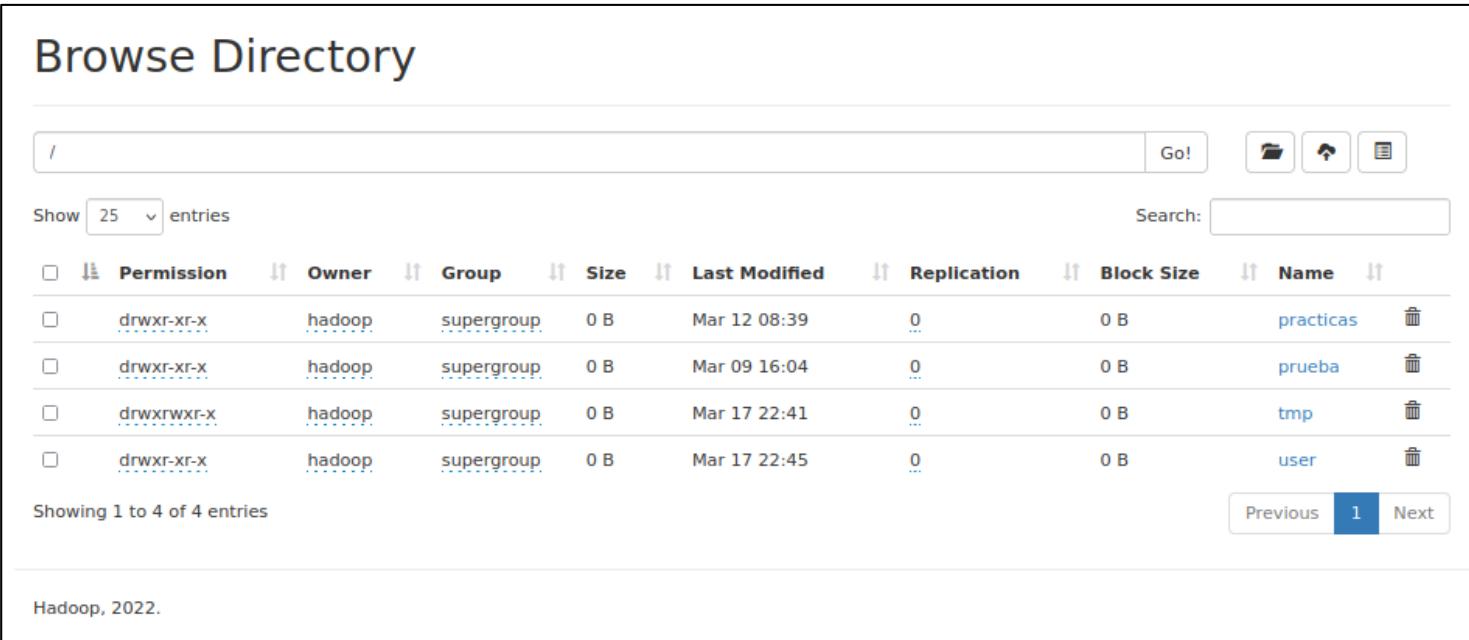
Show 25 entries

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Mar 12 08:39	0	0 B	practicas	
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Mar 09 16:04	0	0 B	prueba	
<input type="checkbox"/>	drwxrwxr-x	hadoop	supergroup	0 B	Mar 17 22:41	0	0 B	tmp	
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Mar 17 22:45	0	0 B	user	

Showing 1 to 4 of 4 entries

Previous 1 Next

Hadoop, 2022.



6. COMANDOS HIVESQL

Comandos DDL de HiveSql

- ❖ HiveSQL es un lenguaje muy parecido a SQL, usado por Hive.
- ❖ Los comandos de tipo DDL nos permiten crear objetos: tablas, base de datos, particiones, vistas, índices, macros, etc.
- ❖ Dentro de Hive podemos construir los metadatos que se guardan en el MetaStore y contienen la información lógica de Hive.
 - Create table
 - Create/Drop/Alter/Use Database
 - Create/Drop/Truncate Table
 - Alter Table/Partition/Column
 - Create/Drop/Alter View
 - Create/Drop/Alter Index
 - Create/Drop Macro

```
create table table_name
(
  id int,
  dtDontQuery string,
  Name string
)
partitioned by (date string)
```

```
Create database db1
```

6. COMANDOS HIVESQL

Comandos DML de HiveSql

- ❖ HiveSQL contiene comandos DML tales como: INSERT, DELETE, UPDATE, MERGE, LOAD
- ❖ El comando LOAD permite cargar datos de ficheros del S.O. al sistema de ficheros HDFS, asociándolo a una tabla de Hive.
- ❖ Permite hacer cargas masivas de datos de una forma mas rápida y eficiente que el comando insert

```
LOAD DATA LOCAL INPATH
'/home/curso/Escritorio/employee.txt' OVERWRITE
INTO TABLE employee_internal;
```

```
INSERT INTO TABLE students
VALUES ('fred flintstone', 35, 1.28), ('barney rubble', 32,
2.32);
```

6. COMANDOS HIVESQL

Comandos DML de HiveSql

- ❖ Hive soporta todas aquellas operaciones de tipo transaccional que tienen el resto de bases de datos: Insert, delete, merge
- ❖ Merge es muy similar a los merges de base datos normales con la cláusulas using y when.

```
DELETE FROM students WHERE gpa <= 1,0;
```

```
merge into customer
using ( select * from
new_customer_stage) sub on sub.id =
customer.id
when matched then update set name =
sub.name, state = sub.new_state
when not matched then insert values (sub.id,
sub.name, sub.state);
```

6. COMANDOS HIVESQL

Queries en HiveSQL

- El comando `SELECT` tiene unas cláusulas muy parecidas a base de datos normales: `FROM`, `WHERE`, `GROUP BY`, `HAVING`, `SORT`
- Esto hace que la curva de aprendizaje de Hive sea bastante ligera, porque salvando las distancias con el tipo de almacenamiento, la filosofía de HiveSQL es muy parecido al SQL convencional

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list] [HAVING
having_condition]
[CLUSTER BY col_list | [DISTRIBUTE BY col_list] [SORT BY
col_list]] [LIMIT number]
;
```

6. COMANDOS HIVESQL

Funciones de HiveSql

- Dentro de Hive tenemos un conjunto de funciones típicas: aritméticas, de colección, de cadena etc. etc.
- No tiene tantas como las bases de datos tradicionales, pero tiene un conjunto de funciones lo suficientemente grande como para poder trabajar sin problemas con ellas.

Mathematical Functions
Collection Functions
Type Conversion Functions
Date Functions
Conditional Functions
String Functions
Misc Functions

6. COMANDOS HIVESQL

Ejemplos de comandos Select

- a) Comando select normal con una condición con un where
- b) comando Select con un orderby para hacer una ordenación
- c) Comando select con un group by para poder sacar el nº de empleados por departamento.
- d) Una típica join entre dos tablas customers y orders.
- e) Una join tipo left outer join

```
SELECT * FROM employee WHERE salary>30000;
```

```
SELECT Id, Name, Dept FROM employee ORDER BY DEPT;
```

```
SELECT Dept,count(*) FROM employee GROUP BY DEPT;
```

```
SELECT c.ID, c.NAME, c.AGE, o.AMOUNT FROM CUSTOMERS c JOIN  
ORDERS o ON (c.ID = o.CUSTOMER_ID);
```

```
SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM  
CUSTOMERS c LEFT OUTER JOIN ORDERS o  
ON (c.ID = o.CUSTOMER_ID);
```

6. COMANDOS HIVESQL

Ventajas de HiveSql

- La mayor parte de los comandos de HiveSql son muy parecidos a los de cualquier base de datos relacional.
- La diferencia para un programador que tiene que atacar un sistema Big Data el poder hacerlo con queries al SQL a tener que hacerlo con algún programa en Java en Python .
- Evidentemente para analistas de datos gente acostumbrada a trabajar con SQL este tipo de base de datos como Hive supone una facilidad de trabajo enorme

7. CREACION BASE DATOS

Vamos a crear y a arrancar nuestro primer programa.

Paso 1. En el directorio hive/bin vemos las dos formas de trabajar.

```
hadoop@nodo1:/opt/hadoop/hive/bin$ ls -l
total 44
-rwxr-xr-x 1 hadoop hadoop 881 oct 24 2019 beeline
drwxrwxr-x 3 hadoop hadoop 4096 mar 15 13:02 ext
-rwxr-xr-x 1 hadoop hadoop 10158 mar 28 2022 hive
-rwxr-xr-x 1 hadoop hadoop 2085 feb 27 2022 hive-config.sh
-rwxr-xr-x 1 hadoop hadoop 885 oct 24 2019 hiveserver2
-rwxr-xr-x 1 hadoop hadoop 880 oct 24 2019 hql
-rwxr-xr-x 1 hadoop hadoop 3064 oct 24 2019 init-hive-dfs.sh
-rwxr-xr-x 1 hadoop hadoop 832 oct 24 2019 metatool
-rwxr-xr-x 1 hadoop hadoop 884 oct 24 2019 schematool
hadoop@nodo1:/opt/hadoop/hive/bin$ █
```

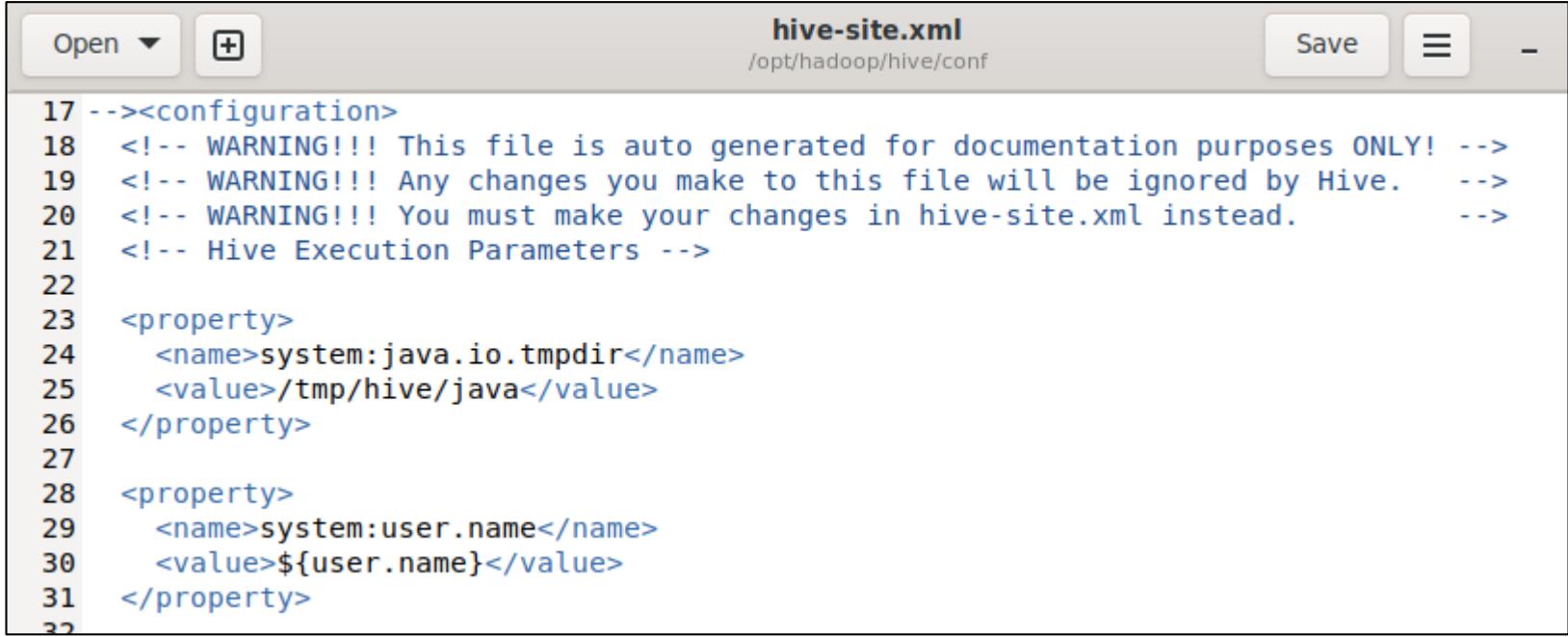
Forma local: Usaremos el cliente hive, un cliente ligero que trabaja en modo local. Se conecta a una base de datos localmente y sólo podemos tener una sesión al mismo tiempo. Sera el utilizado

Forma remota: Hiveserver2 nos permite arrancar un servidor al cual nos podíamos conectar en remoto y para ello utilizamos la herramienta llamada beeline

7. CREACION BASE DATOS

Paso 2. Accedemos al fichero `hive-site.xml` y ponemos dos propiedades al principio del fichero:

- `java.io.tmpdir` → permite crear el directorio `/tmp/hive/java`
- `user.name` → lo selecciona del usuario en el que estamos trabajando en este momento



The screenshot shows a code editor window with the following details:

- Title Bar:** The title is "hive-site.xml" located at "/opt/hadoop/hive/conf".
- Buttons:** On the left are "Open" with a dropdown arrow and a "+" button. On the right are "Save", a three-line menu icon, and a close button.
- Code Area:** The code is displayed in a monospaced font. It includes several XML-style comments and properties. Lines 17-21 are comments about the file being auto-generated. Lines 23-31 define two properties: "java.io.tmpdir" with value "/tmp/hive/java" and "user.name" with value "\${user.name}".

```
17 --><configuration>
18   <!-- WARNING!!! This file is auto generated for documentation purposes ONLY! -->
19   <!-- WARNING!!! Any changes you make to this file will be ignored by Hive. -->
20   <!-- WARNING!!! You must make your changes in hive-site.xml instead. -->
21   <!-- Hive Execution Parameters -->
22
23 <property>
24   <name>system:java.io.tmpdir</name>
25   <value>/tmp/hive/java</value>
26 </property>
27
28 <property>
29   <name>system:user.name</name>
30   <value>${user.name}</value>
31 </property>
32
```

7. CREACION BASE DATOS

Paso 3. Vamos a la carpeta /opt/hadoop/hive/lib y comprobamos la versión del fichero guava (es la versión 19)

```
hadoop@nodo1:/opt/hadoop/hive/lib$ ls guava*
guava-19.0.jar
hadoop@nodo1:/opt/hadoop/hive/lib$ █
```

En la carpeta /opt/hadoop/share/hadoop/common/lib miramos la versión de guava (es la versión 27.0)

```
hadoop@nodo1:/opt/hadoop/share/hadoop/common/lib$ ls guava*
guava-27.0-jre.jar
hadoop@nodo1:/opt/hadoop/share/hadoop/common/lib$ █
```

Si no son iguales, hay que eliminar la versión más antigua y copiar la versión más nueva en ambos directorios. Eliminar la versión 19 en la carpeta lib de Hive y luego copiar la versión 27 de la carpeta Hadoop a Hive

```
hadoop@nodo1:/opt/hadoop/share/hadoop/common/lib$ cp guava-27.0-jre.jar /opt/hadoop/hive/lib
hadoop@nodo1:/opt/hadoop/share/hadoop/common/lib$ rm /opt/hadoop/hive/lib/guava-19.0.jar
hadoop@nodo1:/opt/hadoop/share/hadoop/common/lib$ █
```

7. CREACION BASE DATOS

Paso 4. Creamos el directorio `hive/bbdd` donde guardar la metastore. Aquí se creará la base datos Derby local. La mejor opción es la de trabajar con un hiveserver y atacar a una bd como Mysql

```
1 hadoop@nodo1:/opt/hadoop/hive$ mkdir bbdd
hadoop@nodo1:/opt/hadoop/hive$ cd bbdd/
hadoop@nodo1:/opt/hadoop/hive/bbdd$
```

Paso 5. Lanzamos `schematool -dbType derby -initSchema` en la carpeta `bbdd`, para crear e inicializar el esquema de la base datos Derby con la que trabajar. Se ve la conexión con el MetaStore

```
hadoop@nodo1:/opt/hadoop/hive/bbdd$ schematool -dbType derby -initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hive/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerFactory.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.35.jar!/org/slf4j/impl/StaticLoggerFactory.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:      jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :  org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:    APP
Starting metastore schema initialization to 3.1.0
Initialization script hive-schema-3.1.0.derby.sql
```

```
Initialization script completed
schemaTool completed
hadoop@nodo1:/opt/hadoop/hive/bbdd$
```

7. CREACION BASE DATOS

Paso 6. Observamos que se ha generado un fichero derby.log que es un log de la base datos, y el directorio metastore_db que es la base datos Derby. Indica que no se debe de tocar.

```

hadoop@nodo1:/opt/hadoop/hive/bbdd$ ls -l
total 24
-rw-rw-r-- 1 hadoop hadoop 19985 mar 18 14:35 derby.log
drwxrwxr-x 5 hadoop hadoop 4096 mar 18 14:35 metastore_db
hadoop@nodo1:/opt/hadoop/hive/bbdd$ ls metastore_db/
dbex.lck db.lck log README_DO_NOT_TOUCH_FILES.txt seg0 service.properties tmp
hadoop@nodo1:/opt/hadoop/hive/bbdd$ 
```

Paso 7. Atacamos el metastore con el cliente Hive. Nos abre una sesión de trabajo en línea de comandos para empezar a trabajar. Nos dice que Hive-on-MR es deprecated sobre MapReduce y que conviene utilizar sobre spark o tez. Los comandos en si no van a cambiar

```

hadoop@nodo1:/opt/hadoop/hive/bbdd$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hive/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.35.jar!/org/slf4j/impl/StaticLoggerB
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = dada5b74-070e-46aa-b8d4-2adf69152366

Logging initialized using configuration in file:/opt/hadoop/hive/conf/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engi
e 1.X releases.Hive Session ID = 49dea487-bdb3-4103-8442-eb60e83f1d92

hive> 
```

7. CREACION BASE DATOS

Paso 8. Dado que esto es una simulación de una base de datos relacional, el primer comando será el de crear una base de datos **create DATABASE.ejemplo;** → Comandos HiveSql terminan en :

```
hive> create database ejemplo;
OK
Time taken: 0.318 seconds
hive> █
```

La base de datos ejemplo se ha repartido de la siguiente manera:

- En la carpeta `/opt/hadoop/hive/bbdd` se ha guardado la información de los metadatos de los objetos del metastore
- Y en la carpeta HDFS `/user/hive/warehouse` es donde se ha creado la base de datos, en un directorio con el nombre `ejemplo.db`

Browse Directory

/user/hive/warehouse

Show 25 entries

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop	supergroup	0 B	Mar 18 17:08	0	0 B	ejemplo.db

Showing 1 to 1 of 1 entries

Go!

Search:

Previous **1** Next

7. CREACION BASE DATOS

Paso 9. Trabajaremos con comandos similares a mysql

show databases; → Podemos ver las bases de datos dentro de Hive
La base de datos default sirve para que cuando yo crea algo y no le digo específicamente una database, automáticamente usa default.

use ejemplo; → me conecto a la base de datos ejemplo

show tables; → muestra las tablas que hay dentro de la base de datos ejemplo (vemos que no tenemos ahora misma ninguna tabla)

```
hive> show databases;
OK
default
ejemplo
Time taken: 0.145 seconds, Fetched: 2 row(s)
hive> use ejemplo;
OK
Time taken: 0.016 seconds
hive> show tables;
OK
Time taken: 0.021 seconds
hive>
```

7. CREACION BASE DATOS

Paso 10. Creamos una tabla:

create table if not exists t1

```
(  
    name string  
);  
show tables;  
desc t1;
```

```
hive> create table if not exists t1  
    > (  
    > name string  
    > );  
OK  
Time taken: 0.392 seconds  
hive> show tables;  
OK  
t1  
Time taken: 0.018 seconds, Fetched: 1 row(s)  
hive> desc t1;  
OK  
name                      string  
Time taken: 0.085 seconds, Fetched: 1 row(s)  
hive> █
```

NOTA: Recordar que esto no es una base relacional que va a atacar a ficheros que hay en HFS entonces esos ficheros tienen un formato concreto. Y se lo tenemos que decir de alguna manera

7. CREACION BASE DATOS

Paso 11. Si vamos a hdfs/Browse the file system, y entramos dentro del directorio ejemplo.db, tenemos otro directorio llamado t1 y dentro de este directorio es donde tendremos los ficheros HDFS asociados a esta tabla.

Browse Directory

/user/hive/warehouse/ejemplo.db   

Show 25 entries

<input type="checkbox"/>	<input type="button" value="Permission"/>	<input type="button" value="Owner"/>	<input type="button" value="Group"/>	<input type="button" value="Size"/>	<input type="button" value="Last Modified"/>	<input type="button" value="Replication"/>	<input type="button" value="Block Size"/>	<input type="button" value="Name"/>	
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Mar 18 18:07	0	0 B	t1	

Showing 1 to 1 of 1 entries

Hadoop, 2022.

NOTA: Trataremos todo como si fuera una tabla normal aunque luego por debajo él haga determinados comandos de tipo MapReduce o Spark o Tez dependiendo del motor que utilicemos.

7. CREACION BASE DATOS

Paso 12. Salimos con **quit**; y volvemos a entrar con **hive**. Creamos una nueva tabla sin mas:

create table t2 (codigo integer);

¿Donde se ha creado t2 ya que no nos hemos conectado a ninguna base de datos? → se habrá creado en la bd por defecto (error típico)

```
hive> quit;
hadoop@nodo1:/opt/hadoop/hive/bbdd$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hive/lib/log4j-slf4j-impl-2.17.1.jar!
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-reload4
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 0eae62cf-fe67-420f-a439-8c06751465b8

Logging initialized using configuration in file:/opt/hadoop/hive/conf/hive-log4j2.p
Hive-on-MR is deprecated in Hive 2 and m
e 1.X releases.
Hive Session ID = 49b1bde0-efd6-4df3-8ba
hive> create table t2
    > (
    > codigo integer
    > );
OK
Time taken: 0.667 seconds
hive> show tables;
OK
t2
Time taken: 0.078 seconds, Fetched: 1 ro
hive>
```

Browse Directory										
<input type="text" value="/user/hive/warehouse"/> Go!    										
Show 25 entries Search: <input type="text"/>										
	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name		
	drwxr-xr-x	hadoop	supergroup	0 B	Mar 18 18:07	0	0 B	ejemplo.db		
	drwxr-xr-x	hadoop	supergroup	0 B	Mar 18 19:05	0	0 B	t2		

Showing 1 to 2 of 2 entries

Previous 1 Next

Hadoop, 2022.

7. CREACION BASE DATOS

Paso 13. Creamos table t2 en ejemplo.

`show databases;`

`use ejemplo;`

`create table t2 (codigo integer);`

```
hive> show databases;
OK
default
ejemplo
Time taken: 0.016 seconds, Fetched: 2 row(s)
hive> use ejemplo;
OK
Time taken: 0.016 seconds
hive> create table t2
      > (
      > codigo integer
      > );
OK
Time taken: 0.049 seconds
hive> show tables;
OK
t1
t2
Time taken: 0.017 seconds, Fetched
hive>
```

En HDFS tenemos las dos tablas en la base de datos ejemplo

Browse Directory												
/user/hive/warehouse/ejemplo.db												
Show	25	v	entries								Search:	
□	⬆️	Permission	⬆️	Owner	⬆️	Group	⬆️	Size	⬆️	Last Modified	⬆️	Replication
□	drwxr-xr-x		hadoop		supergroup		0 B		Mar 18 18:07		0	0 B
□	drwxr-xr-x		hadoop		supergroup		0 B		Mar 18 19:35		0	0 B

Showing 1 to 2 of 2 entries

Previous 1 Next

Hadoop, 2022.

7. CREACION BASE DATOS

Paso 14. Aquí podemos hacer casi todas las operaciones permitidas en SQL, incluido un insert: **insert into t2 values (10);** → Va a lanzar un programa MapReduce: lanza un job, de un 1 mapper y 1 reducer.

```

hive> insert into t2 values (10);
Query ID = hadoop_20230318194650_09f53a60-cdbd-4c18-acfc-25bfb2b46175
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1679153125332_0001, Tracking URL = http://nodo1:8088/proxy/application_1679153125332_0001
Kill Command = /opt/hadoop/bin/mapred job -kill job_1679153125332_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-03-18 19:46:58,933 Stage-1 map = 0%,  reduce = 0%
2023-03-18 19:47:04,078 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.6 sec
2023-03-18 19:47:10,216 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.89 sec
MapReduce Total cumulative CPU time: 3 seconds 890 msec
Ended Job = job_1679153125332_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://nodo1:9000/user/hive/warehouse/ejemplo.db/t2/.hive-stagi
Loading data to table ejemplo.t2
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1  Cumulative CPU: 3.89 sec    HDFS Read: 12808 HDFS Writ
Total MapReduce CPU Time Spent: 3 seconds 890 msec
OK
Time taken: 21.392 seconds
hive>
```

Este proceso tarda.
Los inserts no se deberían hacer así habitualmente sino intentar cargarlo de manera masiva con otras herramientas y luego acceder a la tabla

7. CREACION BASE DATOS

Paso 15. Si vamos a HDFS, vemos que Hive ha metido dentro un fichero. Que contiene la información que yo le acabo de cargar.

Browse Directory

/user/hive/warehouse/ejemplo.db/t2

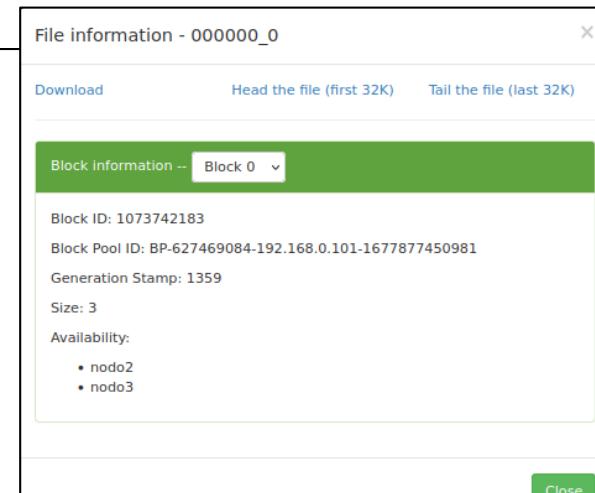
Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	3 B	Mar 18 19:47	2	128 MB	000000_0

Showing 1 to 1 of 1 entries

Previous 1 Next

Hadoop, 2022.



7. CREACION BASE DATOS

Paso 16. Si hacemos select * from t2; como una query normal nos devuelve un 10

```
hive> select * from t2;
OK
10
Time taken: 0.117 seconds, Fetched: 1 row(s)
hive>
```

Paso 17. Si hacemos un cat del fichero que tenemos en /user/hive/warehouse/ejemplo.db/t2/000000_0, nos devuelve el contenido

```
hadoop@nodo1:~$ hdfs dfs -cat /user/hive/warehouse/ejemplo.db/t2/000000_0
10
hadoop@nodo1:~$ █
```

NOTA: Hive va a guardar aquí información dentro de la base de datos. Esto se denomina tablas internas

8. TABLAS INTERNAS Y LOAD

Paso 1. Hive tiene dos tipos de tablas: Internas y externas

- **Tablas internas:** todo el ciclo de vida de una tabla interna, su información, contenidos y determinados comandos los gestiona el propio Hive. Si creamos una tabla interna, la cargamos con datos y cuando desde Hive borramos la tabla, se borran también los ficheros asociados. Se tiene un poco más de control sobre el ciclo de vida de la tabla
- **Tablas externas:** Cuando la tabla es externa, se supone que ya tenemos unos datos que se han cargado de otra manera y que seguramente estén siendo utilizados por otros productos no sólo por hive y que también Hive tiene que utilizar. En este caso no es Hive el que controla el ciclo de vida de esa tabla por lo tanto si yo borro la tabla realmente no se borran

8. TABLAS INTERNAS Y LOAD

Paso 2. Vamos a crear en primer lugar una tabla interna. Abrimos otra sesión de terminal y en la otra dejamos abierto Hive. Creamos un fichero sencillo empleados.txt. Pondremos nombre y edad

```

hadoop@n... x hadoop@n... x + ▾
GNU nano 6.2  empleados.txt
Rosa,50
Pedro,60
Raul,56
Maria,35

```

Paso 3. Volvemos a Hive y creamos una tabla interna (no llevará la clausula external)

```

hive> create table empleados (nombre string, edad integer)
      > row format delimited
      > fields terminated by ',';
OK
Time taken: 0.046 seconds
hive>

```

use ejemplo:

create table empleados (nombre string, edad integer)

row format delimited → Indica que cada línea del fichero es una fila

fields terminated by ',' : → Indica que el carácter divisor de las columnas es el punto y coma. Lo correcto sería que invocara fichero

8. TABLAS INTERNAS Y LOAD

Paso 4. Si hacemos show tables me debe aparecer ya la tabla.

```
hive> show tables;
OK
empleados
t1
t2
Time taken: 0.017 seconds, Fetched: 3 row(s)
hive>
```

Paso 5. Si hacemos un refresh en Browse Directory deberíamos tener la tabla empleados creada.

Browse Directory

/user/hive/warehouse/ejemplo.db

Show 25 entries

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Mar 19 09:19	0	0 B	empleados	
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Mar 18 18:07	0	0 B	t1	
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Mar 18 19:47	0	0 B	t2	

Showing 1 to 3 of 3 entries

Previous **1** Next

Hadoop, 2022.

8. TABLAS INTERNAS Y LOAD

Paso 6. Si Ahora no lo haremos con un insert porque llama a MapReduce y es un proceso muy lento. Asociaremos el fichero empleados directamente con la tabla. Lo podemos cargar desde Linux o también desde HDFS. Copiaremos empleados.txt a /tmp

```
hadoop@nodo1:~$ cp empleados.txt /tmp  
hadoop@nodo1:~$
```

Paso 7. Cargamos el fichero /tmp/empleados.txt en la base de datos empleados mediante el comando load data

Load data local inpath '/tmp/empleados.txt' into table empleados;

```
hive> load data local inpath '/tmp/empleados.txt' into table empleados;  
Loading data to table ejemplo.empleados  
OK  
Time taken: 0.154 seconds  
hive>
```

8. TABLAS INTERNAS Y LOAD

Paso 8. Hacemos un select * from empleados; sale sin problemas el nombre y la edad en columnas.

```
hive> select * from empleados;
OK
Rosa      50
Pedro     60
Raul      56
Maria     35
Time taken: 0.083 seconds, Fetched: 4 row(s)
hive> █
```

Paso 9. Si vamos al Browse de HDFS, dentro de la tabla empleados tenemos el fichero empleados.txt

Browse Directory

/user/hive/warehouse/ejemplo.db/empleados

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	34 B	Mar 19 10:07	2	128 MB	empleados.txt

Showing 1 to 1 of 1 entries

Previous 1 Next

Hadoop, 2022.

8. TABLAS INTERNAS Y LOAD

Paso 10. Podemos seguir cargando cosas. Copiamos empleados.txt en empleados1.txt. Como no hay clave primaria no va a haber problema. Cargamos otra vez el fichero empleados1.txt. Y vemos que un select me va a sacar los datos repetidos

```
hadoop@nodo1:~$ cp /tmp/empleados.txt /tmp/empleados1.txt
hadoop@nodo1:~$ █
```

```
hive> load data local inpath '/tmp/empleados1.txt' into table empleados;
Loading data to table ejemplo.empleados
OK
Time taken: 0.126 seconds
hive> select * from empleados;
OK
Rosa      50
Pedro     60
Raul      56
Maria     35
Rosa      50
Pedro     60
Raul      56
Maria     35
Time taken: 0.082 seconds, Fetched: 8 row(s)
hive> █
```

8. TABLAS INTERNAS Y LOAD

Paso 11. Hacemos un refresh en Browse HDFS y vemos que me carga otro fichero

Browse Directory

/user/hive/warehouse/ejemplo.db/empleados

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	34 B	Mar 19 10:07	2	128 MB	empleados.txt
-rw-r--r--	hadoop	supergroup	34 B	Mar 19 10:31	2	128 MB	empleados1.txt

Showing 1 to 2 of 2 entries

Previous 1 Next

Hadoop, 2022.

Una tabla de Hive está asociada a ficheros dentro de un directorio. Dentro de la base de datos empleados podremos tener muchos ficheros. Mientras se tenga la misma estructura. no va a tener problemas. Hive lee todo el contenido de este directorio HDFS y lo une para tener una sola tabla

8. TABLAS INTERNAS Y LOAD

Paso 12. Podemos hacer una query SQL con where edad>50.

```
hive> select * from empleados where edad>50;
OK
Pedro    60
Raul     56
Pedro    60
Raul     56
Time taken: 0.116 seconds, Fetched: 4 row(s)
hive> █
```

Vemos que HiveSQL es muy parecido al SQL normal.

Paso 13. Hemos dicho que las tablas internas las gestiona Hive. Si hacemos **drop table empleados;** no sólo me va a borrar la tabla a nivel de Hive, a nivel de la estructura de metadatos, sino que también la borra en HDFS. En este caso Hive ha gestionado todo el ciclo de vida de esa tabla

```
hive> drop table empleados;
OK
Time taken: 0.218 seconds
hive> █
```

Browse Directory								
<input type="text" value="/user/hive/warehouse/ejemplo.db"/> Go! 								
Show <input type="button" value="25"/> entries <input type="text" value="Search:"/>								
□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
□	drwxr-xr-x	hadoop	supergroup	0 B	Mar 18 18:07	0	0 B	t1
□	drwxr-xr-x	hadoop	supergroup	0 B	Mar 18 19:47	0	0 B	t2

Showing 1 to 2 of 2 entries

Hadoop, 2022.

9. TABLAS EXTERNAS

Paso 1. Vamos a ver ahora las tablas externas de Hive que son muy parecidas realmente a las tablas internas con la principal diferencia de que su ciclo de vida no lo gestiona el propio Hive.

Cargaremos ahora el fichero empleados desde HDFS. Creamos el directorio /externa y subimos el fichero empleados.txt

```

hadoop@nodo1:~$ hdfs dfs -ls /
Found 4 items
drwxr-xr-x  - hadoop supergroup          0 2023-03-12 08:39 /practicas
drwxr-xr-x  - hadoop supergroup          0 2023-03-09 16:04 /prueba
drwxrwxr-x  - hadoop supergroup          0 2023-03-18 19:46 /tmp
drwxr-xr-x  - hadoop supergroup          0 2023-03-17 22:45 /user
hadoop@nodo1:~$ hdfs dfs -mkdir /externa
hadoop@nodo1:~$ hdfs dfs -put empleados.txt /externa
hadoop@nodo1:~$ █

```

Browse Directory

/externa	Go!						
Show 25 entries	Search:						
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	
-rw-r--r--	hadoop	supergroup	34 B	Nov 04 08:31	2	128 MB	
Showing 1 to 1 of 1 entries							
Previous	1	Next					

9. TABLAS EXTERNAS

Paso 2. Creamos la misma tabla pero ahora con la clausula external:

```
create external table empleados (nombre string, edad integer)
row format delimited
fields terminated by ','
location '/user/hive/datos/empleados';
```

```
hive> create external table empleados (nombre string, edad integer)
      > row format delimited
      > fields terminated by ','
      > location '/user/hive/datos/empleados';
OK
Time taken: 0.034 seconds
hive> █
```

NOTA: La diferencia respecto el ejemplo de la tabla interna, es el uso de las cláusulas **external** y **location**, donde se encuentran los datos de la base de datos una vez cargados

9. TABLAS EXTERNAS

Paso 3. Si vamos a Browse HDFS en /user/hive vemos que ha creado un directorio llamado datos y otro llamado empleados, que es donde vamos a guardar la información

Browse Directory

/user/hive

Show 25 entries

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Mar 19 11:50	0	0 B	datos 
<input type="checkbox"/>	drwxrwxr-x	hadoop	supergroup	0 B	Mar 18 19:05	0	0 B	warehouse 

Showing 1 to 2 of 2 entries

Previous 1 Next

Hadoop, 2022.

Browse Directory

/user/hive/datos

Show 25 entries

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Mar 19 11:50	0	0 B	empleados 

Showing 1 to 1 of 1 entries

Hadoop, 2022.

9. TABLAS EXTERNAS

Paso 4. Ahora vamos a cargar los datos a la tabla empleados desde el directorio HDFS /externa/empleados.txt (no usamos la clausula local):

load data inpath '/externa/empleados.txt' into table empleados;

Vemos que un select en la tabla empleados funciona correctamente. Y los datos están cargados en el directorio HDFS /user/hive/datos/empleados

```
hive> load data inpath '/externa/empleados.txt' into table empleados;
Loading data to table ejemplo.empleados
OK
Time taken: 0.113 seconds
hive> select * from empleados;
OK
Rosa      50
Pedro     60
Raul      56
Maria     35
Time taken: 0.109 seconds, Fetched: 4 row
hive>
```

Browse Directory

/user/hive/datos/empleados

Show 25 entries

Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	34 B	Mar 19 11:33	2	128 MB	empleados.txt

Showing 1 to 1 of 1 entries

Previous 1 Next

Hadoop, 2022.

9. TABLAS EXTERNAS

Paso 5. Si hacemos un **drop table empleados**, la table desaparece de Hive. Lo vemos mediante **show tables**:

```
hive> drop table empleados;
OK
Time taken: 0.047 seconds
hive> show tables;
OK
t1
t2
Time taken: 0.021 seconds, Fetched: 2 row(s)
hive>
```

Paso 6. Pero los datos originales no desaparecen al tratarse de una tabla externa. En /user/hive/datos/empleados seguimos teniendo el fichero empleados.txt. No ha desaparecido como pasaba con las tablas internas.

Browse Directory

/user/hive/datos/empleados

Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	34 B	Mar 19 11:33	2	128 MB	empleados.txt

Showing 1 to 1 of 1 entries

Previous 1 Next

Hadoop, 2022.

10. PRACTICA TABLAS HIVE

Tablas Internas

Paso 1. Comprobar si hay bases de datos

```
hive> show databases;  
OK  
default  
ejemplo  
Time taken: 0.017 seconds, Fetched: 2 row(s)  
hive>
```

Paso 2. Nos conectamos a la Base de Datos de ejemplo

```
hive> use ejemplo;  
OK  
Time taken: 0.015 seconds  
hive>
```

10. PRACTICA TABLAS HIVE

Paso 3. Crea la siguiente tabla interna:

```
CREATE TABLE IF NOT EXISTS empleados_internal (
    name string,
    work_place array<string>,
    sex_age struct<sex:string,age:int>,
    skills_score map<string,int>,
    depart_title map<string,array<string>>
)
```

comment 'Esto es una tabla interna'

row format delimited

fields terminated by '|'

collection items terminated by ','

map keys terminated by ':';

```
hive> create table if not exists empleados_internal (
>     name string,
>     work_place array<string>,
>     sex_age struct<sex:string,age:int>,
>     skills_score map<string,int>,
>     depart_title map<string,array<string>>
> )
> comment "ESTo es una tabla interna"
> row format delimited
> fields terminated by '|'
> collection items terminated by ','
> map keys terminated by ':';
OK
Time taken: 0.051 seconds
hive> █
```

10. PRACTICA TABLAS HIVE

Paso 4. Lo cargamos con los datos del fichero empleados.txt que está en los recursos del drive:

```
load data local inpath '/home/hadoop/Downloads/empleados.txt'
overwrite into table empleados_internal;
```

Probamos de hacer un select a la tabla empleados_internal:

```
hive> load data local inpath '/home/hadoop/Downloads/empleados.txt' overwrite into table empleados_internal;
Loading data to table ejemplo.empleados_internal
OK
Time taken: 0.112 seconds
hive> select * from empleados_internal;
OK
Michael ["Montreal","Toronto"] {"sex":"Male","age":30} {"DB":80} {"Product":["Developer","Lead"]}
Will ["Montreal"] {"sex":"Male","age":35} {"Perl":85} {"Product":["Lead"],"Test":["Lead"]}
Shelley ["New York"] {"sex":"Female","age":27} {"Python":80} {"Test":["Lead"],"COE":["Architect"]}
Lucy ["Vancouver"] {"sex":"Female","age":57} {"Sales":89,"HR":94} {"Sales":["Lead"]}
Time taken: 0.093 seconds, Fetched: 4 row(s)
hive>
```

GNU nano 6.2	empleados.txt
Michael Montreal,Toronto Male,30 DB:80 Product:Developer^DLead	
Will Montreal Male,35 Perl:85 Product:Lead,Test:Lead	
Shelley New York Female,27 Python:80 Test:Lead,COE:Architect	
Lucy Vancouver Female,57 Sales:89,HR:94 Sales:Lead	

10. PRACTICA TABLAS HIVE

Paso 5. Comprueba que existe en el directorio warehouse de HIVE, dentro de la base de datos ejemplo. También se puede ver con HDFS

hdfs dfs -ls /user/hive/warehouse/ejemplo.db

```

hadoop@nodo1:~/Downloads$ hdfs dfs -ls /user/hive/warehouse/ejemplo.db
Found 4 items
drwxr-xr-x  - hadoop supergroup          0 2023-03-19 11:46 /user/hive/warehouse/ejemplo.db/empleados
drwxr-xr-x  - hadoop supergroup          0 2023-03-19 15:20 /user/hive/warehouse/ejemplo.db/empleados_internal
drwxr-xr-x  - hadoop supergroup          0 2023-03-18 18:07 /user/hive/warehouse/ejemplo.db/t1
drwxr-xr-x  - hadoop supergroup          0 2023-03-18 19:47 /user/hive/warehouse/ejemplo.db/t2
hadoop@nodo1:~/Downloads$ 

```

Browse Directory

/user/hive/warehouse/ejemplo.db								
Show 25 entries								
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
drwxr-xr-x	hadoop	supergroup	0 B	Mar 19 11:46	0	0 B	empleados	
drwxr-xr-x	hadoop	supergroup	0 B	Mar 19 15:20	0	0 B	empleados_internal	
drwxr-xr-x	hadoop	supergroup	0 B	Mar 18 18:07	0	0 B	t1	
drwxr-xr-x	hadoop	supergroup	0 B	Mar 18 19:47	0	0 B	t2	

Showing 1 to 4 of 4 entries

Hadoop, 2022.

Browse Directory

/user/hive/warehouse/ejemplo.db/empleados_internal								
Show 25 entries								
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
-rw-r--r--	hadoop	supergroup	227 B	Mar 19 15:20	2	128 MB	empleados.txt	

Showing 1 to 1 of 1 entries

Hadoop, 2022.

10. PRACTICA TABLAS HIVE

Tablas externas

Paso 6. Creamos ahora una tabla externa.

```
create external table if not exists empleados_external (
    name string,
    work_place array<string>,
    sex_age struct<sex:string,age:int>,
    skills_score map<string,int>,
    depart_title map<string,array<string>>
)
```

comment 'This is an external table'

row format delimited

fields terminated by '|'

collection items terminated by ','

map keys terminated by ':'

location '/ejemplo/empleados';

```
hive> create external table if not exists empleados_external (
>     name string,
>     work_place array<string>,
>     sex_age struct<sex:string,age:int>,
>     skills_score map<string,int>,
>     depart_title map<string,array<string>>
> )
> comment "Esto es una tabla externa"
> row format delimited
> fields terminated by '|'
> collection items terminated by ','
> map keys terminated by ':'
> location '/ejemplo/empleados';
OK
Time taken: 0.046 seconds
hive> █
```

10. PRACTICA TABLAS HIVE

Tablas externas

Paso 7. Comprobamos que ya se ha creado el directorio HDFS /ejemplo, donde se van a guardar los datos.

```
hadoop@nodo1:~/Downloads$ hdfs dfs -ls /ejemplo
Found 1 items
drwxr-xr-x - hadoop supergroup          0 2023-03-19 16:11 /ejemplo/empleados
hadoop@nodo1:~/Downloads$ hdfs dfs -ls /ejemplo/empleados
hadoop@nodo1:~/Downloads$
```

Paso 8. Lo cargamos con los mismos datos

**load data local inpath '/home/hadoop/Downloads/empleados.txt'
overwrite into table empleados_external;**

```
hive> load data local inpath '/home/hadoop/Downloads/empleados.txt' overwrite into table empleados_external;
Loading data to table ejemplo.empleados_external
OK
Time taken: 0.146 seconds
hive> select * from empleados_external;
OK
Michael ["Montreal", "Toronto"] {"sex": "Male", "age": 30} {"DB": 80} {"Product": ["Developer", "Lead"]}
Will ["Montreal"] {"sex": "Male", "age": 35} {"Perl": 85} {"Product": ["Lead"], "Test": ["Lead"]}
Shelley ["New York"] {"sex": "Female", "age": 27} {"Python": 80} {"Test": ["Lead"], "COE": ["Architect"]}
Lucy ["Vancouver"] {"sex": "Female", "age": 57} {"Sales": 89, "HR": 94} {"Sales": ["Lead"]}
Time taken: 0.087 seconds, Fetched: 4 row(s)
hive>
```

10. PRACTICA TABLAS HIVE

Paso 9. Comprobar que existe el directorio de datos

Browse Directory

/ejemplo/empleados

Show 25 entries

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	227 B	Mar 19 16:28	2	128 MB	empleados.txt

Showing 1 to 1 of 1 entries

Previous 1 Next

Hadoop, 2022.

Paso 10. Hacer alguna SELECT por ejemplo para buscar al empleado "Lucy"

```
hive> select * from empleados_external where name='Lucy';
OK
Lucy      ["Vancouver"]      {"sex":"Female", "age":57}          {"Sales":89, "HR":94}      {"Sales":["Lead"]}
Time taken: 0.091 seconds, Fetched: 1 row(s)
hive>
```

10. PRACTICA TABLAS HIVE

Paso 11. Borrar la dos tablas

```
hive> drop table empleados_external;
OK
Time taken: 0.069 seconds
hive> drop table empleados_internal;
OK
Time taken: 0.015 seconds
hive> ■
```

Paso 12. Comprobar que ha borrado la interna, pero los datos de la externa permanecen.

Browse Directory

Browse Directory								
<input type="text" value="/ejemplo/empleados"/> Go!								
Show <input type="button" value="25"/> entries	<input type="text" value="Search:"/>							
<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	227 B	Mar 19 16:28	2	128 MB	empleados.txt

Showing 1 to 1 of 1 entries

Hadoop, 2022.

Browse Directory

Browse Directory								
<input type="text" value="/user/hive/warehouse/ejemplo.db"/> Go!								
Show <input type="button" value="25"/> entries	<input type="text" value="Search:"/>							
<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Mar 19 11:46	0	0 B	empleados
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Mar 18 18:07	0	0 B	t1
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Mar 18 19:47	0	0 B	t2

Showing 1 to 3 of 3 entries

Hadoop, 2022.

11. HIVE DE FORMA REMOTA

Paso 1. Vamos a ver cómo configurar hive para que pueda ser accedido de forma remota. El cliente hive permite trabajar en local. Para trabajar de forma remota, se utiliza el servidor hiveserver2. Permite conexiones remotas contra el metaStore de Hive, el cual almacena toda la información de metadatos de Hive.

Desde el directorio bbdd donde tenemos los metadatos ejecutamos:

hiveserver2 & → Se ejecuta en modo background. De esta manera tenemos libre la pantalla principal para trabajar. Si todo es correcto saldrá el id del proceso con el que se lanza y arrancará el servidor.

```
hadoop@nodo1:/opt/hadoop/hive/bbdd$ hiveserver2 &
[1] 1535237
hadoop@nodo1:/opt/hadoop/hive/bbdd$ 2023-03-19 18:09:38: Starting HiveServer2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hive/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.35.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 3b43db29-7d66-4529-af33-89fb3909cddb
Hive Session ID = 1d7a4e9c-b39f-4d74-82ef-3564b1fdfd6e

hadoop@nodo1:/opt/hadoop/hive/bbdd$
```

11. HIVE DE FORMA REMOTA

Paso 2. Si ejecutamos **ps -ef | grep hiveserver**, vemos que el hiveserver ha arrancado como un proceso java

```

hadoop@nodo1:/opt/hadoop/hive/bbdd$ ps -ef | grep hiveserver
hadoop 1535237 1797 2 18:09 pts/1 00:00:09 /usr/lib/jvm/java-8-openjdk-amd64/bin/java -Dproc_jar -Dproc_hiveserver2 -Dlog4j2.formatMsgNoLookups=true
-Dlog4j.configurationFile=hive-log4j2.properties -Djava.util.logging.config.file=/opt/hadoop/hive/conf/parquet-logging.properties -Djline.terminal=jline.UnsupportedTerminal -Dyarn.log.dir=/opt/hadoop/logs -Dyarn.log.file=hadoop.log -Dyarn.home.dir=/opt/hadoop -Dyarn.root.logger=INFO,console -Djava.library.path=/opt/hadoop/lib/native -Xmx256m -Dhadoop.log.dir=/opt/hadoop/logs -Dhadoop.log.file=hadoop.log -Dhadoop.home.dir=/opt/hadoop -Dhadoop.id.str=hadoop -Dhadoop.root.logger=INFO,console -Dhadoop.policy.file=hadoop-policy.xml -Dhadoop.security.logger=INFO,NullAppender org.apache.hadoop.util.RunJar /opt/hadoop/hive/lib/hive-service-3.1.3.jar org.apache.hive.service.server.HiveServer2
hadoop 1543907 1797 0 18:15 pts/1 00:00:00 grep --color=auto hiveserver
hadoop@nodo1:/opt/hadoop/hive/bbdd$ █

```

Paso 3. Para conectarse con el servidor hiveserver podemos utilizar distintos clientes, a través de JDBC (driver por defecto) o HTTP.

Usaremos el cliente beeline, herramienta parecida al Hive, pero que nos permite conectarnos contra el servidor hiveserver. Después de una serie de mensajes, ofrece un entorno de línea de comandos como el cliente hive:

```

hadoop@nodo1:/opt/hadoop/hive/bbdd$ beeline
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hive/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.35.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Beeline version 3.1.3 by Apache Hive
beeline>

```

11. HIVE DE FORMA REMOTA

Paso 4. Primero ejecutamos el siguiente comando para conectarnos:

beeline> !connect jdbc:hive2://nodo1:10000

La url de conexión es parecida a la de otras bases de datos, como por ejemplo mysql → jdbc:mysql://localhost:3306/bd

driver_jdbc:tipo://nombre_maquina:puerto

En este caso ya no tenemos que estar en el directorio bbdd

Siempre escucha por el puerto 10000 aunque también se puede configurar en el hive-site.xml

La conexión va a pedir usuario y contraseña. Por defecto podemos poner vacío o utilizar el usuario con el que hacemos la ejecución

```
beeline> !connect jdbc:hive2://nodo1:10000
Connecting to jdbc:hive2://nodo1:10000
Enter username for jdbc:hive2://nodo1:10000:
Enter password for jdbc:hive2://nodo1:10000:
23/03/19 20:48:54 [main]: WARN jdbc.HiveConnection: Failed to connect to nodo1:10000
Error: Could not open client transport with JDBC Uri: jdbc:hive2://nodo1:10000: Failed to open new session: java.lang.RuntimeException:
RemoteException(org.apache.hadoop.security.authorize.AuthorizationException): User: hadoop is not allowed to impersonate anonymous (sta
beeline>
```

11. HIVE DE FORMA REMOTA

Paso 5. Tenemos que configurar un parámetro muy importante en `hive-site.xml`. Poniendo `hive.server2.enable.doAs` a false podemos conectarnos al hiveserver directamente sin necesidad de tener que pedir credenciales, usando el usuario que lo ha ejecutado (hadoop)

`sudo nano /opt/hadoop/hive/conf/hive-site.xml`

```
GNU nano 6.2          /opt/hadoop/hive/conf/hive-site.xml
<property>
  <name>hive.server2.enable.doAs</name>
  <value>false</value>
  <description>
    Setting this property to true will have HiveServer2 execute
    Hive operations as the user making the calls to it.
  </description>
</property>
<property>
  <name>hive.distcp.privileged.doAs</name>
  <value>false</value>
  <description>
    This property allows privileged distcp executions done by hive
    to run as this user.
  </description>
</property>
```

NOTA: Si hay definido un usuario para hive, se puede entrar con el. Se pueden poner credenciales más seguras a Hive, incluso trabajar con Kerberos y productos similares.

11. HIVE DE FORMA REMOTA

Paso 6. Refrescamos el hiveserver:

a) Salimos de beeline

beeline> !quit

b) Matamos hiveserver

kill -9 <process>

c) Ejecutamos hiveserver

hiveserver2 &

d) Entramos en beeline

beeline

e) Nos conectamos

!connect jdbc:hive2://nodo1:10000

f) Hacemos enter+enter

```

beeline> !quit
hadoop@nodo1:/opt/hadoop/hive/bbdd$ kill -9^C
130 hadoop@nodo1:/opt/hadoop/hive/bbdd$ kill -9 1535237
hadoop@nodo1:/opt/hadoop/hive/bbdd$ hiveserver2 &
[2] 1845194
[1] Killed hiveserver2
hadoop@nodo1:/opt/hadoop/hive/bbdd$ 2023-03-20 05:16:07: Starting HiveServer2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hive/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.35.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 38c46560-1141-4434-b6b2-e4e016f20f0d
Hive Session ID = 093eceba-4096-4cef-887f-e1bb83dedc93

hadoop@nodo1:/opt/hadoop/hive/bbdd$ beeline
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hive/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.35.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Beeline version 3.1.3 by Apache Hive
beeline> !connect jdbc:hive2://nodo1:10000
Connecting to jdbc:hive2://nodo1:10000
Enter username for jdbc:hive2://nodo1:10000:
Enter password for jdbc:hive2://nodo1:10000:
Connected to: Apache Hive (version 3.1.3)
Driver: Hive JDBC (version 3.1.3)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://nodo1:10000> ■

```

11. HIVE DE FORMA REMOTA

Paso 7. Los comandos desde beeline funcionan igual que desde hive.
show databases; → muestra todas las bases de datos que hemos ido creando dentro de mi servidor

use ejemplo;

show tables;

```
f-7fa52840adec): show databases
INFO : Starting task [Stage-0:DD
INFO : Completed executing command
f-4111-b77f-7fa52840adec); Time t
INFO : OK
INFO : Concurrency mode is disabled
+-----+
| database_name |
+-----+
| default       |
| ejemplo        |
+-----+
2 rows selected (0,782 seconds)
0: jdbc:hive2://nodo1:10000> ■
```

```
0: jdbc:hive2://nodo1:10000> show tables;
OK
INFO : Compiling command(queryId=hadoop_20
d-0cb6a6ea0756): show tables
INFO : Concurrency mode is disabled, not creatin
INFO : Semantic Analysis Completed (retrial = fa
INFO : Returning Hive schema: Schema(fieldSchem
type:string, comment:from deserializer]),
INFO : Completed compiling command(queryId=hadoo
6-4838-902d-0cb6a6ea0756); Time taken: 0.02
INFO : Concurrency mode is disabled, not creatin
INFO : Executing command(queryId=hadoop_20
d-0cb6a6ea0756): show tables
INFO : Starting task [Stage-0:DDL] in seri
INFO : Completed executing command(queryId=had
6-4838-902d-0cb6a6ea0756); Time taken: 0.01
INFO : OK
INFO : Concurrency mode is disabled, not creatin
+-----+
| tab_name      |
+-----+
| t1            |
| t2            |
+-----+
2 rows selected (0,052 seconds)
0: jdbc:hive2://nodo1:10000> ■
```

```
0: jdbc:hive2://nodo1:10000> select * from t2;
OK
INFO : Compiling command(queryId=hadoop_20230320
9-556b708b81f3): select * from t2
INFO : Concurrency mode is disabled, not creatin
INFO : Semantic Analysis Completed (retrial = fa
INFO : Returning Hive schema: Schema(fieldSchema
, type:int, comment:null)), properties:null
INFO : Completed compiling command(queryId=hadoo
1-4aa2-b969-556b708b81f3); Time taken: 0.143 sec
INFO : Concurrency mode is disabled, not creatin
INFO : Executing command(queryId=hadoop_20230320
9-556b708b81f3): select * from t2
INFO : Completed executing command(queryId=hadoo
1-4aa2-b969-556b708b81f3); Time taken: 0.001 sec
INFO : OK
INFO : Concurrency mode is disabled, not creatin
+-----+
| t2.codigo    |
+-----+
| 10          |
+-----+
1 row selected (0,302 seconds)
0: jdbc:hive2://nodo1:10000>
```

11. HIVE DE FORMA REMOTA

Paso 8. Desde beeline podemos utilizar comandos de tipo SQL HiveSQL y comandos propios de beeline, los cuales vienen precedidos con una especie de signo de admiración !

```
0: jdbc:hive2://nodo1:10000> help
                                client side.
!batch                      Start or execute a batch of statements
!close                       Close the current connection to the database
!commit                      Commit the current transaction (if autocommit is off)
!connect                     Open a new connection to the database.
!dbinfo                      Give metadata information about the database
!delimiter                   Sets the query delimiter, defaults to ;
!describe                    Describe a table
!dropall                     Drop all tables in the current database
!exportedkeys                List all the exported keys for the specified table
!go                          Select the current connection
!help                        Print a summary of command usage
!history                     Display the command history
!importedkeys                List all the imported keys for the specified table
!indexes                     List all the indexes for the specified table
!isolation                   Set the transaction isolation for this connection
!list                        List the current connections
!manual                      Display the BeeLine manual
!metadata                    Obtain metadata information
!nativesql                   Show the native SQL for the specified statement
!nullemptystring             Set to true to get historic behavior of printing null as
```

11. HIVE DE FORMA REMOTA

Paso 9. Podemos ejecutar show tables o !tables para ver las tablas del esquema

```
0: jdbc:hive2://nodo1:10000> !tables
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| TABLE_CAT | TABLE_SCHEMA | TABLE_NAME | TABLE_TYPE | REMARKS | TYPE_CAT | TYPE_SCHEMA | TYPE_NAME | SELF_REFERENCING_COL_NAME | REF_GENERATION |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ejemplo   | t1          | TABLE      | NULL       | NULL     | NULL     | NULL     | NULL     | NULL           | NULL
| default   | t2          | TABLE      | NULL       | NULL     | NULL     | NULL     | NULL     | NULL           | NULL
| ejemplo   | t2          | TABLE      | NULL       | NULL     | NULL     | NULL     | NULL     | NULL           | NULL
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
0: jdbc:hive2://nodo1:10000> ■
```

```
0: jdbc:hive2://nodo1:10000> show tables;
OK
INFO : Compiling command(queryId=hadoop_20230320060621_6774c2b9-3cdc-4d96-85cb-6610d0c826c4): show tables
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Semantic Analysis Completed (retrial = false)
INFO : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:tab_name, type:string, comment:from deserializer)], properties:nul
INFO : Completed compiling command(queryId=hadoop_20230320060621_6774c2b9-3cdc-4d96-85cb-6610d0c826c4); Time taken: 0.019 seconds
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Executing command(queryId=hadoop_20230320060621_6774c2b9-3cdc-4d96-85cb-6610d0c826c4): show tables
INFO : Starting task [Stage-0:DDL] in serial mode
INFO : Completed executing command(queryId=hadoop_20230320060621_6774c2b9-3cdc-4d96-85cb-6610d0c826c4); Time taken: 0.012 seconds
INFO : OK
INFO : Concurrency mode is disabled, not creating a lock manager
+-----+
| tab_name |
+-----+
| t1      |
| t2      |
+-----+
2 rows selected (0.067 seconds)
0: jdbc:hive2://nodo1:10000> ■
```

11. HIVE DE FORMA REMOTA

Paso 10. Ahora nos conectaremos en remoto desde otro nodo, por ejemplo el nodo3. Lo primero que tenemos que hacer es salir de beeline en el nodo1, mediante !close, !quit

```
0: jdbc:hive2://nodo1:10000> !close
Closing: 0: jdbc:hive2://nodo1:10000
beeline> !quit
hadoop@nodo1:/opt/hadoop/hive/bbdd$
```

Paso 11. Debemos copiar toda la configuración de hive y el fichero .bashrc al nodo3

scp -r /opt/hadoop/hive nodo3://opt/hadoop

scp /home/hadoop/.bashrc nodo3://home/hadoop

```
hadoop@nodo1:/opt/hadoop/hive/bbdd$ scp -r /opt/hadoop/hive nodo3:/opt/hadoop
LICENSE                                         100%   20KB  4.2MB/s  00:00
hive-jdbc-3.1.3-standalone.jar                  100%   69MB 129.1MB/s  00:00
hcat-config.sh                                    100% 2595    2.8MB/s  00:00
hcatcfg.py                                       100% 3400    5.0MB/s  00:00
webhcat_config.sh                                100% 4310    6.1MB/s  00:00
webhcat_server.sh                               100% 7251   10.0MB/s  00:00
hcat_server.sh                                   100% 4569    7.8MB/s  00:00
update-hcatalog-env.sh                           100%  10KB 13.2MB/s  00:00
hcat_server.py                                    100% 5450    7.5MB/s  00:00
hive-hcatalog-core-3.1.3.jar                   100% 264KB 92.9MB/s  00:00
hive-hcatalog-server-extensions-3.1.3.jar      100%  76KB 54.8MB/s  00:00
hive-hcatalog-pig-adapter-3.1.3.jar            100%  56KB 29.7MB/s  00:00
hive-hcatalog-streaming-3.1.3.jar              100% 132KB 54.7MB/s  00:00
README.txt                                       100% 1703     1.5MB/s  00:00
hive-webhcat-java-client-3.1.3.jar             100% 113KB 46.3MB/s  00:00
xml-apis-1.3.04.jar                            100% 190KB 57.7MB/s  00:00
```

```
hadoop@nodo1:/opt/hadoop/hive/bbdd$ scp /home/hadoop/.bashrc nodo3:/home/hadoop
.bashrc                                         100% 4054    1.6MB/s  00:00
hadoop@nodo1:/opt/hadoop/hive/bbdd$
```

11. HIVE DE FORMA REMOTA

Paso 12. En otra pestaña hacemos ssh nodo3 e intentaremos conectarnos al hiveserver del nodo1.

Vamos al directorio **/opt/hadoop/hive/bin**, ejecutamos beeline y nos conectamos mediante jdbc al hiveserver del nodo1

```
hadoop@nodo3:/opt/hadoop/hive/bin$ ./beeline
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hive/lib/log4j-slf4j-impl-2.17.1.ja
4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-relo
.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Beeline version 3.1.3 by Apache Hive
beeline> !connect jdbc:hive2://nodo1:10000
Connecting to jdbc:hive2://nodo1:10000
Enter username for jdbc:hive2://nodo1:10000:
Enter password for jdbc:hive2://nodo1:10000:
Connected to: Apache Hive (version 3.1.3)
Driver: Hive JDBC (version 3.1.3)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://nodo1:10000>
```

11. HIVE DE FORMA REMOTA

Paso 13. Hacemos show databases; pero en este caso desde el nodo3 conectado contra el nodo1

```
hadoop@nodo1 (192.168.0.101) - byobu x      hadoop@nodo3: /opt/hadoop/hive/bin x

Driver: Hive JDBC (version 3.1.3)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://nodo1:10000> show databases;
INFO  : Compiling command(queryId=hadoop_20230320071206_2ae39d32-124c-4c47-a2ab-2af): show databases
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retryal = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:database_name
ring, comment:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=hadoop_20230320071206_2ae39d32-124c-4ca4674c2e15f); Time taken: 0.013 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hadoop_20230320071206_2ae39d32-124c-4c47-a2ab-2af): show databases
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hadoop_20230320071206_2ae39d32-124c-4ca4674c2e15f); Time taken: 0.022 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
+-----+
| database_name |
+-----+
| default      |
| ejemplo      |
+-----+
2 rows selected (0,115 seconds)
```

12. EJEMPLO REAL

Paso 1. Este ejemplo, lo realizaremos físicamente desde el nodo1. Nos conectaremos mediante ssh al nodo3 y desde aquí estableceremos una conexión remota con el hiveserver2 del nodo1. En nodo3 recargamos el fichero .bashrc para poder acceder a las nuevas variables de entorno de hive y ejecutamos beeline:

```

hadoop@nodo3:~$ pwd
/home/hadoop
hadoop@nodo3:~$ source .bashrc
hadoop@nodo3:~$ beeline
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hive/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-reloadable.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Beeline version 3.1.3 by Apache Hive

```

Paso 2. En la conexión contra hiveserver le vamos a indicar contra qué base de datos vamos a trabajar. Así nos ahorraremos hacer el use

```

beeline> !connect jdbc:hive2://nodo1:10000/ejemplo
Connecting to jdbc:hive2://nodo1:10000/ejemplo
Enter username for jdbc:hive2://nodo1:10000/ejemplo:
Enter password for jdbc:hive2://nodo1:10000/ejemplo:
Connected to: Apache Hive (version 3.1.3)
Driver: Hive JDBC (version 3.1.3)
Transaction isolation: TRANSACTION_REPEATABLE_READ

```

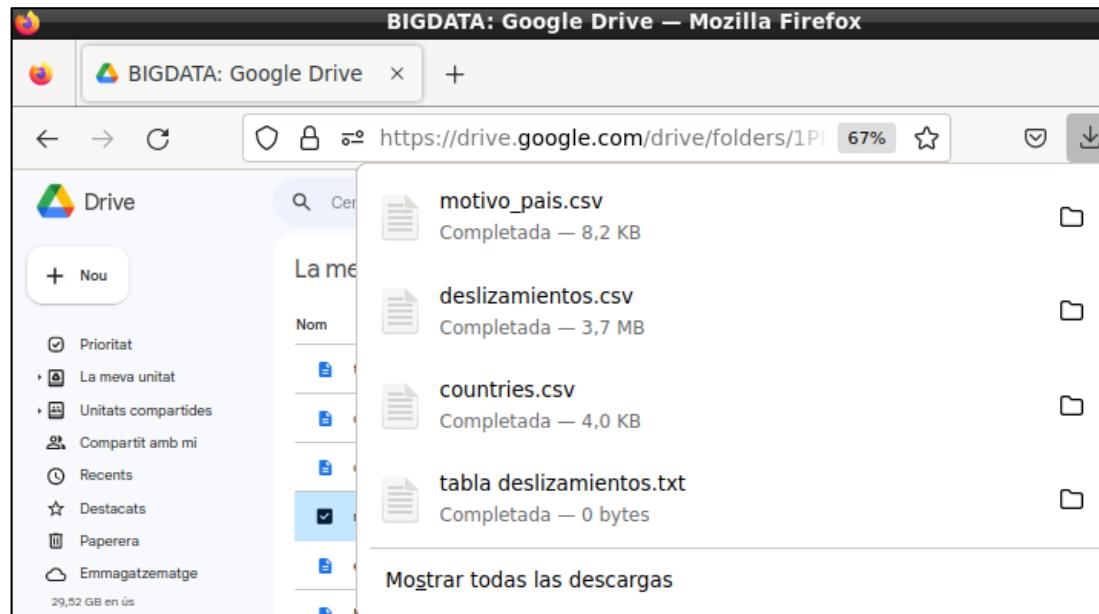
```

INFO : Concurrency mode is disabled, n
+-----+
| tab_name |
+-----+
| t1       |
| t2       |
+-----+
2 rows selected (0,075 seconds)
0: jdbc:hive2://nodo1:10000/ejemplo>

```

12. EJEMPLO REAL

Paso 3. Trabajaremos con un ejemplo de un DataSet descargado desde la página web de la NASA. Desde la web data.nasa.gov podemos descargar múltiples datos de todos tipos. Probaremos uno que es de los deslizamientos de tierras que han ocurrido en los últimos años en todo. No tiene muchas filas (unas 10000), pero podríamos tener millones. Descargamos los documentos necesarios de la carpeta BIGDATA compartida en la maquina nodo1.



12. EJEMPLO REAL

Paso 4. Primero crearemos la tabla de deslizamientos. Desde nodo1, vamos a Downloads y con gedit abrimos tabla_deslizamientos.txt. Copiamos su contenido y la pegamos en beeline del nodo3. Comprobamos que se ha creado la tabla:

```
nodo3 [S'està executant] - Oracle VM VirtualBox
Fitxer Mànquina Visualitza Entrada Dispositius Ajuda
Open + tabla_deslizamientos.txt
/home/hadoop/Downloads

1 create table deslizamientos
2 (
3 id bigint ,
4 fecha string ,
5 hora string ,
6 country string ,
7 nearest_places string ,
8 hazard_type string ,
9 landslide_type string ,
10 motivo string ,
11 storm_name string ,
12 fatalities bigint ,
13 injuries string ,
14 source_name string ,
15 source_link string ,
16 location_description string ,
17 location_accuracy string ,
18 landslide_size string ,
19 photos_link string ,
20 cat_src string ,
21 cat_id bigint ,
22 countryname string ,
23 near_string
```

```
location accuracy string ,
near_string ,
distance double ,
adminname1 string ,
adminname2 string ,
population bigint ,
countrycode string ,
continentcode string ,
key string ,
version string ,
tstamp string ,
changeset_id string ,
latitude double ,
longitude double ,
geolocation string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ';'
;
INFO : Compiling command(queryId=hadoop_20230320084556_fc034ec9-2b703090c825): create table deslizamientos
```

tab_name
deslizamientos
t1
t2

3 rows selected (0,029 seconds)
0: jdbc:hive2://nodo1:10000/ejemplo>

12. EJEMPLO REAL

Paso 5. Ejecutamos **desc deslizamientos**, que tiene una serie de columnas: id, fecha deslizamiento, la hora, el país, el tipo, cuántos víctimas, si es de una tormenta, etc.

source_link	string
location_description	string
location_accuracy	string
landslide_size	string
photos_link	string
cat_src	string
cat_id	bigint
countryname	string
near	string
distance	double
adminname1	string
adminname2	string
population	bigint
countrycode	string
continentcode	string
key	string
version	string
tstamp	string
changeset_id	string
latitude	double
longitude	double
geolocation	string

```
+-----+-----+-----+
34 rows selected (0,046 seconds)
0: jdbc:hive2://nodo1:10000/ejemplo> █
```

12. EJEMPLO REAL

Paso 6. Ahora cargaremos los datos con el fichero deslizamientos.csv

En nodo1 → cp /home/hadoop/Downloads/deslizamientos.csv /tmp

En nodo3 → load data local inpath '/tmp/deslizamientos.csv' into table deslizamientos;

Nota: deslizamientos.csv debe de estar en el sistema de ficheros del nodo1 (/tmp) donde se ejecuta el hiveserver, no en nodo3.

```
0: jdbc:hive2://nodo1:10000/ejemplo> load data local inpath '/tmp/deslizamientos.csv' into table deslizamientos;
INFO : Compiling command(queryId=hadoop_20230320094650_6c813dc3-a285-4537-a0df-34e4637b37b9): load data local inpath '/tmp/deslizamientos.csv' into table deslizamientos
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Semantic Analysis Completed (retrial = false)
INFO : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO : Completed compiling command(queryId=hadoop_20230320094650_6c813dc3-a285-4537-a0df-34e4637b37b9); Time taken: 0.032 seconds
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Executing command(queryId=hadoop_20230320094650_6c813dc3-a285-4537-a0df-34e4637b37b9): load data local inpath '/tmp/deslizamientos.csv' into table deslizamientos
INFO : Starting task [Stage-0:MOVE] in serial mode
INFO : Loading data to table ejemplo.deslizamientos from file:/tmp/deslizamientos.csv
INFO : Starting task [Stage-1:STATS] in serial mode
INFO : Completed executing command(queryId=hadoop_20230320094650_6c813dc3-a285-4537-a0df-34e4637b37b9); Time taken: 0.194 seconds
INFO : OK
INFO : Concurrency mode is disabled, not creating a lock manager
No rows affected (0,279 seconds)
0: jdbc:hive2://nodo1:10000/ejemplo> █
```

12. EJEMPLO REAL

Paso 7. Ya podemos hacer queries. Hacemos un select count para ver el numero de deslizamientos que contiene el fichero.

select count(*) from deslizamientos;

Vemos que la select realmente lanza por debajo un proceso de tipo MapReduce, lanzando los mappers y los reducers necesarios para hacer esta operación. Cuando ha terminado vemos que tenemos unas 9563 filas

```
INFO  : MapReduce Total cumulative CPU time: 2 seconds 170 msec
INFO  : Ended Job = job_1679153125332_0002
INFO  : MapReduce Jobs Launched:
INFO  : Stage-Stage-1: Map: 1  Reduce: 1  Cumulative CPU: 2.17 sec    HDFS R
0 HDFS Write: 104 SUCCESS
INFO  : Total MapReduce CPU Time Spent: 2 seconds 170 msec
INFO  : Completed executing command(queryId=hadoop_20230320095402_46020d06-9
c7-0304f7e78bbe); Time taken: 13.375 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
+----+
| _c0 |
+----+
| 9563 |
+----+
1 row selected (13,789 seconds)
0: jdbc:hive2://nodo1:10000/ejemplo> select count (*) from deslizamientos;
```

12. EJEMPLO REAL

Paso 8. Hacemos una query que nos indique los 5 primeros registros del country y la fecha de los deslizamientos. La cláusula limit nos limita el número de filas máximo que quiero ver.

```
0: jdbc:hive2://nodo1:10000/ejemplo> select country, fecha from deslizamientos limit 5;
INFO : Compiling command(queryId=hadoop_20230320103845_c13d091e-6dbf-4468-8d08-ef78249ae): select country, fecha from deslizamientos limit 5
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Semantic Analysis Completed (retrial = false)
INFO : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:country, type:string, comment:null), FieldSchema(name:fecha, type:string, comment:null)], properties:null)
INFO : Completed compiling command(queryId=hadoop_20230320103845_c13d091e-6dbf-4468-8df78249a9b0e); Time taken: 0.075 seconds
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Executing command(queryId=hadoop_20230320103845_c13d091e-6dbf-4468-8d08-ef78249ae): select country, fecha from deslizamientos limit 5
INFO : Completed executing command(queryId=hadoop_20230320103845_c13d091e-6dbf-4468-8df78249a9b0e); Time taken: 0.0 seconds
INFO : OK
INFO : Concurrency mode is disabled, not creating a lock manager
+-----+-----+
| country | fecha |
+-----+-----+
| United Kingdom | 01/02/2007 |
| Peru | 01/03/2007 |
| Brazil | 01/05/2007 |
| Brazil | 01/05/2007 |
| Brazil | 01/05/2007 |
+-----+-----+
5 rows selected (0.099 seconds)
0: jdbc:hive2://nodo1:10000/ejemplo>
```

12. EJEMPLO REAL

Paso 9. Hacemos una query que nos diga el tipo de deslizamientos y en qué país ha habido más de 100 víctimas.

```
select country, fecha, landslide_type, motivo, fatalities from
deslizamientos where fatalities>100;
```

INFO : Concurrency mode is disabled, not creating a lock manager				
country	fecha	landslide_type	motivo	fatalities
Afghanistan	03/28/2007	Landslide	Flooding	114
Bangladesh	06/11/2007	Landslide	Monsoon	128
China	05/17/2008	Mudslide	Earthquake	200
China	09/08/2008	Complex	Dam_Embankment_Collapse	277
Brazil	11/24/2008	Landslide	Continuous_rain	109
Taiwan	08/10/2009	Complex	Tropical_Cyclone	491
Philippines	10/09/2009	Landslide	Tropical_Cyclone	104
Uganda	03/01/2010	Complex	Downpour	388
Brazil	04/07/2010	Mudslide	Downpour	196
India	08/06/2010	Landslide	Downpour	234
India	08/06/2010	Landslide	Downpour	182
China	08/07/2010	Landslide	Downpour	1765
Indonesia	10/04/2010	Landslide	Downpour	145
Brazil	01/12/2011	Mudslide	Downpour	424
Brazil	01/12/2011	Mudslide	Downpour	378
Philippines	12/04/2012	Mudslide	Tropical_Cyclone	430
India	06/16/2013	Debris_Flow	Downpour	5000
Afghanistan	05/02/2014	Landslide	Continuous_rain	2100
India	07/30/2014	Mudslide	Continuous_rain	151
Nepal	08/02/2014	Landslide	Continuous_rain	174
	01/04/2006	Mudslide	Downpour	240
	12/12/2014	Landslide	Monsoon	108
	04/28/2015	Mudslide	Snowfall_snowmelt	250
	10/01/2015	Mudslide	Rain	280
	08/02/2015	Landslide	Downpour	253
	05/18/2016	Mudslide	Monsoon	101
	04/02/2016	Landslide	Unknown	104

27 rows selected (0.158 seconds)

0: jdbc:hive2://nodo1:10000/ejemplo> select country, fecha, landslide_type, motivo, fatalities from deslizamientos where fatalities>100;

Nos indica el motivo, el tipo si ha sido de tierra o de barro y el número de víctimas que se han producido.

En este caso no le ha hecho falta hacer un MapReduce porque es capaz de hacerlo sin hacer el proceso.

Ya estamos haciendo análisis de datos. Si tuviéramos que hacer esto mediante un programa en MapReduce ¿cuanto nos llevaría?

12. EJEMPLO REAL

Paso 10. Otro ejemplo: Nos piden contar el numero de deslizamientos de tierra que ha habido por cada tipo (`landslide_type`). Debemos agrupar por este campo. Lanzara también un programa MapReduce

**Select `landslide_type`, count(*) from `deslizamientos`
group by `landslide_type`:**

<code>landslide_type</code>	<code>_c1</code>
"and Netala, all on the Rishikesh-Yamunotri National Highway"	18
Complex	232
Creep	5
Debris_Flow	173
Earthflow	3
Lahar	7
Landslide	6657
Mudslide	1828
Other	66
Riverbank_Collapse	28
Rockfall	485
Rockslide	1
Snow_Avalanche	7
Translational_Slide	6
Unknown	18
<code>landslide</code>	21
<code>mudslide</code>	7

```
18 rows selected (13,69 seconds)
0: jdbc:hive2://nodo1:10000/ejemplo> select landslide_type, count(*) from deslizamientos group by landslide_type;
```

12. EJEMPLO REAL

Paso 11. Podemos averiguar cuales han sido los principales motivos de que se hayan producido estos corrimientos.

select motivo, count(*) from deslizamientos group by motivo;

INFO : Concurrency mode is disabled, not creating a lock manager	
motivo	c1
Construction	756
Continuous_Rain	52
Continuous_rain	39
Dam_Embankment_Collapse	518
Downpour	9
Earthquake	4447
Flooding	76
Freeze_thaw	49
Landslide	26
Mining_digging	14
Monsoon	74
Mudslide	122
No_Apparent_Trigger	2
No_Apparent_trigger	2
Other	18
Rain	15
Rockfall	1914
Snowfall_snowmelt	1
Tropical_Cyclone	75
Unknown	539
Volcano	750
landslide	1
monsoon	1
unknown	61

25 rows selected (14,671 seconds)

```
0: jdbc:hive2://nodo1:10000/ejemplo> select motivo, count(*) from deslizamientos
group by motivo;
```

Preguntando a Hive, no tenemos que hacer un programa en Java o Python, para recuperar esta información.

No importa que por debajo lance MapReduce, Spark, o Tez.

Lo importante es que con un lenguaje más fácil de utilizar y mas conocido, podemos hacer este tipo de análisis

12. EJEMPLO REAL

Paso 12. Otro ejemplo: cuales son los 10 países que tienen más deslizamientos registrados

```
select country, count(*) as total from deslizamientos  
group by country order by total desc limit 10;
```

Nota: Total es un alias de una columna de la query

```
INFO  : OK  
INFO  : Concurrency mode is disabled, not creating a lock manager  
+-----+-----+  
|   country      | total    |  
+-----+-----+  
|           | 3397    |  
| United States | 1444    |  
| India        | 890     |  
| Philippines  | 548     |  
| China        | 348     |  
| Nepal        | 324     |  
| Indonesia    | 288     |  
| Brazil        | 205     |  
| United Kingdom | 149     |  
| Malaysia     | 112     |  
+-----+-----+  
10 rows selected (31,554 seconds)  
0: jdbc:hive2://nodo1:10000/ejemplo> select country, count(*) as total from deslizamientos group by country order by total desc limit 10;
```

12. EJEMPLO REAL

Paso 13. Mediante HiveSql también podemos unir dos tablas usando la clausula join de SQL (outer, left, right). Para ello primero crearemos una segunda tabla llamada paises.

create table paises (nombre string, cod string)

row format delimited

fields terminated by ',';

```
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
+-----+
|    tab_name    |
+-----+
| deslizamientos |
| paises          |
| t1              |
| t2              |
+-----+
4 rows selected (0,033 seconds)
0: jdbc:hive2://nodo1:10000/ejemplo> █
```

12. EJEMPLO REAL

Paso 14. Cargaremos el fichero countries.csv en la tabla países (se encuentra en el carpeta del drive de recursos). Haremos:

nodo1 → cp /home/hadoop/Downloads/countries.csv /tmp

nodo3→load data local inpath '/tmp/countries.csv' into table paises;

```
0: jdbc:hive2://nodo1:10000/ejemplo> load data local inpath '/home/home/Downloads/countries.csv' into table paises;
Error: Error while compiling statement: FAILED: SemanticException Line 1:23 Invalid path ''/home/home/Downloads/cour
home/home/Downloads/countries.csv (state=42000,code=40000)
0: jdbc:hive2://nodo1:10000/ejemplo> load data local inpath '/home/hadoop/Downloads/countries.csv' into table paises;
Error: Error while compiling statement: FAILED: SemanticException Line 1:23 Invalid path ''/home/hadoop/Downloads/cd
:/home/hadoop/Downloads/countries.csv (state=42000,code=40000)
0: jdbc:hive2://nodo1:10000/ejemplo> load data local inpath '/home/hadoop/Downloads/countries.csv' into table paises;
INFO  : Compiling command(queryId=hadoop_20230322214414_5113d941-0b46-477c-a9a7-470b14e430a9): load data local inpat
nto table paises
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO  : Completed compiling command(queryId=hadoop_20230322214414_5113d941-0b46-477c-a9a7-470b14e430a9); Time taken:
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hadoop_20230322214414_5113d941-0b46-477c-a9a7-470b14e430a9): load data local inpat
nto table paises
INFO  : Starting task [Stage-0:MOVE] in serial mode
INFO  : Loading data to table ejemplo.paises from file:/home/hadoop/Downloads/countries.csv
INFO  : Starting task [Stage-1:STATS] in serial mode
INFO  : Completed executing command(queryId=hadoop_20230322214414_5113d941-0b46-477c-a9a7-470b14e430a9); Time taken:
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
No rows affected (0,296 seconds)
0: jdbc:hive2://nodo1:10000/ejemplo> █
```

12. EJEMPLO REAL

Paso 15. Hacemos un select en la tabla recién creada para examinar su contenido:

select * from paises limit 10;

Vemos el nombre del país más su código correspondiente

```
INFO  : Concurrency mode is disabled, not creating a lock manager
+-----+-----+
| paises.nombre | paises.cod |
+-----+-----+
| Name          | Code      |
| Afghanistan   | AF        |
| Åland Islands | AX        |
| Albania       | AL        |
| Algeria       | DZ        |
| American Samoa| AS        |
| Andorra       | AD        |
| Angola         | AO        |
| Anguilla       | AI        |
| Antarctica    | AQ        |
+-----+-----+
10 rows selected (0,115 seconds)
0: jdbc:hive2://nodo1:10000/ejemplo> select * from paises limit 10;
```

12. EJEMPLO REAL

Paso 16. Vamos a enlazar las dos tablas a través del nombre del país para así acceder a su código correspondiente. Hacemos una query de tipo join entre los campos nombre de países y country de deslizamientos y agrupamos por el código, por el país y por el motivo.

```
select a.cod, b.country, b.motivo, count(*) from paises a join
deslizamientos b on a.nombre=b.country group by a.cod,
b.country, b.motivo;
```

US	United States	Mining_digging	1	
US	United States	Mudslide	1	
US	United States	Other	3	
US	United States	Rain	409	
US	United States	Rockfall	1	
US	United States	Snowfall_snowmelt	11	
US	United States	Tropical_Cyclone	10	
US	United States	Unknown	37	
US	United States	unknown	35	
VC	Saint Vincent and the Grenadines	Tropical_Cyclone	7	
VU	Vanuatu	Rain	1	
YE	Yemen	Downpour	4	
YE	Yemen	Rain	1	
ZA	South Africa	Downpour	7	
ZA	South Africa	Other	1	
ZA	South Africa	Rain	9	
ZM	Zambia	Downpour	1	

347 rows selected (16,447 seconds)

```
0: jdbc:hive2://nod01:10000/ejemplo> select a.cod, b.country, b.motivo, count(*) from paises a join
deslizamientos b on a.nombre=b.country group by a.cod,b.country, b.motivo;
```

12. EJEMPLO REAL

Paso 17. La idea de la join es que en la tabla deslizamiento no tenemos el código del país, pero se podría obtener de la tabla países que sí lo tiene. Uniendo las dos tablas en la misma query a partir de la columna del nombre del país, podemos sacar el código que está en una tabla países.

```

INFO : Concurrency mode is disabled,
+-----+
| tab_name      |
+-----+
| deslizamientos |
| paises         |
| t1             |
| t2             |
+-----+
4 rows selected (0,028
0: jdbc:hive2://nodo1:10000/ejemplo> select * from
INFO : Concurrency mode is disabled, not creating a lock manager
+-----+-----+
| paises.nombre | paises.cod |
+-----+-----+
| Name          | Code   |
+-----+-----+
| Afghanistan   | AF
| Åland Islands | AX
| Albania       | AL
| Algeria       | DZ
| American Samoa | AS
| Andorra       | AD
| Angola         | AO
| Anguilla       | AI
| Antarctica    | AQ
+-----+-----+
10 rows selected (0,115 seconds)
0: jdbc:hive2://nodo1:10000/ejemplo> select * from
+-----+-----+-----+-----+
| id   | country        | nearest_places           | landslide_type |
+-----+-----+-----+-----+
| 1    | United Kingdom | Whitehaven, Cumbria     | Landslide     |
| 2    | Peru            | Alto Mesapata, in the Pasco province | Complex      |
| 3    | Brazil          | Nova Friburgo, Rio de Janeiro | Landslide    |
| 4    | Brazil          | Sumidouro, Rio de Janeiro | Landslide    |
| 5    | Brazil          | Jundiaí, São Paulo      | Landslide    |
| 6    | Pakistan        | Hallar Bridge, Kotli    | Landslide    |
| 7    | Brunei          | Bandar Seri Begawan    | Landslide    |
| 8    | Indonesia       | Sungai Sariak, Sumatra  | Complex      |
| 9    | Philippines     | Inupuan in Barangay Mainit, Nabunturan, Compostela Valley Province | Landslide    |
| 10   | Sri Lanka        | Nuwara Eliya, Sri Lanka | Landslide    |
+-----+-----+-----+-----+
10 rows selected (0,09 seconds)
0: jdbc:hive2://nodo1:10000/ejemplo> select id,country, nearest_places, landslide_type  from deslizamientos limit 10;

```

Cuando se ejecuta esta query, empieza a lanzar distintos procesos mappers y reducers dependiendo de la complejidad

12. EJEMPLO REAL

Paso 18. ¿Como se podría mostrar la información obtenida en la consulta de join con un gráfico de barras?

Hay herramientas en BigData para hacer eso, pero una alternativa fácil seria que BigData generara un fichero CSV con esta información, de manera que el programa Excel nos permitiera hacer el gráfico

El típico ciclo de vida de BigData es:

- Cargamos un fichero CSV en Hive
- Hacemos una serie de operaciones de transacciones
- Generamos la información deseada y la guardamos en un fichero
- Llevamos ese fichero a mi Excel para hacer el tratamiento grafico

Big Data Hadoop es un entorno de procesamiento donde lo normal es que la información que genere muchas veces, no la usamos dentro de Big Data, nos la llevamos a otro sitio para tratarla o para gestionarla.

12. EJEMPLO REAL

Paso 19. El comando overwrite nos va a permitir guardar en un fichero del directorio /tmp/datos, la información generada por la query-join, con el formato de los campos separados por coma:

insert overwrite local directory '/tmp/datos/' row format delimited fields terminated by ',' select a.cod, b.country, b.motivo, count(*) from paises a join deslizamientos b on a.nombre=b.country group by a.cod, b.country, b.motivo;

```
0: jdbc:hive2://nodo1:10000/ejemplo> insert overwrite local directory '/tmp/datos' row format delimited fields terminated by ',' select a.cod, b.country, b.motivo, count(*) from paises a join deslizamientos b on a.nombre=b.country group by a.cod,b.country, b.motivo;
```

Paso 20. El resultado de la query se guarda dentro del directorio local /tmp/datos del nodo1, en un fichero que se llama 000000_0:

```
hadoop@nodo1:~/Downloads$ cd /tmp/datos
hadoop@nodo1:/tmp/datos$ ls
000000_0
hadoop@nodo1:/tmp/datos$
```

12. EJEMPLO REAL

Paso 21. Si lo editamos vemos que contiene los registros de la query-join separados por comas.

```
GNU nano 6.2          000000 0
AE,United Arab Emirates,Rain,1
AF,Afghanistan,Continuous_rain,1
AF,Afghanistan,Downpour,4
AF,Afghanistan,Flooding,1
AF,Afghanistan,Rain,4
AL,Albania,Downpour,1
AM,Armenia,Downpour,3
AO,Angola,Downpour,3
AR,Argentina,Downpour,5
AR,Argentina,Rain,1
AS,American Samoa,Downpour,4
AS,American Samoa,Rain,1
AT,Austria,Downpour,7
AT,Austria,Rain,2
AT,Austria,Snowfall_snowmelt,1
AU,Australia,Continuous_rain,2
AU,Australia,Downpour,56
AU,Australia,Mining_digging,1
AU,Australia,Rain,15
AU,Australia,Tropical_Cyclone,1
AU,Australia,Unknown,4
AZ,Azerbaijan,Downpour,16
AZ,Azerbaijan,Snowfall_snowmelt,1
```

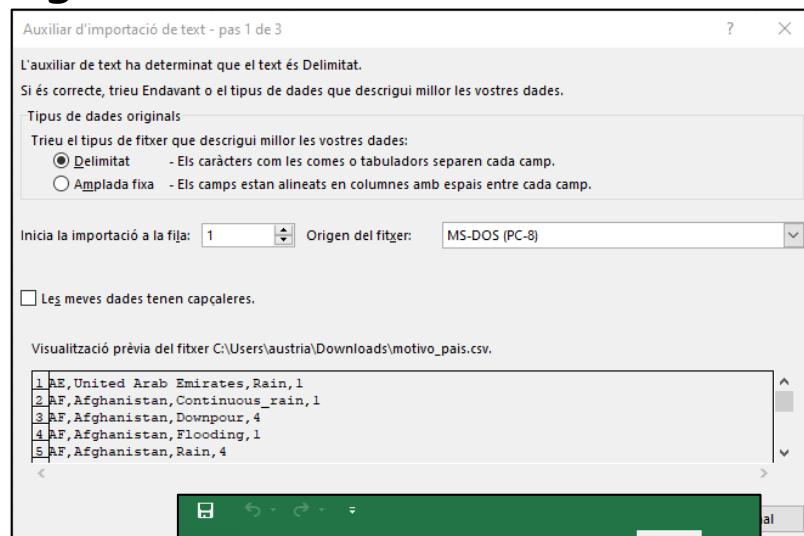
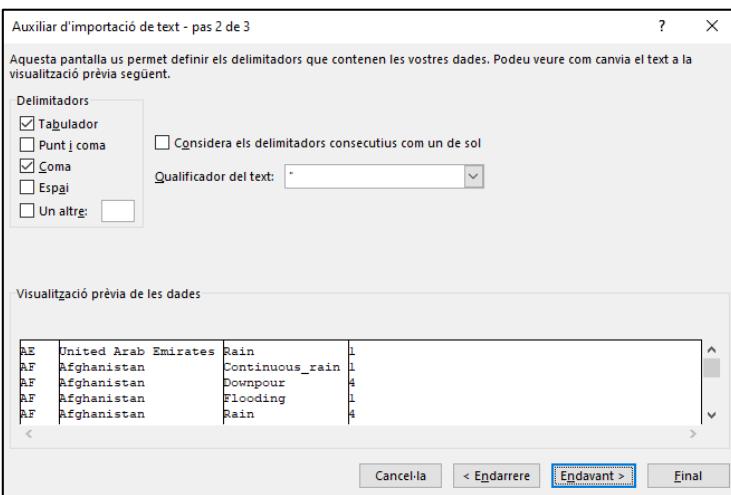
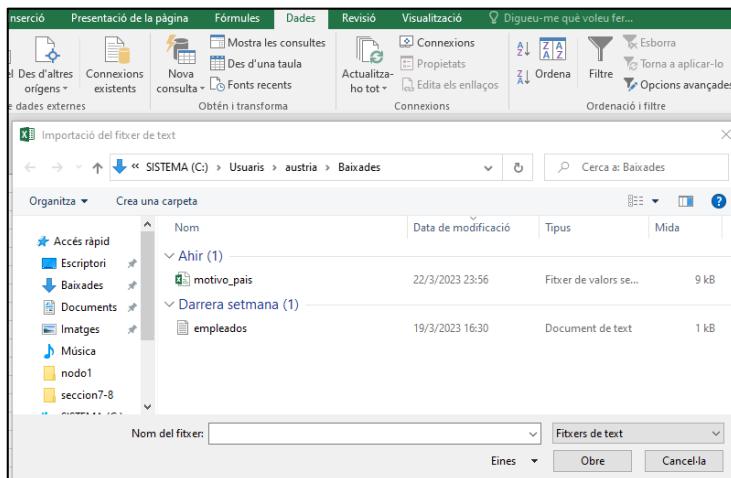
Paso 22. Llevamos el fichero a un equipo Windows mediante mail o ftp
Le cambiamos el nombre y la extensión.

Nom	
▼ Avui (1)	—
 000000_0	

Nom	
▼ Avui (1)	—
 motivo_pais	

12. EJEMPLO REAL

Paso 23. Abrimos un excel y cargamos el fichero csv desde el menú Datos/Desde el texto



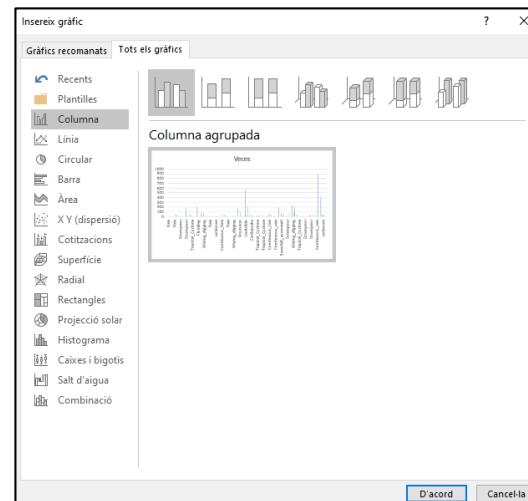
	A	B	C	D	E
1	AE	United Arab Emirates	Rain	1	
2	AF	Afghanistan	Continuous_rain	1	
3	AF	Afghanistan	Downpour	4	
4	AF	Afghanistan	Flooding	1	
5	AF	Afghanistan	Rain	4	
6	AL	Albania	Downpour	1	
7	AM	Armenia	Downpour	3	
8	AO	Angola	Downpour	3	
9	AR	Argentina	Downpour	5	
10	AR	Argentina	Rain	1	
11	AS	American Samoa	Downpour	4	
12	AS	American Samoa	Rain	1	
13	AT	Austria	Downpour	7	
14	AT	Austria	Rain	2	
15	AT	Austria	Snowfall_snowmelt	1	
16	AU	Australia	Continuous_rain	2	

12. EJEMPLO REAL

Paso 24. Ponemos el nombre de las columnas de los datos: código, países, motivo y veces

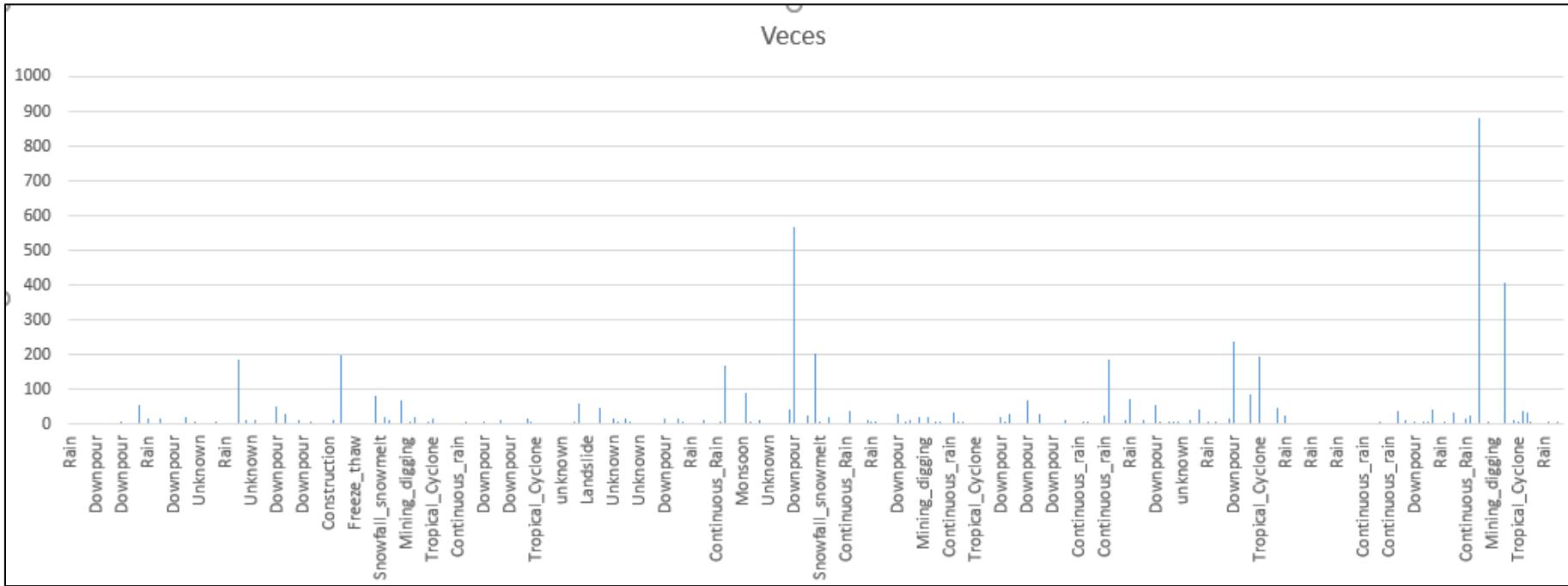
	A	B	C	D
1	Codigo	Paises	Motivo	Veces
2	AE	United Arab Emirates	Rain	1
3	AF	Afghanistan	Continuous_rain	1
4	AF	Afghanistan	Downpour	4
5	AF	Afghanistan	Flooding	1

Paso 25. Como es un excel seleccionamos solo las celdas motivo y veces e indicamos que nos inserte un grafico recomendado. Elegimos por ejemplo columna agrupada



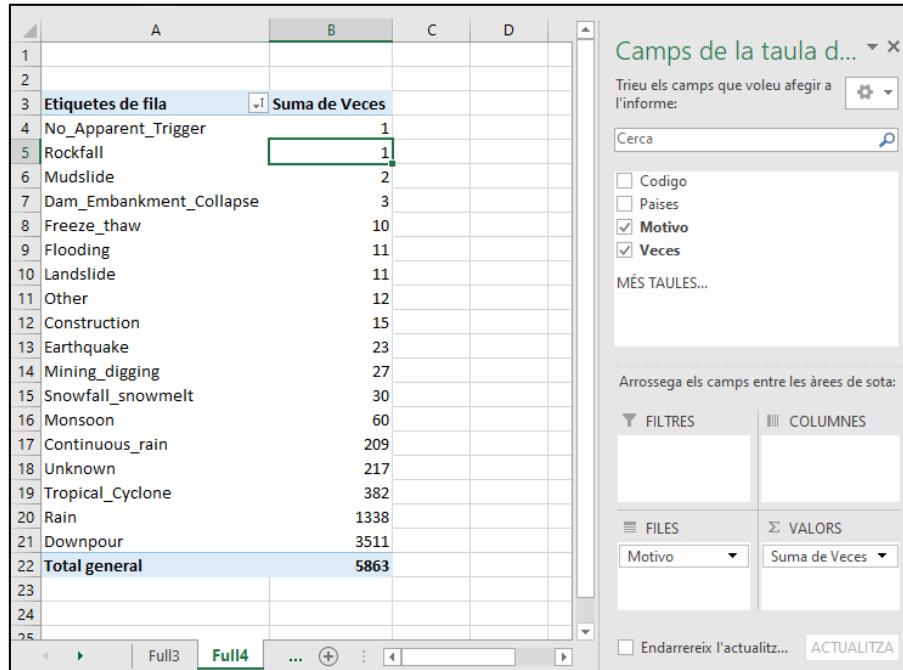
12. EJEMPLO REAL

Paso 26. Obtenemos una grafica con toda la información de el motivo y la suma de veces.



12. EJEMPLO REAL

Paso 27. También podríamos hacer una tabla dinámica y ordenar la información en formato histograma



NOTA: Hemos conseguido desde un entorno Big Data hacer una serie de procesos en Hive y luego traernos los resultados a un Excel para hacer un gráfico que permita ver con mas detalles la información