

Herramientas del ecosistema

Hadoop

© EDICIONES ROBLE, S.L.

< S.L.

Indice

Herramientas del ecosistema Hadoop	4
I. Introducción	4
II. Objetivos	4
III. PIG	4
3.1. Introducción y objetivos	4
3.2. Instalación	6
3.3. Grunt Shell	10
3.3.1. .pigbootup	10
3.3.2. Acceso a HDFS	12
3.3.3. Otros comandos	16
3.3.4. log4j	18
3.4. Introducción a Pig Latin	18
3.4.1. Tipos de datos	19
3.4.2. Esquemas	19
3.4.3. Operadores	20
3.4.4. UDF (user defined functions)	36
3.5. Pig Latin avanzado	43
3.5.1. Algunas funciones	43
3.5.2. Ejemplo	46
3.5.3. Ejemplo de análisis de logs	50
3.6. Lectura recomendada	56
IV. HIVE	56
4.1. Introducción y objetivos	57
4.2. Instalación	57
4.3. Configuración	60
4.4. Comprobación	63
4.5. Comandos consola	66
4.6. Arquitectura	72
4.7. Metastore	74
4.8. Tipos de datos	74
4.8.1. Tipos simples	74
4.8.2. Tipos complejos	75
4.9. Almacenamiento de información	76
4.10. Operadores	77
4.11. Otras funciones	78
4.12. Ejemplo: word count	79
4.13. Estructuras de datos	84
4.13.1. Bases de datos	85
4.13.2. Tablas	88
4.13.3. Tablas particionadas	90
4.13.4. Tablas internas y externas	91
4.13.5. Buckets	92
4.13.6. Índices	92
4.13.7. Vistas	94
4.14. Ejemplo: estructuras complejas	95
4.15. Trabajando con Json	103
4.16. Data Definition Language (DDL)	106
4.17. Data Manipulation Language (DML)	107

4.18. Ejemplo de almacenamiento	114
4.19. Almacenamiento de datos	124
4.19.1 Almacenamiento en ficheros	124
4.19.2. Almacenamiento en registros	126
4.20. HIVE y UDF13	129
4.21. Lectura recomendada	131
V. SQOOP	132
5.1. Ejemplo	141
5.2. Sqoop: conexión	142
5.3. Sqoop: migración	143
5.4. Sqoop: migraciones a otros sistemas	144
VI. HBASE	145
6.1. Introducción y objetivos	145
6.2. Terminología del modelo de datos de HBase	147
6.3. Instalación	148
6.4. Configuración de ficheros	151
6.5. Arrancar HBase	153
6.6. Arrancar shell de comandos	153
6.7. Zookeeper	154
6.8. Estructura de directorios	155
6.9. Modos de funcionamiento	158
6.10. Comandos disponibles en HBase shell	159
6.10.1. Comandos generales	159
6.10.2. Comandos DDL (definición de datos)	160
6.10.3. Comandos de manipulación de datos CRUD	166
6.10.4. Comandos avanzados	169
6.11. Ejemplo	173
6.12. Ejemplo shell	180
6.13. Integración con PIG	188
6.14. Lectura recomendada	193
VII. Y esto no termina aquí...	194
VIII. Resumen	195
IX. Anexos	195
Ejercicios	196
Ejercicio 1: PIG	196
Solución	197
Ejercicio 2: HIVE	205
Solución	206
Ejercicio 3: HBASE	210
Solución	211
Recursos	219
Documentos	219
Enlaces de Interés	219
Bibliografía	220
Glosario.	220

Herramientas del ecosistema Hadoop

I. Introducción

Tras el repaso general al ecosistema Hadoop de la UD1, en esta segunda unidad vamos a ver algunas de las herramientas que complementan a los elementos principales del ecosistema.

Veremos herramientas como PIG, HIVE o HBASE, tecnologías que cubren distintas funcionalidades alrededor del núcleo de Hadoop.

II. Objetivos



Los objetivos que alcanzarán los alumnos tras el estudio de esta unidad son:

- Poder explicar el ecosistema de Hadoop.
- Aprender a configurar un servidor Hadoop y algunas de sus herramientas más comunes.
- Entender el funcionamiento del sistema de almacenamiento de Hadoop.
- Entender qué es y para qué sirven las técnicas de Map Reduce.
- Aprender conceptos básicos de manejo de datos en HDFS empleando herramientas como PIG y HIVE.

Debido a la extensión del temario, hemos dividido la unidad en varios bloques con temáticas distintas.



Enlaces: Se puede encontrar más información en los siguientes enlaces

- [Página web del proyecto Hadoop de la Fundación Apache](#).
- [Hadoop Wiki: "Apache Hadoop"](#).

III. PIG

3.1. Introducción y objetivos



Los objetivos de este apartado son:

- Conocer el origen y la historia de PIG.
- Aprender a instalarlo sobre un servidor Hadoop.
- Aprender la sintaxis básica de Grunt.
- Conocer la sintaxis de programación en PIG (Pig Latin).
- Saber cómo se pueden definir funciones en PIG.
- Resolver ejercicios simples con PIG.



PIG es un lenguaje procedural de alto nivel empleado para el análisis de datos y ejecución de procesos en paralelo.

Apache PIG es una plataforma de análisis de grandes conjuntos de datos que proporciona un lenguaje de programación de alto nivel y toda la infraestructura que permite la evaluación de los programas desarrollados en dicho lenguaje. La característica más destacada de los programas de PIG es que su estructura es susceptible de parallelización, lo que a su vez permite manejar grandes conjuntos de datos.

En la actualidad, la capa de infraestructura de PIG se compone de un compilador que produce secuencias de programas de Map Reduce y de una capa de lenguaje textual denominado Pig Latin, que tiene las siguientes propiedades clave:

Facilidad de programación

Es sencillo lograr la ejecución paralela de tareas de análisis de datos simples. Las tareas complejas compuestas de múltiples transformaciones de datos relacionados entre sí están codificadas explícitamente como secuencias de flujo de datos, lo que hace que sean fáciles de escribir, entender y mantener.

Flujo de datos o grafos acíclicos (DAG)

Pig Latin es un lenguaje de flujo de datos. Esto significa que permite a los usuarios describir cómo los datos de una o más entradas deben ser leídos, procesados y almacenados en una o varias salidas en paralelo. Debido a esta forma de codificación de tareas, permite que el sistema pueda optimizar su ejecución de forma automática en los distintos nodos del clúster, ayudando al usuario a centrarse en la semántica en vez de en la eficiencia. Los flujos de datos pueden ser lineales o complejos, con varias entradas que se unen entre sí.

Map Reduce

Pig Latin ofrece operadores para muchas de las operaciones de datos tradicionales (unir, ordenar, agrupar, filtrar, etc.), así como la posibilidad de que los usuarios desarrollen sus propias funciones (*User Defined Functions*: UDF) de lectura, procesamiento y escritura de datos. Estas funciones pueden desarrollarse en múltiples lenguajes como JRuby, Python y Java.

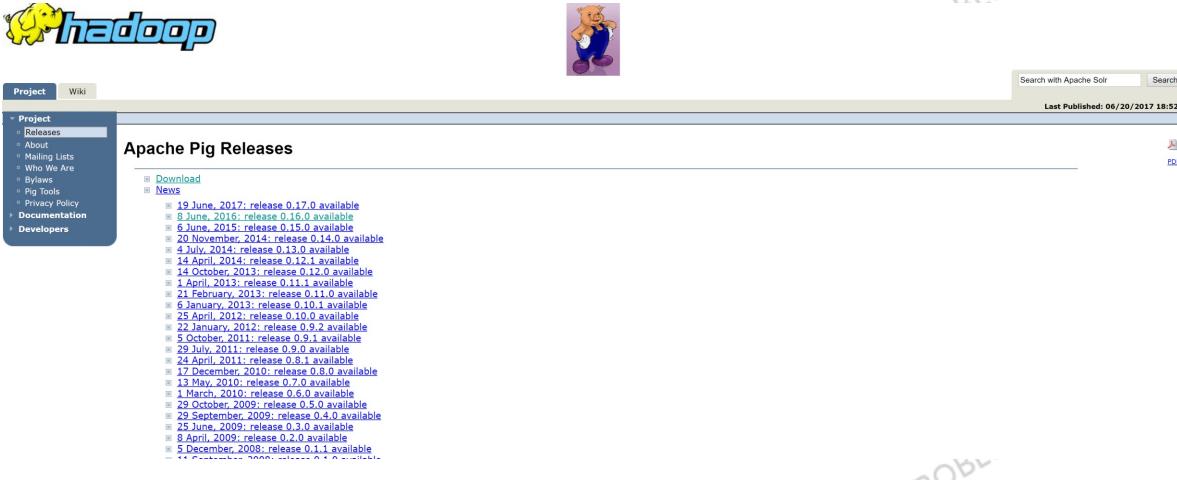
Código abierto

PIG es un proyecto de código abierto de Apache, es decir, los usuarios son libres dedescargarlo como fuente o binario, utilizarlo para sí mismos, contribuir a él y, bajo los términos de la licencia de uso de Apache, modificarlo si lo consideran conveniente.

3.2. Instalación

1

Para proceder a la instalación de PIG, hay que acceder a sus repositorios de descargas y seleccionar la última versión. Para este curso trabajaremos con la versión 0.16 (si se desea se puede trabajar con versiones posteriores), se ha dejado la herramienta descargada en el directorio home del usuario BigData para que se pueda seguir correctamente el módulo.



The screenshot shows the Apache Pig Releases page. At the top, there are two small icons: a yellow elephant icon for Hadoop and a purple bear icon for Pig. Below the header, there's a search bar and a link to the Apache Solr search results. The main content area is titled "Apache Pig Releases" and lists a chronological history of releases:

- 19 June, 2017: release 0.17.0 available
- 8 June, 2016: release 0.16.0 available
- 9 May, 2015: release 0.15.0 available
- 20 November, 2014: release 0.14.0 available
- 4 July, 2014: release 0.13.0 available
- 14 April, 2014: release 0.12.1 available
- 14 October, 2013: release 0.12.0 available
- 21 February, 2013: release 0.11.0 available
- 6 January, 2013: release 0.10.1 available
- 25 April, 2012: release 0.10.0 available
- 24 January, 2012: release 0.9.2 available
- 5 October, 2011: release 0.9.1 available
- 29 July, 2011: release 0.9.0 available
- 24 April, 2011: release 0.8.1 available
- 17 December, 2010: release 0.8.0 available
- 13 May, 2010: release 0.7.0 available
- 23 March, 2010: release 0.6.0 available
- 29 October, 2009: release 0.5.0 available
- 29 September, 2009: release 0.4.0 available
- 25 June, 2009: release 0.3.0 available
- 8 April, 2009: release 0.2.0 available
- 2 December, 2008: release 0.1.1 available

2

Antes de descargarse una versión, hay que comprobar que es compatible con la versión de Hadoop instalada:

8 June, 2016: release 0.16.0 available

The highlights of this release is the stablization of Pig on Tez

Note

This release works with Hadoop 0.23.X, 1.X and 2.X

3

Una vez comprobado este prerequisito, se procede a efectuar la descarga.



We suggest the following mirror site for your download:

<http://ftp.cixug.es/apache/pig>

Other mirror sites are suggested below. Please use the backup mirrors only to download PGP and MD5 signatures to verify your downloads or if no other mirrors are working.

HTTP

<http://apache.rediris.es/pig>

<http://apache.uvigo.es/pig>

<http://ftp.cixug.es/apache/pig>

4

Para ello tenemos que seleccionar el *mirror* correspondiente, en nuestro caso <http://apache.rediris.es/pig>. Una vez seleccionado, descargaremos la versión deseada.



Servicio de replicas de RedIRIS - <ftp.rediris.es>

Pig Releases

Please make sure you're downloading from a [nearby mirror site](#), not from www.apache.org.

Older releases are available from the [archives](#).

Name	Last modified	Size	Description
Parent Directory		-	
latest/	2017-06-26 19:50	-	
pig-0.16.0/	2017-06-26 19:50	-	
pig-0.17.0/	2017-06-26 19:50	-	

5

Para realizar la descarga desde la consola de Linux usaremos el siguiente comando:

```
$ wget http://apache.rediris.es/pig/pig-0.16.0/pig-0.16.0.tar.gz
```

```
bigdata@bigdata:~$ wget http://apache.rediris.es/pig/pig-0.16.0/pig-0.16.0.tar.gz
--2017-08-27 13:44:16--  http://apache.rediris.es/pig/pig-0.16.0/pig-0.16.0.tar.gz
Resolviendo apache.rediris.es (apache.rediris.es)... 130.206.13.2
Conectando con apache.rediris.es (apache.rediris.es)[130.206.13.2]:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 177279333 (169M) [application/x-gzip]
Guardando como: "pig-0.16.0.tar.gz"

pig-0.16.0.tar.gz          6%[=>                                ] 11,66M   225KB/s  eta 3m 37s
```

6

Tras descargarlo, se debe descomprimir y mover al directorio de trabajo:

```
$ sudo tar -xvf pig-0.16.0.tar.gz
```

```
bigdata@bigdata:~$ sudo tar -xvf pig-0.16.0.tar.gz
```

```
$ sudo mv pig-0.16.0 /home/bigdata/pig
```

```
bigdata@bigdata:~$ sudo mv pig-0.16.0 /home/bigdata/pig
```

```
$ sudo rm pig-0.16.0.tar.gz
```

```
bigdata@bigdata:~$ sudo rm pig-0.16.0.tar.gz
```

A continuación, es necesario generar la variable de entorno PIG_HOME en el \$HOME/.bashrc apuntando al directorio de instalación de PIG.

```
$ sudo nano ~/.bashrc
```

```
bigdata@bigdata:~$ sudo nano ~/.bashrc
```

```
#PIG VARIABLES START
```

```
export PIG_HOME=/home/bigdata/pig
```

```
export PATH=$PATH:$PIG_HOME/bin
```

```
#PIG VARIABLES end
```

```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_HOME=/home/bigdata/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
#HADOOP VARIABLES END
#
#PIG VARIABLES START
export PIG_HOME=/home/bigdata/pig
export PATH=$PATH:$PIG_HOME/bin
#PIG VARIABLES end
```

Por último, se recarga la nueva configuración del .bashrc y se comprueba que PIG está correctamente instalado.

```
$ source ~/.bashrc
```

```
$ pig -version
```

```
bigdata@bigdata:~$ pig -version
Apache Pig version 0.16.0 (r1746530)
compiled Jun 01 2016, 23:10:49
```

3.3. Grunt Shell

Grunt¹ es el *shell* de comandos de PIG. El acceso al *shell* se realiza escribiendo **pig -x local**, si se arranca en modo local, o solo **pig** si es en modo clúster. Como resultado muestra el *prompt* **grunt>**. Para salir, hay que escribir **quit** o ejecutar **Ctrl + D**.

¹Pig Wiki: “Grunt Shell”. [En línea] URL disponible en: <https://wiki.apache.org/pig/Grunt>

```
bigdata@bigdata:~$ pig -x local
17/08/27 14:08:27 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
17/08/27 14:08:27 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2017-08-27 14:08:27,703 [main] INFO org.apache.pig.Main - Apache Pig version 0.16.0 (r1746530) compiled Jun 01 2016, 23:10:49
2017-08-27 14:08:27,703 [main] INFO org.apache.pig.Main - Logging error messages to: /home/bigdata/pig_1503835707699.log
2017-08-27 14:08:27,742 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file /home/bigdata/.pigbootup not found
2017-08-27 14:08:28,021 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-08-27 14:08:28,022 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-08-27 14:08:28,024 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: file:///
2017-08-27 14:08:28,358 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2017-08-27 14:08:28,399 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-53a6f906-3dba-494f-8209-8a6fb89d29ed7
2017-08-27 14:08:28,413 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
grunt> ■
```

Grunt incluye dos tipos de comandos: comandos de acceso al sistema de archivos DFS y comandos propios de PIG. A continuación se describen esos comandos, pero antes veremos un fichero de arranque que es muy útil para cargar algunas instrucciones por defecto.

3.3.1. .pigbootup

Como se ha visto anteriormente, nada más arrancar PIG aparecen algunos mensajes de información. El más importante es el mensaje asociado al fichero *pigbootup*. Este fichero se emplea para indicar aquellos comandos o sentencias que se deben arrancar una vez se arranque PIG. Se va a emplear para registrar las librerías necesarias para eliminar el resto de mensajes de error.

La ubicación del fichero *.pigbootup* se puede configurar a través de la propiedad *pig.load.default.statements*. Se puede añadir en el fichero *pig.properties* una ruta en la que se quiera ubicar este fichero.

```
$ sudo nano /home/bigdata/pig/.pigbootup
```

```
bigdata@bigdata:~$ sudo nano /home/bigdata/pig/.pigbootup
```

```
REGISTER '/home/bigdata/mapreduce/wc.jar';
DEFINE MY_UDF com.myudfs.UPPER();
SET default_parallel 10;
```

```
REGISTER '/home/bigdata/mapreduce/wc.jar';
DEFINE MY_UDF com.myudfs.UPPER();
SET default_parallel 10;
```

Para modificar el fichero de propiedades:

```
$ cd pig
$ sudo nano conf/pig.properties
```

```
bigdata@bigdata:~$ cd pig
bigdata@bigdata:~/pig$ sudo nano conf/pig.properties
```

Descomentamos la línea `pig.load.default.statements` y añadimos lo siguiente:

```
pig.load.default.statements= /home/bigdata/pig/.pigbootup

# Bootstrap file with default statements to execute in every Pig job, similar to
# .bashrc. If blank, uses the file '.pigbootup' from your home directory; If a
# value is supplied, that file is NOT loaded. This does not do tilde expansion
# -- you must supply the full path to the file.
#
pig.load.default.statements= /home/bigdata/pig/.pigbootup
# pig.load.default.statements=/home/bob/.pigrc
```

Tras realizar estos cambios, ya se puede arrancar PIG desde el `root`, tal y como se muestra en la siguiente figura:

```
bigdata@bigdata:~/pig$ pig -x local
17/08/27 14:13:27 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
17/08/27 14:13:27 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
17/08/27 14:13:27 WARN pig.Main: Cannot write to log file: /home/bigdata/pig/pig_1503836007984.log
2017-08-27 14:13:27,990 [main] INFO org.apache.pig.Main - Apache Pig version 0.16.0 (r1746530) compiled Jun 01 2016, 23:10:49
2017-08-27 14:13:28,371 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-08-27 14:13:28,371 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-08-27 14:13:28,372 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: file:///
2017-08-27 14:13:28,813 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2017-08-27 14:13:28,860 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-44ac5226-4e51-4f5d-b8cb-90622444bbd8
2017-08-27 14:13:28,871 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
grunt> REGISTER '/home/bigdata/mapreduce/wc.jar';
2017-08-27 14:13:29,142 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2017-08-27 14:13:29,154 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> DEFINE MY_UDF com.myudfs.UPPER();
grunt> SET default_parallel 10;
grunt>
```

Como se observa en el *log* de la figura anterior, se ha arrancado PIG sin errores y además ha ejecutado todos los comandos incluidos en el *script* de arranque. El resto de los mensajes que aparecen se deben a que PIG está usando unos comandos que en la nueva versión de Hadoop están obsoletos. Se podría ocultar modificando el *log4j*, pero no es necesario para su correcto funcionamiento.

3.3.2. Acceso a HDFS

En el siguiente enlace, se pueden encontrar los comandos básicos para acceder a la información del sistema de archivos HDFS: <https://wiki.apache.org/pig/Grunt>

cat

imprime el contenido de un fichero en pantalla.

cd

permite navegar por el sistema de ficheros.

copyFromLocal

permite copiar un fichero de una máquina local al sistema DFS, la sintaxis es copyFromLocal <SRC PATH> <DST PATH>.

copyToLocal

permite copiar un fichero del sistema DFS a la máquina local, la sintaxis es copyToLocal <SRC PATH> <DST PATH>.

cp

permite copiar ficheros o directorios en el sistema DFS, la sintaxis es cp <SRC PATH> <DST PATH>.

ls

lista el contenido de un directorio.

mkdir

crea un directorio, la sintaxis es mkdir <DIR>.

mv

su funcionamiento es el mismo que el del comando cp pero borra el fichero o directorio original una vez que este se copia.

pwd

permite conocer la ruta del directorio de trabajo actual.

rm

elimina los ficheros o directorios especificados, la sintaxis es rm <PATH1> <PATH2>...

1

A continuación, se muestra un ejemplo de creación de ficheros con los álbumes de la discografía de Pink Floyd y el puesto que alcanzaron en las listas británicas y estadounidenses separados por coma (,). Para ello, se deben ejecutar los siguientes comandos:

```
sudo mkdir /home/bigdata/pig/ejemplos
```

```
sudo chmod -R 777 /home/bigdata/pig/ejemplos
```

```
sudo nano /home/bigdata/pig/ejemplos/discografia
```

```
bigdata@bigdata:~/pig$ sudo mkdir /home/bigdata/pig/ejemplos
bigdata@bigdata:~/pig$ sudo chmod -R 777 /home/bigdata/pig/ejemplos
bigdata@bigdata:~/pig$ sudo nano /home/bigdata/pig/ejemplos/discografia
```

2

1967, *The Piper at the Gates of Dawn*, 131,6
 1968, *A Saucerful of Secrets*, 999,9
 1969, *Music from the Film More*, 153,9
 1969, *Ummagumma*, 74,5
 1970, *Atom Heart Mother*, 55,1
 1972, *Obscured by Clouds*, 46,6
 1973, *The Dark Side of the Moon*, 1,1
 1975, *Wish you Were Here*, 1,1
 1977, *Animals*, 3,2
 1979, *The Wall*, 1,3
 1983, *The Final Cut*, 6,1
 1987, *A Momentary Lapse of Reason*, 3,3
 1994, *The Division Bell*, 1,1
 2014, *The Endless River*, 3,1



```
bigdata@bigdata: ~pig
GNU nano 2.2.6          Archivo: ejemplos/discografia

1967, The Piper at the Gates of Dawn,131,6
1968, A Saucerful of Secrets,999,9
1969, Music from the Film More,153,9
1969, Ummagumma,74,5
1970, Atom Heart Mother,55,1
1972, Obscured by Clouds, 46,6
1973, The Dark Side of the Moon, 1,1
1975, Wish you Were Here, 1,1
1977, Animals, 3,2
1979, The Wall, 1,3
1983, The Final Cut, 6,1
1987, A Momentary Lapse of Reason,3,3
1994, The Division Bell, 1,1
2014, The Endless River, 3, 1
```

3

Debemos subir el fichero que hemos creado a HDFS; para ello, ejecutaremos los siguientes comandos (Hemos de iniciar los demonios de dfs: /home/bigdata/hadoop/sbin/start-dfs.sh):

```
hdfs dfs -mkdir /ejemplo
```

```
hdfs dfs -put /home/bigdata/pig/ejemplos/discografia /ejemplo
```

```
hdfs dfs -ls /ejemplo
```

```
bigdata@bigdata:~/pig$ hdfs dfs -mkdir /ejemplo
bigdata@bigdata:~/pig$ hdfs dfs -put /home/bigdata/pig/ejemplos/discografia /ejemplo
```

```
bigdata@bigdata:~/pig$ hdfs dfs -ls /ejemplo
Found 1 items
-rw-r--r-- 1 bigdata supergroup 401 2017-08-27 14:24 /ejemplo/discografia
```

Posteriormente, se arranca PIG con el comando `pig -x mapreduce` y ya es posible consultar el fichero introducido. Los siguientes comandos sirven para listar los ficheros almacenados (`ls`) y el contenido del fichero (`cat`):

```
bigdata@bigdata:~/pig$ pig -x mapreduce
17/08/27 14:29:13 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
17/08/27 14:29:13 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
17/08/27 14:29:13 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
17/08/27 14:29:14 WARN PigMain: Snapshot write to log file: /home/bigdata/pig/pig_1503836954041.log
2017-08-27 14:29:14,049 [main] INFO org.apache.pig.Main - Apache Pig version 0.16.0 (r1746530) compiled Jun 01 2016, 23:10:49
2017-08-27 14:29:14,981 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-08-27 14:29:14,981 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-08-27 14:29:14,984 [main] INFO org.apache.hadoop.executionengine.HExecutionEngine - Connecting to Hadoop file system at: hdfs://localhost:9000
2017-08-27 14:29:15,764 [main] INFO org.apache.pig.PigServer - Pig Script ID For the session: PIG-default-057af43b-7b20-44fa-933a-553232f4b2a2
2017-08-27 14:29:15,766 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
grunt> REGISTER '/home/bigdata/mapreduce/wc.jar';
2017-08-27 14:29:15,998 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> DEFINE MY_UDF com.myudfs.UPPER();
grunt> SET default_parallel 10;
```

```
grunt> cat hdfs://localhost:9000/ejemplo/disco
```

```
grunt> cat hdfs://localhost:9000/ejemplo/disco
1967,The Piper at the Gates of Dawn,131,6
1968,A Saucerful of Secrets,999,9
1969,Music from the Film More,153,9
1969,Ummagumma,74,5
1970,Atom Heart Mother,55,1
1972,Obscured by Clouds,46,6
1973,The Dark Side of the Moon,1,1
1975,Wish you Were Here,1,1
1977,Animals,3,2
1979,The Wall,1,3
1983,The Final Cut,6,1
1987,A Momentary Lapse of Reason,3,3
1994,The Division Bell,1,1
2014,The Endless River,3,1
grunt> █
```

NOTA. Modos de ejecución:

- Local: todos los *scripts* se ejecutarán en una máquina sin emplear HDFS ni Map Reduce.
`pig -x local`
- Modo distribuido: funciona en modo clúster con las ventajas de Hadoop.
`pig -x mapreduce`

El mismo ejemplo en el sistema de archivos local sería:

```
pig -x local
```

```
ls /home/bigdata/pig/ejemplos
```

```
cat /home/bigdata/pig/ejemplos/discografia
```

```
bigdata@bigdata:~/pig$ pig -x local
17/08/27 14:32:25 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
17/08/27 14:32:25 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
17/08/27 14:32:25 WARN pig.Main: Cannot write to log file: /home/bigdata/pig/pig_1503837145343.log
2017-08-27 14:32:25,358 [main] INFO org.apache.pig.Main - Apache Pig version 0.16.0 (r1746530) compiled Jun 01 2016, 23:10:49
2017-08-27 14:32:25,817 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-08-27 14:32:25,817 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-08-27 14:32:25,819 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: file:///
2017-08-27 14:32:26,261 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2017-08-27 14:32:26,302 [main] INFO org.apache.pig.PigServer - Plg Script ID for the session: PIG-default-f1840c80-514b-41c9-b993-c833d1c75f97
2017-08-27 14:32:26,302 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
grunt> REGISTER '/home/bigdata/mapreduce/wc.jar';
2017-08-27 14:32:26,620 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2017-08-27 14:32:26,621 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> DEFINE MY_UDF com.myudfs.UPPER();
grunt> SET default_parallel 10;
grunt>
grunt> ls /home/bigdata/pig/ejemplos
file:/home/bigdata/pig/ejemplos/discografia<r 1>          401
grunt> cat /home/bigdata/pig/ejemplos/discografia
1967,The Piper at the Gates of Dawn,131,6
1968,A Saucerful of Secrets,999,9
1969,Music from the Film More,153,9
1969,Ummagumma,74,5
1970,Atom Heart Mother,55,1
1972,Obscure by Clouds,46,6
1973,The Dark Side of the Moon,1,1
1975,Wish You Were Here,1,1
1977,Animals,3,2
1979,The Wall,1,3
1983,The Final Cut,6,1
1987,A Momentary Lapse of Reason,3,3
1994,The Division Bell,1,1
2014,The Endless River,3,1
grunt>
```

Sin embargo, desde la versión 0.5 de PIG es recomendable emplear el prefijo fs para poder acceder a todos los comandos de HDFS ya que los comandos de acceso a ficheros de Grunt pueden quedar obsoletos.

Por ejemplo:

```
grunt> fs -ls /home/bigdata/pig/ejemplos
```

```
grunt> fs -ls /home/bigdata/pig/ejemplos
Found 1 items
-rw-r--r-- 1 root root      401 2017-08-27 14:17 /home/bigdata/pig/ejemplos/discografia
grunt>
```

3.3.3. Otros comandos

Grunt incluye también una serie de comandos para controlar PIG y Map Reduce:

define

permite definir funciones de usuario parametrizadas.

describe

muestra el esquema definido para un determinado alias.

dump

muestra el contenido de un alias de PIG en la pantalla.

explain

permite visualizar las etapas de ejecución de un determinado plan.

help

lista los comandos disponibles.

illustrate

muestra una ejecución de muestra de un *script*.

kill

detiene un trabajo en curso en función de su ID.

quit

sale de Grunt.

set

permite *parsear* pares clave-valor a PIG de la forma set <key> '<value>'.

store

permite almacenar el contenido de un alias de PIG a un fichero.



La descripción de cada uno de ellos se puede encontrar en el siguiente enlace:<https://wiki.apache.org/pig/Grunt>

A continuación, se muestran algunos ejemplos de uso de los comandos anteriores describe e ilustrate sobre los discos de la sección anterior. En primer lugar, cargamos el fichero discografía al alias discos mediante el comando:

```
discos = load '/home/bigdata/pig/ejemplos/disco&gt;grafia' using PigStorage(',') as (anio, disco, eeuu, uk);
```

Una vez cargado, el comando describe muestra la información relativa al alias y el comando ilustrate muestra un ejemplo de registro almacenado en el alias. En las siguientes figuras se muestra el proceso:

```
grunt> discos = load '/home/bigdata/pig/ejemplos/disco&gt;grafia' using PigStorage(',') as (anio, disco, eeuu, uk);
2017-08-27 15:23:35,386 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checks
2017-08-27 15:23:35,396 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
```

```
grunt> describe discos;
discos: {anio: bytearray,disco: bytearray,eeuu: bytearray,uk: bytearray}
```

```
grunts> illustrate discos
2017-08-27 15:25:37,268 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-08-27 15:25:37,268 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2017-08-27 15:25:37,445 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: file:///home/bigdata/hadoop
2017-08-27 15:25:37,445 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-08-27 15:25:37,452 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
```

discos	anio:bytearray	disco:bytearray	eeuu:bytearray	uk:bytearray
	1969	Music from the Film More 153		9

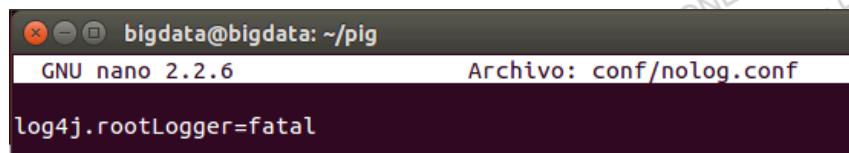
3.3.4. log4j

Para poder continuar con la formación, se debe cambiar el nivel de *log* a Fatal, para que se pueda ver mejor la información. Para ello, hay que acceder a la carpeta conf y crear un fichero nolog.conf con la línea *log4j.rootLogger=fatal*:

```
sudo nano conf/nolog.conf
```

```
log4j.rootLogger=fatal
```

```
bigdata@bigdata:~/pig$ sudo nano conf/nolog.conf
```



Una vez modificado el nivel del *log*, es necesario modificar el arranque de PIG para indicarle qué fichero de *log* debe utilizar. El comando para hacer este cambio es el siguiente:

```
pig -4 conf/nolog.conf -x local
```

```
bigdata@bigdata:~/pig$ pig -4 conf/nolog.conf -x local
17/08/27 15:30:38 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
17/08/27 15:30:38 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
17/08/27 15:30:38 WARN pig.Main: Cannot write to log file: /home/bigdata/pig/pig_1503840638375.log
17/08/27 15:30:38 INFO pig.Main: Loaded log4j properties from file: conf/nolog.conf
grunt> REGISTER '/home/bigdata/mapreduce/wc.jar';
grunt> DEFINE MY_UDF com.myudfs.UPPER();
grunt> SET default_parallel 10;
grunt>
```

3.4. Introducción a Pig Latin

Pig Latin es el lenguaje de flujo de datos empleado en PIG. Esto significa que permite a los usuarios introducir datos de uno o más orígenes de datos para después poder realizar operaciones de lectura y procesamiento y finalmente almacenar la información en una o varias salidas en paralelo aprovechando las ventajas de Map Reduce, pero abstrayendo al usuario del código Java que rige el proceso. De esta forma no es necesario que el usuario se preocupe por las fases de mapeo, mezcla y reducción, PIG se encarga de ello.

Los flujos de datos pueden ser lineales o complejos, con múltiples entradas y salidas. En ambos casos, los *scripts* de Pig Latin describen el flujo de los datos y su procesamiento.

3.4.1. Tipos de datos

Una de las ventajas del uso de PIG frente al Map Reduce es que permite definir el tipo del dato. Estos tipos pueden clasificarse en dos categorías: simples y complejos.

Tipos simples

- **int:** entero con signo de 32 bits.
- **long:** entero con signo de 64 bits.
- **float:** número en punto flotante de 32 bits.
- **double:** número en punto flotante de 64 bits.
- **chararray:** array de *strings* codificados en UTF8.
- **bytearray:** array de bytes.
- **boolean:** *true* o *false*.
- **datetime:** dato de tiempo con el formato 1970-01-01T00:00:00.000+00:00.
- **Null:** en Pig Latin, los nulos se implementan usando la definición de SQL *denull*, como desconocido o inexistente.

Tipos complejos

- **Tuple.**
 - Definición: un *tuple* es un conjunto desordenado de campos (pueden ser de cualquier tipo de datos, incluso *tuple* y *bag*).
 - Sintaxis: (field [, field ...])
 - Ejemplo: tuple con dos campos: ('John',32)
- **Bag.**
 - Definición: un *bag* es una colección de *tuples* desordenados.
 - Sintaxis: { tuple [, tuple ...] }
 - Características:
 - Un *bag* puede tener tuples duplicados.
 - Un *bag* puede tener tuples de distinto número de campos. Sin embargo, si PIG intenta acceder a un campo que no existe, devolverá un null.
 - Un *bag* puede tener tuples con diferentes tipos de datos.
 - Ejemplo: {('John', 32), ('Mary', 25), ('Jack', 36)}
- **Map.**
 - Definición: un *map* es un conjunto de pares clave-valor.
 - Sintaxis: [key#value <, key#value ...>]
 - Ejemplo: ['nombre'#'John', 'edad'#32]

3.4.2. Esquemas

Los esquemas sirven para asignar a los campos el tipo de datos. Aunque son opcionales por la filosofía de PIG (los cerdos comen cualquier cosa y en ese sentido PIG debe gestionar cualquier tipo de elemento), es recomendable declararlos ya que eso ayuda a controlar errores y mejora la eficiencia del código.

El modo más sencillo de comunicar un esquema a PIG es decírselo de forma explícita cuando se cargan datos:

```
grunt> discos = load '/home/bigdata/pig/ejemplos/discografia' using PigStorage(',') as (anio:int, disco:chararray, eeuu:int, uk:int);
```

```
grunt> describe discos;
```

En este ejemplo PIG espera cuatro campos. Si hay más, los descarta y, si tiene menos, añadirán `nulls`.

```
grunt> discos = load '/home/bigdata/pig/ejemplos/discografia' using PigStorage(',') as (anio:int, disco:chararray, eeuu:int, uk:int)
grunt> describe discos;
discos: {anio: int,disco: chararray,eeuu: int,uk: int}
```

También se puede especificar el esquema sin indicar el tipo de datos, como se vio anteriormente:

```
grunt> discos2 = load '/home/bigdata/pig/ejemplos/discografia' using PigStorage(',') as (anio, disco, eeuu, uk);
```

```
grunt> describe discos2;
```

En este caso asumirá que el tipo de datos es `bytearray`.

```
grunt> discos2 = load '/home/bigdata/pig/ejemplos/discografia' using PigStorage(',') as (anio, disco, eeuu, uk);
grunt> describe discos2;
discos2: {anio: bytearray,disco: bytearray,eeuu: bytearray,uk: bytearray}
```



A continuación, se muestran algunos ejemplos de definición de tipos de datos complejos:

`tupla: as (a:tuple(), b:tuple(x:int, y:int))`

`map: as (a:map[], b:map[int])`

`bag: as (a:bag{}, b:bag{t:(x:int, y:int)})`

`A = LOAD 'mydata' AS (T1:tuple(f1:int, f2:int), B:bag{T2:tuple(t1:float,t2:float)}, M:map[]);`

`A = LOAD 'mydata' AS (T1:(f1:int, f2:int), B:{T2:(t1:float,t2:float)},`

`M:[]);`

3.4.3. Operadores

A continuación, se muestran los operadores principales soportados por Pig Latin, así como algunos ejemplos básicos de uso.

Operadores aritméticos

- +
- -
- *
- /
- %
- ?:
- case when then else end

Operadores booleanos

- AND
- OR
- IN
- NOT

Operadores de comparación

- ==
- !=
- <
- >
- <=
- >=
- matches

Operadores sobre nulos

- is null
- is not null



A continuación, se muestra un ejemplo de uso de los operadores **and**, **>=** y **<** sobre los discos de la discografía de Pink Floyd de ejemplos anteriores. Sobre el fichero que contiene la información se crea un filtro que devolverá los registros que cumplen que el campo “anio” es mayor o igual a 1960 y menos estricto que 1970:

```
discos_sesenta = filter discs by anio >= 1960 and anio < 1970;
```

```
dump discs_sesenta;
```

```
grunt> discs_sesenta = filter discs by anio >= 1960 and anio < 1970;
grunt> dump discs_sesenta;
Total input paths to process : 1
(1967, The Piper at the Gates of Dawn,131,6)
(1968, A Saucerful of Secrets,999,9)
(1969, Music from the Film More,153,9)
(1969, Ummagumma,74,5)
```

Operadores relacionales

- CROSS: producto cartesiano de dos o más relaciones.
- DISTINCT: elimina tuplas duplicados de una relación.
- FILTER: selecciona tuplas de una relación basados en una condición.
- FOREACH: genera transformaciones de datos en función de los datos de las columnas.
- GROUP: agrupa los datos en una o más relaciones.
- JOIN (inner): *inner join* de dos o más relaciones basadas en los valores de campos comunes.
- JOIN (outer): *outer join* de dos o más relaciones basadas en los valores de campos comunes.
- LIMIT: limita el resultado de salida de los tuples.
- LOAD: carga datos desde el sistema de archivos.
- ORDER: ordena la relación en función de uno o más campos.
- SAMPLE: particiona una relación en dos o más relaciones.
- SPLIT: divide una relación en dos o más relaciones.
- STORE: almacena o guarda los resultados al sistema de archivos.
- UNION: calcula la relación de dos o más relaciones.

CROSS

Permite hacer el producto cartesiano de dos o más relaciones.

Ejemplo:

Supónganse dos relaciones cross1 y cross2 que contienen la siguiente información:

cross1: (separar con tabulador)

1 2 3

4 2 1

cross2:

2 4

8 9

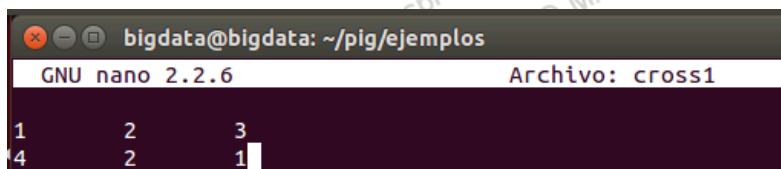
1 3

Creamos los dos ficheros con el contenido de cross1 y cross2:

sudo nano ejemplos/cross1.txt

sudo nano ejemplos/cross2.txt

```
bigdata@bigdata:~/pig$ sudo nano ejemplos/cross1
bigdata@bigdata:~/pig$ sudo nano ejemplos/cross2
```



```
bigdata@bigdata:~/pig/ejemplos
GNU nano 2.2.6                               Archivo: cross1
1      2      3
4      2      1
```



```
bigdata@bigdata:~/pig/ejemplos
GNU nano 2.2.6                               Archivo: cross2
2      4
8      9
1      3
```

Si se cargan en PIG con los alias cross1, cross2, el comando CROSS con el alias cartesiano y se ejecutan los *dumps* correspondientes, entonces se obtiene el producto cartesiano, tal y como se muestra en la siguiente figura:

```
cross1 = load 'ejemplos/cross1.txt';
dump cross1;

cross2 = load 'ejemplos/cross2.txt';
dump cross2;

cartesiano = CROSS cross1, cross2;
```

```
dump cartesiano;
```

```
grunt> cross1 = load 'ejemplos/cross1';
grunt> dump cross1;
(1,2,3)
(4,2,1)
grunt> cross2 = load 'ejemplos/cross2';
grunt> dump cross2;
(2,4)
(8,9)
(1,3)
grunt> cartesiano = CROSS cross1, cross2;
grunt> dump cartesiano
(1,2,3,8,9)
(1,2,3,1,3)
(1,2,3,2,4)
(4,2,1,1,3)
(4,2,1,2,4)
(4,2,1,8,9)
grunt> █
```

DISTINCT

Elimina los tuples duplicados en una relación.

Ejemplo:

Dada la relación (SEPARADOR ',')

```
8, 3, 4
1, 2, 3
4, 3, 3
4, 3, 3
1, 2, 3
```

Crear un fichero llamado duplicados.txt en el directorio ejemplos con el contenido anterior.

```
sudo nano ejemplos/duplicados.txt
```

```
bigdata@bigdata:~/pig$ sudo nano ejemplos/duplicados.txt
GNU nano 2.7.4                                     Archivo: ejemplos/duplicados.txt

8, 3, 4
1, 2, 3
4, 3, 3
4, 3, 3
1, 2, 3
```

Comandos que se deben ejecutar en Grunt:

```
duplicados = load 'ejemplos/duplicados.txt';
```

```
dump duplicados;
```

```
distintos = distinct duplicados;
```

```
dump distintos;
```

```
grunt> duplicados = load 'ejemplos/duplicado';
grunt> dump duplicados;
(8,3,4)
(1,2,3)
(4,3,3)
(4,3,3)
(1,2,3)
grunt> distintos = distinct duplicados;
grunt> dump distintos;
(1,2,3)
(4,3,3)
(8,3,4)
```

FILTER

Selecciona tuples de una relación basándose en una condición.

Ejemplo:

Dada la relación siguiente (SEPARADOS POR TABULADOR)

1 2 3

4 2 1

1 2 3

8 3 4

4 3 3

7 2 5

8 4 3

Crear un fichero llamado duplicados.txt en el directorio ejemplos con el contenido anterior.

```
sudo nano ejemplos/filter.txt
```

```
bigdata@bigdata:~/pig$ sudo nano ejemplos/filter.txt
```

```
GNU nano 2.7.4
1      2      3
4      2      1
1      2      3
8      3      4
4      3      3
7      2      5
8      4      3
```

Comandos que se deben ejecutar en Grunt:

```
data = load 'ejemplos/filter.txt' as (c1: int, c2: int, c3:int);
describe data;
dump data;
result = filter data by c3 ==3;
dump result;
result2= filter data by (c1==8) or (not (c2+c3>c1));
dump result2;
```

```
grunt> data = load 'ejemplos/filter.txt' as (c1: int, c2: int, c3:int);
grunt> describe data;
data: {c1: int,c2: int,c3: int}
grunt> dump data;
(1,2,3)
(4,2,1)
(1,2,3)
(8,3,4)
(4,3,3)
(7,2,5)
(8,4,3)
grunt> result = filter data by c3 ==3;
grunt> dump result;
(1,2,3)
(1,2,3)
(4,3,3)
(8,4,3)
grunt> result2= filter data by (c1==8) or (not (c2+c3>c1));
grunt> dump result2;
(4,2,1)
(8,3,4)
(7,2,5)
(8,4,3)
grunt>
```

GROUP

Agrupa los datos en una o más relaciones.

Ejemplo:

Dada la relación siguiente de estudiantes:

Juan, 30

Maria, 32

Paco, 28

Marta, 30

José, 33

Luis, 36

Crear un fichero llamado group.txt en el directorio ejemplos con el contenido anterior.

`sudo nano ejemplos/group.txt`

```
|bigdata@bigdata:~/pig$ sudo nano ejemplos/group.txt
```

```
Juan, 30
Maria, 32
Paco, 28
Marta, 30
Jose, 33
Luis, 36
```

Comandos que se deben ejecutar en Grunt:

```
estudiantes= load 'ejemplos/group.txt' using PigStorage(',') as (nombre: chararray, edad: int);
```

```
describe estudiantes;
```

```
dump estudiantes;
```

Supóngase, además, que se agrupa la relación en función del campo edad con el alias edad. Como resultado, el segundo campo de la relación edad será de tipo BAG.

```
edad= group estudiantes by edad;
```

```
describe edad;
```

```
dump edad;
```

```
grunt> estudiantes= load 'ejemplos/group.txt' using PigStorage(',') as (nombre: chararray, edad: int);
grunt> describe estudiantes;
estudiantes: {nombre: chararray,edad: int}
grunt> dump estudiantes;
(Juan,30)
(Maria,32)
(Paco,28)
(Marta,30)
(Jose,33)
(Luis,36)
grunt> edad= group estudiantes by edad;
grunt> describe edad;
edad: {group: int,estudiantes: {(nombre: chararray,edad: int)[]}}
grunt> dump edad;
(30,[{(Marta,30),(Juan,30)}])
(32,[{(Maria,32)}])
(33,[{(Jose,33)}])
(36,[{(Luis,36)}])
(28,[{(Paco,28)}])
grunt> █
```

ILLUSTRATE

Comandos que se deben ejecutar en Grunt:

illustrate edad;

```
grunt> illustrate edad;
(Juan, 30)
-----
| estudiantes | nombre:chararray | edad:int |
-----+-----+-----+
|          | Juan           | 30   |
|          | Marta          | 30   |
-----+-----+-----+
| edad      | group:int    | estudiantes:bag{:tuple(nombre:chararray,edad:int)} |
-----+-----+-----+
| 30        |             | {(Juan, 30), (Marta, 30)} |
-----+-----+-----+
grunt> █
```

FOREACH

Genera transformaciones de datos basadas en la información de las columnas.

Utilizando el fichero de alumnos del ejemplo anterior:

ej_foreach = FOREACH edad GENERATE group, COUNT(estudiantes);dump ej_foreach;

```
grunt> ej_foreach = FOREACH edad GENERATE group, COUNT(estudiantes);
grunt> dump ej_foreach;
(30,2)
(32,1)
(33,1)
(36,1)
(28,1)
grunt>
```

ej_foreach2 = FOREACH edad GENERATE \$0, \$1.nombre; dump ej_foreach2;

```
grunt> ej_foreach2 = FOREACH edad GENERATE $0, $1.nombre;
grunt> dump ej_foreach2;
(30,{(Marta),(Juan)})
(32,{(Maria)})
(33,{(Jóse)})
(36,{(Luis)})
(28,{(Paco)})
grunt> █
```



Ejemplo: proyección

En este ejemplo, el asterisco (*) es para proyectar todos los campos de la relación A a la relación B, de modo que al final las dos relaciones son idénticas.

```
dump data;  
  
result = foreach data generate *;  
  
dump result;
```

```
grunt> dump data;  
(1,2,3)  
(4,2,1)  
(8,3,4)  
(4,3,3)  
(7,2,5)  
(8,4,3)  
grunt> result = foreach data generate *;  
grunt> dump result;  
(1,2,3)  
(4,2,1)  
(8,3,4)  
(4,3,3)  
(7,2,5)  
(8,4,3)  
grunt> █
```

campusproyectosnebrija.imf.com ©
JOAO MANUEL DA SILVA FONTES

campusproyectosnebrija.imf.com © EDICIONES ROBLE, S.L.
JOAO MANUEL DA SILVA FONTES COELHO

EDICIONES ROBLE, S.L.
COELHO

JOIN

Une dos o más relaciones empleando campos que tienen en común.

```
data1 = load 'ejemplos/cross1.txt' as (c1:int, c2:int, c3:int);
```

```
data2 = load 'ejemplos/cross2.txt' as (c1:int, c2:int);
```

```
dump data1;
```

```
dump data2;
```

```
result = join data1 by c1, data2 by c1;
```

```
dump result;
```

```
grunt> data1 = load 'ejemplos/cross1.txt' as (c1:int, c2:int, c3:int);
grunt> data2 = load 'ejemplos/cross2.txt' as (c1:int, c2:int);
grunt> dump data1;
(1,2,3)
(4,2,1)
grunt> dump data2;
(2,4)
(8,9)
(1,3)
grunt> result = join data1 by c1, data2 by c1;
grunt> dump result;
(1,2,3,1,3)
grunt>
```

LIMIT

Limita el número de tuplas a la salida.

```
data = load 'ejemplos/filter.txt';
```

```
dump data;
```

```
result = limit data 3;
```

```
dump result;
```

```
grunt> data = load 'ejemplos/filter';
grunt> dump data;
(1,2,3)
(4,2,1)
(8,3,4)
(4,3,3)
(7,2,5)
(8,4,3)
grunt> result = limit data 3;
grunt> dump result;
(1,2,3)
(4,2,1)
(8,3,4)
grunt> 
```

LOAD

Carga datos del sistema de archivos.

```
estudiantes = load 'ejemplos/group.txt' as (nombre:chararray, edad:int);
```

```
describe estudiantes;
```

```
dump estudiantes;
```

```
grunt> estudiantes = load 'ejemplos/group' as (nombre:chararray, edad:int);
grunt> describe estudiantes;
estudiantes: {nombre: chararray,edad: int}
grunt> dump estudiantes;
(Juan,30)
(Maria,32)
(Paco,28)
(Marta,30)
(José,33)
(Luis,36)
```

Por defecto, la función empleada es PigStorage, que recibe un parámetro de entrada que por defecto es un tabulador. Sin embargo, como se ha visto en otros ejemplos es posible emplear otro valor como separador o incluso emplear una función definida por el usuario como función de troceo.

```
a = load 'ejemplos/duplicados.txt' using PigStorage(',') as (col1:int, col2:int, col3:int);
```

```
describe a;
```

```
dump a;
```

```
grunt> a = load 'ejemplos/duplicados.txt' using PigStorage(',') as (col1:int, col2:int, col3:int);
grunt> describe a;
a: {col1: int,col2: int,col3: int}
grunt> dump a;
(8,3,4)
(1,2,3)
(4,3,3)
(4,3,3)
(1,2,3)
grunt>
```

ORDER BY

Sirve para ordenar la relación en función de uno o más campos.

```
a = load 'ejemplos/duplicados.txt' using PigStorage(',') as (col1:int, col2:int, col3:int);
```

```
describe a;
```

```
x = order a by col1 desc;
```

```
dump x;
```

```
grunt> a = load 'ejemplos/duplicados.txt' using PigStorage(',') as (col1:int, col2:int, col3:int);
grunt> describe a;
a: {col1: int,col2: int,col3: int}
grunt> x = order a by col1 desc;
grunt> dump x;
(1,2,3)
(8,3,4)
(1,2,3)
(4,3,3)
(4,3,3)
grunt>
```

SAMPLE

Devolver un tanto por ciento de un fichero, por ejemplo el 20 %.

```
a = load 'ejemplos/duplicados.txt' using PigStorage(',') as (col1:int, col2:int, col3:int);
```

```
veinte_por_ciento = sample a 0.2;
```

```
dump veinte_por_ciento;
```

```
grunt> a = load 'ejemplos/duplicados.txt' using PigStorage(',') as (col1:int, col2:int, col3:int);
grunt> veinte_por_ciento = sample a 0.2;
grunt> dump veinte_por_ciento;
(4,3,3)
(1,2,3)
grunt>
```

Devuelve aproximadamente mil registros de un fichero grande, utilizar el siguiente archivo [enlace](#). (puedes descargar este archivo en el apartado recursos. Título: 3.4.3 SAMPLE)

```
a = load 'ejemplos/fichero_muy_grande.txt';
b= group a all;
c = foreach b generate count(a) as num_rows;
d = sample a 1000 / c.num_rows;
dump d;
```

SPLIT

Parte una relación en una o varias relaciones.

```
a = load 'ejemplos/duplicados.txt' using PigStorage(',') as (col1: int, col2:int, col3:int);
```

```
dump a;
```

```
split a into x if col1>7, y if col2==2, z if (col3<4 or col3>4);
```

```
dump x;
```

```
dump y;
```

```
dump z;
```

```
grunt> split a into x if col1>7, y if col2==2, z if (col3<4 or col3>4);
grunt> dump x;
(8,3,4)
grunt> dump y;
(1,2,3)
(1,2,3)
grunt> dump z;
(1,2,3)
(4,3,3)
(4,3,3)
(1,2,3)
grunt>
```

STORE

Almacena los resultados en un fichero del sistema y puede indicarse el carácter separador.

```
a = load 'ejemplos/duplicados.txt' using PigStorage(',') as (col1: int, col2:int, col3:int);
store a into 'ejemplos/cambiar_separador.txt' using PigStorage ('*');
cat ejemplos/cambiar_separador.txt;
```

```
grunt> a = load 'ejemplos/duplicados.txt' using PigStorage(' ') as (col1: int, col2:int, col3:int);
grunt> store a into 'ejemplos/cambiar_separador.txt' using PigStorage ('*');
grunt> cat ejemplos/cambiar_separador.txt;
8*3*4
1*2*3
4*3*3
4*3*3
1*2*3
grunt>
```

```
a = load 'ejemplos/duplicados.txt' using PigStorage(',') as (col1: int, col2:int, col3:int);
```

```
b = foreach a generate CONCAT ('col1:', (chararray)col1), CONCAT('col2:', (chararray) col2),
CONCAT('col3:', (chararray) col3);
```

```
store b into 'ejemplos/columnas.txt' using PigStorage(',');
cat ejemplos/columnas.txt;
```

```
grunt> a = load 'ejemplos/duplicados.txt' using PigStorage(',') as (col1: int, col2:int, col3:int);
grunt> b = foreach a generate CONCAT ('col1:', (chararray)col1), CONCAT('col2:', (chararray) col2), CONCAT('col3:', (chararray) col3);
grunt> store b into 'ejemplos/columnas.txt' using PigStorage(',');
grunt> cat ejemplos/columnas.txt;
col1:8,col2:3,col3:4
col1:1,col2:2,col3:3
col1:4,col2:3,col3:3
col1:4,col2:3,col3:3
col1:1,col2:2,col3:3
grunt>
```

UNION

Permite unir dos o más relaciones.

```
a = load 'ejemplos/cross1.txt' as (a1:int, a2::int, b3:int);
```

```
b = load 'ejemplos/cross2.txt' as (b1:int, b2:int);
```

```
dump a;
```

```
dump b;
```

```
result = union a,b;
```

```
dump result;
```

```
grunt> a = load 'ejemplos/cross1.txt' as (a1:int, a2::int, b3:int);
grunt> b = load 'ejemplos/cross2.txt' as (b1:int, b2:int);
grunt> dump a;
(1,2,3)
(4,2,1)
grunt> dump b;
(2,4)
(8,9)
(1,3)
grunt> result = union a,b;
grunt> dump result;
(1,2,3)
(4,2,1)
(2,4)
(8,9)
(1,3)
grunt>
```

3.4.4. UDF (*user defined functions*)

Gran parte de la potencia de PIG reside en la posibilidad de permitir a los usuarios combinar los comandos propios de PIG con los definidos por el usuario (*user defined functions*: UDF). Hasta la versión 0.7 solo se permitía emplear Java para crear UDF. Desde la versión 0.8 las UDF pueden estar escritas en seis lenguajes: Java, Jython, Python, JavaScript, Ruby y Groovy. Las funciones Java son más eficientes y cuentan con mayor soporte ya que están en el mismo lenguaje que el propio PIG.

En el caso del resto de lenguajes, solo existe soporte para algunas funcionalidades (por ejemplo, no están soportadas las funciones de carga y almacenamiento). Además, Javascript, Ruby y Groovy son características experimentales que no han tenido el mismo número de pruebas que Java y Jython.

En tiempo de ejecución, PIG detecta automáticamente el uso de UDF y ejecuta automáticamente el correspondiente jar en Jython, Rhino, JRuby o Groovy-all. Python no requiere ningún motor de ejecución ya que se invoca directamente el comando Python.

El propio PIG incluye un paquete con algunas UDF. Hasta la versión 0.8 era un conjunto limitado que incluía una serie de funciones de agregación y algunas otras, pero desde la versión 0.8 hay un gran número de funciones de procesamiento de cadenas, matemáticas y procesamiento de tipos complejos. Este conjunto de funciones se encuentra dentro de Piggy Bank, que contiene todas aquellas UDF definidas por otros usuarios. Estas funciones deben registrarse de forma manual al no incluirse en el jar de PIG.

A continuación, se explica cómo se pueden definir, registrar e invocar este tipo de funciones.

DEFINE

Cuando se quiere emplear una función que ha sido construida como UDF, es necesario indicar a PIG dónde se encuentra el fichero con el UDF. Esto se hace mediante el comando DEFINE que asigna un alias a una UDF.

Sintaxis: *DEFINE alias {function | [`command` [input] [output] [ship] [cache] [stderr]]};*

REGISTER

Registra un archivo jar para que las UDF incluidas en el jar puedan utilizarse.

Sintaxis: *REGISTER path;*

REGISTER /src/myfunc.js;
A = LOAD 'students';
B = FOREACH A GENERATE myfunc.MyEvalFunc(\$0);

En este ejemplo se registran varios jar con la variable de entorno PIG_OPTS.

```
export PIG_OPTS="-Dpig.additional.jars.uris=hdfs://nn.mydomain.com:9020/myjars/my.jar,file:///home/bigdata/pig/your.jar"
```

EJEMPLO CON PIGGYBANK

A continuación se muestra un ejemplo sencillo de uso de librerías existentes. El ejemplo seleccionado es el jar org.apache.pig.piggybank.evaluation.string, cuya documentación queda disponible en la [API](#).

Herramientas del ecosistema Hadoop

The screenshot shows the JavaDoc interface for the Pig 0.15.0 API. At the top, there's a navigation bar with tabs for Overview, Package, Class, Use, Tree, Deprecated, Index, and Help. Below the navigation bar, there are links for Prev, Next, Frames, and No Frames. The main content area is titled "Pig 0.15.0 API". It contains a brief description: "Pig is a platform for a data flow programming on large data sets in a parallel environment." and a "See: Description" link. A search bar labeled "pig" is present. The left sidebar lists various packages and classes under "All Classes". The right sidebar lists the "Description" for the org.apache.pig package, which includes a detailed list of sub-packages and their descriptions.

En el ejemplo se emplea el método UPPER del paquete `piggybank.evaluation.string`, que transforma el texto a mayúsculas. En primer lugar se registra el jar que contiene las librerías, posteriormente se carga el fichero que contiene la discografía y, finalmente, por cada registro del fichero cargado se genera una salida que contiene el campo anio y el campo disco en mayúsculas.

The screenshot shows the JavaDoc for the `UPPER` class. The navigation bar at the top has the "Class" tab selected. The class summary section includes the package name (`org.apache.pig.piggybank.evaluation.string`), the class name (`UPPER`), its superclass (`java.lang.Object`), and its interface (`org.apache.pig.EvalFunc<String>`). It also notes that the class is deprecated. The "Nested Class Summary" section lists the `EvalFunc.SchemaType` nested class. The "Field Summary" section lists the fields `log`, `pigLogger`, and `reporter`.

1

1. Registrar librería:

`REGISTER /home/bigdata/pig/lib/piggybank.jar`

```
grunt> REGISTER /home/bigdata/pig/lib/piggybank.jar
```

2

2. Cargar el fichero que contiene la discografía:

A = LOAD 'ejemplos/discografia' using PigStorage(',') AS (anio: int, disco: chararray);
dump A;

```
grunt> A = LOAD 'ejemplos/discografia' using PigStorage(',') AS (anio: int, disco: chararray);
grunt> dump A;
(1967,The Piper at the Gates of Dawn)
(1968,A Saucerful of Secrets)
(1969,Music from the Film More)
(1969,Ummagumma)
(1970,Atom Heart Mother)
(1972,Obscured by Clouds)
(1973,The Dark Side of the Moon)
(1975,Wish you Were Here)
(1977,Animals)
(1979,The Wall)
(1983,The Final Cut)
(1987,A Momentary Lapse of Reason)
(1994,The Division Bell)
(2014,The Endless River)
grunt>
```

3

3. Por cada registro del fichero cargado, generar una salida con el campo anio y el campo disco en mayúsculas:

B = FOREACH A GENERATE anio, org.apache.pig.piggybank.evaluation.string.UPPER
(disco);
dump B;

```
grunt> B = FOREACH A GENERATE anio, org.apache.pig.piggybank.evaluation.string.UPPER (disco);
grunt> dump B;
(1967,THE PIPER AT THE GATES OF DAWN)
(1968,A SAUCERFUL OF SECRETS)
(1969,MUSIC FROM THE FILM MORE)
(1969,UMMAGUMMA)
(1970,ATOM HEART MOTHER)
(1972,OBSCURED BY CLOUDS)
(1973,THE DARK SIDE OF THE MOON)
(1975,WISH YOU WERE HERE)
(1977,ANIMALS)
(1979,THE WALL)
(1983,THE FINAL CUT)
(1987,A MOMENTARY LAPSE OF REASON)
(1994,THE DIVISION BELL)
(2014,THE ENDLESS RIVER)
grunt> █
```

EJEMPLO DE UDF

Partiendo del siguiente ejemplo de fichero, crear una función Java que transforme un texto a mayúsculas.

```
$ cat ejemplos/discografia
```

```
bigdata@bigdata:~/pig$ cat ejemplos/discografia
1967,The Piper at the Gates of Dawn,131,6
1968,A Saucerful of Secrets,999,9
1969,Music from the Film More,153,9
1969,Ummagumma,74,5
1970,Atom Heart Mother,55,1
1972,Obscured by Clouds,46,6
1973,The Dark Side of the Moon,1,1
1975,Wish you Were Here,1,1
1977,Animals,3,2
1979,The Wall,1,3
1983,The Final Cut,6,1
1987,A Momentary Lapse of Reason,3,3
1994,The Division Bell,1,1
2014,The Endless River,3,1
```

campusproyectosnebrija.imf.com © EDICIONES ROBLE, S.L.
JOAO MANUEL DA SILVA FONTES COELHO

campusproyectosnebrija.imf.com © EDICIONES ROBLE, S.L.
JOAO MANUEL DA SILVA FONTES COELHO

EDICIONES ROBLE, S.L.
JOAO MANUEL DA SILVA FONTES COELHO

1

1. Crear estructura de directorios y fichero .java:

```
sudo mkdir ejemplos/udfs
```

```
sudo mkdir ejemplos/udfs/com
```

```
sudo mkdir ejemplos/udfs/com/master
```

```
sudo nano ejemplos/udfs/com/master/UPPER.java
```

```
bigdata@bigdata:~/pig$ sudo mkdir ejemplos/udfs  
[sudo] password for bigdata:  
bigdata@bigdata:~/pig$ sudo mkdir ejemplos/udfs/com  
bigdata@bigdata:~/pig$ sudo mkdir ejemplos/udfs/com/master  
bigdata@bigdata:~/pig$ sudo nano ejemplos/udfs/com/master/UPPER.java  
bigdata@bigdata:~/pig$
```

Para ver el código que necesitas, pulsa [aquí](#).

```
bigdata@bigdata: ~/pig  
GNU nano 2.2.6          Archivo: ejemplos/udfs/com/master/UPPER.java  
  
package com.master;  
import java.io.IOException;  
import org.apache.pig.EvalFunc;  
import org.apache.pig.data.Tuple;  
  
public class UPPER extends EvalFunc<String>{  
    public String exec(Tuple input) throws IOException {  
        if (input == null || input.size() == 0 || input.get(0) == null)  
            return null;  
        try{  
            String str = (String)input.get(0);  
            return str.toUpperCase();  
        }catch(Exception e){  
            throw new IOException("Caught exception processing input row ", e);  
        }  
    }  
}
```

2

2. Compilar:

```
cd ejemplos/udfs/
```

```
sudo javac -classpath $HADOOP_HOME/share/hadoop/common/hadoop-common-2.8.0.jar:$PIG_HOME/pig-0.16.0-core-h1.jar com/master/UPPER.java
```

```
sudo jar cf /home/bigdata/pig/ejemplos/udfs/UPPER.jar com/master/*.class
```

```
ls
```

```
bigdata@bigdata:~/pig$ cd ejemplos/udfs/
bigdata@bigdata:~/pig/ejemplos/udfs$ sudo javac -classpath $HADOOP_HOME/share/hadoop/common/hadoop-common-2.8.0.jar:$PIG_HOME/pig-0.16.0-core-h1.jar com/master/UPPER.java
bigdata@bigdata:~/pig/ejemplos/udfs$ sudo jar cf /home/bigdata/pig/ejemplos/udfs/UPPER.jar com/master/*.class
bigdata@bigdata:~/pig/ejemplos/udfs$ ls
com_UPPER.jar
```

3

3. Crear fichero .pig:

```
cd /home/bigdata/pig/ejemplos
```

```
sudo nano ejemplos/script_ejemplo.pig
```

```
bigdata@bigdata:~/pig$ sudo nano ejemplos/script_ejemplo.pig
```

```
REGISTER /home/bigdata/pig/ejemplos/udfs/UPPER.jar
```

```
A= LOAD 'ejemplos/discografia' using PigStorage(',') as (anio: int, disco: chararray);
```

```
B= FOREACH A GENERATE com.master.UPPER(disco);
DUMP B;
```

```
bigdata@bigdata: ~pig
GNU nano 2.2.6          Archivo: ejemplos/script_ejemplo.pig
REGISTER /home/bigdata/pig/ejemplos/udfs/UPPER.jar
A= LOAD 'ejemplos/discografia' using PigStorage(',') as (anio: int, disco: chararray);
B= FOREACH A GENERATE com.master.UPPER(disco);
DUMP B;
```

4

4. Ejecutar el *script*:

```
pig -4 conf/nolog.conf -x local ejemplos/script_ejemplo.pig
```

```
bigdata@bigdata:~/pig$ sudo nano ejemplos/script_ejemplo.pig
bigdata@bigdata:~/pig$ pig -4 conf/nolog.conf -x local ejemplos/script_ejemplo.pig
17/08/27 17:23:20 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
17/08/27 17:23:20 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
17/08/27 17:23:20 WARN pig.Main: Cannot write to log file: /home/bigdata/pig/pig_1503847400283.log
17/08/27 17:23:20 INFO pig.Main: Loaded log4j properties from file: conf/nolog.conf
(THE PIPER AT THE GATES OF DAWN)
(A SAUCERFUL OF SECRETS)
(MUSIC FROM THE FILM MORE)
(UMMAGUMMA)
(ATOM HEART MOTHER)
(OBSCURED BY CLOUDS)
(THE DARK SIDE OF THE MOON)
(WISH YOU WERE HERE)
(ANIMALS)
(THE WALL)
(THE FINAL CUT)
(A MOMENTARY LAPSE OF REASON)
(THE DIVISION BELL)
(THE ENDLESS RIVER)
bigdata@bigdata:~/pig$
```

3.5. Pig Latin avanzado

Además de los comandos básicos, los jar de Piggybank y las funciones que pueda definir el usuario, PIG facilita una serie de funciones para evaluar, cargar y almacenar datos, así como una serie de funciones matemáticas, de cadenas y de manejo de estructuras más complejas como bags y tuples.



A continuación solo se enumeran algunas de ellas; no obstante, se puede encontrar más información la [página de documentación de Hadoop para Pig 0.16.0](#)

3.5.1. Algunas funciones

Funciones de evaluación

- AVG.
- BagToString.
- CONCAT.
- COUNT.
- COUNT_STAR.
- DIFF.
- IsEmpty.
- MAX.
- MIN.
- PluckTuple.
- SIZE.
- SUBTRACT.
- SUM.
- TOKENIZE.

Funciones de carga y almacenamiento

- Handling Compression.
- BinStorage.
- JsonLoader, JsonStorage.
- PigDump.
- PigStorage.
- TextLoader.
- HBaseStorage.
- AvroStorage.
- TrevniStorage.
- AccumuloStorage.
- OrcStorage.

Funciones matemáticas

- ABS.
- ACOS.
- ASIN.
- ATAN.
- CBRT.
- CEIL.
- COS.
- COSH.
- EXP.
- FLOOR.
- LOG.

- LOG10.
- RANDOM.
- ROUND.
- ROUND_TO.
- SIN.
- SINH.
- SQRT.
- TAN.
- TANH.

Funciones de manejo de cadenas

- ENDSWITH.
- EqualsIgnoreCase.
- INDEXOF.
- LAST_INDEX_OF.
- LCFIRST.
- LOWER.
- LTRIM.
- REGEX_EXTRACT.
- REGEX_EXTRACT_ALL.
- REPLACE.
- RTRIM.
- SPRINTF.
- STARTSWITH.
- STRSPLIT.
- STRSPLITTOBAG.
- SUBSTRING.
- TRIM.
- UCFIRST.
- UPPER.
- UniqueID.

Funciones de procesamiento de fechas

- AddDuration.
- CurrentTime.
- DaysBetween.
- GetDay.
- GetHour.
- GetMillisecond.
- GetMinute.
- GetMonth.
- GetSecond.

- GetWeek.
- GetWeekYear.
- GetYear.
- HoursBetween.
- MilliSecondsBetween.
- MinutesBetween.
- MonthsBetween.
- SecondsBetween.
- SubtractDuration.
- ToDate.
- ToMilliSeconds.
- ToString.
- ToUnixTime.
- WeeksBetween.
- YearsBetween.

Funciones de manejo de estructuras complejas

- TOTUPLE.
- TOBAG.
- TOMAP.
- TOP.

3.5.2. Ejemplo

En este epígrafe, se va a trabajar sobre un ejemplo práctico:



Dados los resultados de la liga 2014-2015, obtener el promedio de goles marcados en local por el Real Madrid o el Barcelona: se pueden descargar los datos en este [enlace](#).

1

1. Crear fichero con los datos que vamos a analizar:

sudo nano ejemplos/liga

```
GNU nano 2.7.4                               Archivo: ejemplos/liga

25/08/2014,J1,Real Madrid,Córdoba,2,0
31/08/2014,J2,R.Sociedad,Real Madrid,4,2
13/09/2014,J3,Real Madrid,Atlético,1,2
20/09/2014,J4,Deportivo,Real Madrid,2,8
23/09/2014,J5,Real Madrid,Elche,5,1
27/09/2014,J6,Villareal,Real Madrid,0,2
05/10/2014,J7,Real Madrid,Athletic,5,0
18/10/2014,J8,Levante,Real Madrid,0,5
25/10/2014,J9,Real Madrid,Barcelona,3,1
01/11/2014,J10,Granada,Real Madrid,0,4
08/11/2014,J11,Real Madrid,Rayo Vallecano,5,1
22/11/2014,J12,Eibar,Real Madrid,0,4
29/11/2014,J13,Málaga,Real Madrid,1,2
06/12/2014,J14,Real Madrid,Celta,3,0
12/12/2014,J15,Almería,Real Madrid,1,4
04/02/2015,J16,Real Madrid,Sevilla,2,1
04/01/2015,J17,Valencia,Real Madrid,2,1
10/01/2015,J18,Real Madrid,Espanyol,3,0
18/01/2015,J19,Getafe,Real Madrid,0,3
20/01/2015,J20,Córdoba,Real Madrid,1,2
31/01/2015,J21,Real Madrid,R.Sociedad,4,1
07/02/2015,J22,Atlético,Real Madrid,4,0
14/02/2015,J23,Real Madrid,Deportivo,2,0
22/02/2015,J24,Elche,Real Madrid,0,2
01/03/2015,J25,Real Madrid,Villareal,1,1
07/03/2015,J26,Athletic,Real Madrid,1,0
15/03/2015,J27,Real Madrid,Levante,2,0
22/03/2015,J28,Barcelona,Real Madrid,2,1
05/04/2015,J29,Real Madrid,Granada,9,1
08/04/2015,J30,Rayo,Real Madrid,0,2
11/04/2015,J31,Real Madrid,Eibar,3,0
18/04/2015,J32,Real Madrid,Málaga,3,1
26/04/2015,J33,Celta,Real Madrid,2,4
29/04/2015,J34,Real Madrid,Almería,3,0
02/05/2015,J35,Sevilla,Real Madrid,2,3
09/05/2015,J36,Real Madrid,Valencia,2,2
```

2

2. Cargar el fichero y ejecutar la instrucción de consulta para ver los datos:

```
a = LOAD 'ejemplos/liga' using PigStorage(',') AS (fecha: chararray, jornada: chararray, local:chararray, invitado:chararray, gol_local:int, gol_invitado: int);
```

```
b= FOREACH a GENERATEToDate (fecha, 'dd/MM/yyyy') as fecha, jornada, local, invitado, gol_local, gol_invitado;
```

```
aux = limit b 5;
```

```
dump aux;
```

```
grunt> a = LOAD 'ejemplos/liga' using PigStorage(',') AS (fecha: chararray, jornada: chararray, local:chararray, invitado:chararray, gol_local:int, gol_invitado: int);
grunt> b= FOREACH a GENERATEToDate (fecha, 'dd/MM/yyyy') as fecha, jornada, local, invitado, gol_local, gol_invitado;
grunt> aux = limit b 5;
grunt> dump aux;
(2014-08-25T00:00:00.000+02:00,J1,Real Madrid,Córdoba,2,0)
(2014-08-31T00:00:00.000+02:00,J2,R.Sociedad,Real Madrid,4,2)
(2014-09-13T00:00:00.000+02:00,J3,Real Madrid,Atlético,1,2)
(2014-09-20T00:00:00.000+02:00,J4,Deportivo,Real Madrid,2,8)
(2014-09-23T00:00:00.000+02:00,J5,Real Madrid,Elche,5,1)
grunt>
```

```
grunt> a = LOAD 'ejemplos/liga.txt' using PigStorage(',') AS (fecha: chararray, jornada: chararray, local:chararray, invitado:chararray, gol_local:int, gol_invitado: int);
grunt> b= FOREACH a GENERATEToDate (fecha, 'dd/MM/yyyy') as fecha, jornada, local, invitado, gol_local, gol_invitado;
grunt> aux = limit b 5;
grunt> dump aux;
```

3

3. Ejecutar filtrados sobre los datos anteriores:

```
c = filter b by (local == 'Real Madrid' or local =='Barcelona');
dump c;
```

```

grunt> c = filter b by (local == 'Real Madrid' or local =='Barcelona');
grunt> dump c;
(2014-08-25T00:00:00.000+02:00,J1,Real Madrid,Córdoba,2,0)
(2014-09-13T00:00:00.000+02:00,J3,Real Madrid,Atlético,1,2)
(2014-09-23T00:00:00.000+02:00,J5,Real Madrid,Elche,5,1)
(2014-10-05T00:00:00.000+02:00,J7,Real Madrid,Athletic,5,0)
(2014-10-25T00:00:00.000+02:00,J9,Real Madrid,Barcelona,3,1)
(2014-11-08T00:00:00.000+01:00,J11,Real Madrid,Rayo Vallecano,5,1)
(2014-12-06T00:00:00.000+01:00,J14,Real Madrid,Celta,3,0)
(2015-02-04T00:00:00.000+01:00,J16,Real Madrid,Sevilla,2,1)
(2015-01-10T00:00:00.000+01:00,J18,Real Madrid,Espanyol,3,0)
(2015-01-31T00:00:00.000+01:00,J21,Real Madrid,R.Sociedad,4,1)
(2015-02-14T00:00:00.000+01:00,J23,Real Madrid,Deportivo,2,0)
(2015-03-01T00:00:00.000+01:00,J25,Real Madrid,Villareal,1,1)
(2015-03-15T00:00:00.000+01:00,J27,Real Madrid,Levante,2,0)
(2015-03-22T00:00:00.000+01:00,J28,Barcelona,Real Madrid,2,1)
(2015-04-05T00:00:00.000+02:00,J29,Real Madrid,Granada,9,1)
(2015-04-11T00:00:00.000+02:00,J31,Real Madrid,Eibar,3,0)
(2015-04-18T00:00:00.000+02:00,J32,Real Madrid,Málaga,3,1)
(2015-04-29T00:00:00.000+02:00,J34,Real Madrid,Almería,3,0)
(2015-05-09T00:00:00.000+02:00,J36,Real Madrid,Valencia,2,2)
(2015-05-23T00:00:00.000+02:00,J38,Real Madrid,Getafe,7,3)
(2014-08-24T00:00:00.000+02:00,J1,Barcelona,Elche,3,0)
(2014-09-13T00:00:00.000+02:00,J3,Barcelona,Athletic,2,0)
(2014-09-27T00:00:00.000+02:00,J6,Barcelona,Granada,6,0)
(2014-10-18T00:00:00.000+02:00,J8,Barcelona,Eibar,3,0)
(2014-11-01T00:00:00.000+01:00,J10,Barcelona,Celta,0,1)
(2014-11-22T00:00:00.000+01:00,J12,Barcelona,Sevilla,5,1)
(2014-12-07T00:00:00.000+01:00,J14,Barcelona,Espanyol,5,1)
(2014-12-20T00:00:00.000+01:00,J16,Barcelona,Córdoba,5,0)
(2015-01-11T00:00:00.000+01:00,J18,Barcelona,Atlético,3,1)
(2015-02-01T00:00:00.000+01:00,J21,Barcelona,Villareal,3,2)
(2015-02-15T00:00:00.000+01:00,J23,Barcelona,Levante,5,0)
(2015-02-21T00:00:00.000+01:00,J24,Barcelona,Málaga,0,1)
(2015-03-08T00:00:00.000+01:00,J26,Barcelona,Rayo,6,1)
(2015-04-08T00:00:00.000+02:00,J30,Barcelona,Almería,4,0)
(2015-04-18T00:00:00.000+02:00,J32,Barcelona,Valencia,2,0)
(2015-04-28T00:00:00.000+02:00,J34,Barcelona,Getafe,6,0)
(2015-05-09T00:00:00.000+02:00,J36,Barcelona,R.Sociedad,2,0)
(2015-05-23T00:00:00.000+02:00,J38,Barcelona,Deportivo,2,2)
grunt>
```

```

c2 = filter b by (jornada matches 'J3.*' and local =='Barcelona');
dump c2;
```

```

grunt> c2 = filter b by (jornada matches 'J3.*' and local =='Barcelona');
grunt> dump c2;
(2014-09-13T00:00:00.000+02:00,J3,Barcelona,Athletic,2,0)
(2015-04-08T00:00:00.000+02:00,J30,Barcelona,Almería,4,0)
(2015-04-18T00:00:00.000+02:00,J32,Barcelona,Valencia,2,0)
(2015-04-28T00:00:00.000+02:00,J34,Barcelona,Getafe,6,0)
(2015-05-09T00:00:00.000+02:00,J36,Barcelona,R.Sociedad,2,0)
(2015-05-23T00:00:00.000+02:00,J38,Barcelona,Deportivo,2,2)
```

4

4. Emplear una función de agrupación:

d = group c by local;

dump d;

```
grunt> d = group c by local;
grunt> dump d;
(Barcelona,{(2015-05-23T00:00:00.000+02:00,J38,Barcelona,Deportivo,2,2),(2015-04-28T00:00:00.000+02:00,J34,Barcelona,Getafe,6,0),(2015-04-18T00:00:00.000+02:00,J32,Barcelona,Valencia,2,0),(2015-04-08T00:00:00.000+02:00,J30,Barcelona,Almeria,4,0),(2015-03-08T00:00:00.000+01:00,J26,Barcelona,Rayo,6,1),(2015-03-22T00:00:00.000+01:00,J28,Barcelona,Real Madrid,2,1),(2015-02-21T00:00:00.000+01:00,J24,Barcelona,Málaga,0,1),(2015-02-15T00:00:00.000+01:00,J23,Barcelona,Levante,5,0),(2015-02-01T00:00:00.000+01:00,J21,Barcelona,Villareal,3,2),(2015-01-11T00:00:00.000+01:00,J18,Barcelona,Atlético,3,1),(2014-12-20T00:00:00.000+01:00,J16,Barcelona,Córdoba,5,0),(2014-12-07T00:00:00.000+01:00,J14,Barcelona,Espanyol,5,1),(2014-11-22T00:00:00.000+01:00,J12,Barcelona,Sevilla,5,1),(2014-11-01T00:00:00.000+01:00,J10,Barcelona,Celta,0,1),(2014-10-18T00:00:00.000+02:00,J8,Barcelona,Eibar,3,0),(2014-09-27T00:00:00.000+02:00,J6,Barcelona,Granada,6,0),(2014-09-13T00:00:00.000+02:00,J3,Barcelona,Athletic,2,0),(2014-08-24T00:00:00.000+02:00,J1,Barcelona,Elche,3,0),(2015-05-09T00:00:00.000+02:00,J36,Barcelona,R.Sociedad,2,0)})
(Real Madrid,{(2015-05-09T00:00:00.000+02:00,J36,Real Madrid,Valencia,2,2),(2015-05-23T00:00:00.000+02:00,J38,Real Madrid,Getafe,7,3),(2015-04-29T00:00:00.000+02:00,J34,Real Madrid,Almeria,3,0),(2015-04-18T00:00:00.000+02:00,J32,Real Madrid,Málaga,3,1),(2015-04-11T00:00:00.000+02:00,J31,Real Madrid,Eibar,3,0),(2015-04-05T00:00:00.000+02:00,J29,Real Madrid,Granada,9,1),(2015-03-15T00:00:00.000+01:00,J27,Real Madrid,Levante,2,0),(2015-03-01T00:00:00.000+01:00,J25,Real Madrid,Villareal,1,1),(2015-02-14T00:00:00.000+01:00,J23,Real Madrid,Deportivo,2,0),(2015-01-31T00:00:00.000+01:00,J21,Real Madrid,R.Sociedad,4,1),(2015-01-10T00:00:00.000+01:00,J18,Real Madrid,Espanyol,3,0),(2015-02-04T00:00:00.000+01:00,J16,Real Madrid,Sevilla,2,1),(2014-12-06T00:00:00.000+01:00,J14,Real Madrid,Celta,3,0),(2014-11-08T00:00:00.000+01:00,J11,Real Madrid,Rayo Vallecano,5,1),(2014-10-25T00:00:00.000+02:00,J9,Real Madrid,Barcelona,3,1),(2014-10-05T00:00:00.000+02:00,J7,Real Madrid,Athletic,5,0),(2014-09-23T00:00:00.000+02:00,J5,Real Madrid,Elche,5,1),(2014-09-13T00:00:00.000+02:00,J3,Real Madrid,Atlético,1,2),(2014-08-25T00:00:00.000+02:00,J1,Real Madrid,Córdoba,2,0)})
grunt> ■
```

5

5. Calcular la media:

e = foreach d generate group, AVG (c.gol_local);

dump e;

```
grunt> e = foreach d generate group, AVG (c.gol_local);
grunt> dump e;
(Barcelona,3.3684210526315788)
(Real Madrid,3.4210526315789473)
grunt> ■
```

3.5.3. Ejemplo de análisis de logs

Este ejemplo se basará en la estructura de los logs de un servidor web apache.



Para ampliar información, se puede consultar la [página de la comunidad](#).

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 232
```

- 127.0.0.1 (%h): IP del cliente (host remoto) que ha realizado la petición.
- (%l): el guion indica que esta información no está disponible. Esta información es poco fiable y no se debería emplear.
- frank (%u): Identifica el ID del usuario que está realizando la petición del documento determinado en una petición HTTP autenticada. Si el *estatus code* (más abajo) de la petición es 401, entonces este valor no debe tenerse en cuenta.
- [10/Oct/2000:13:55:36 -0700] (%t): la fecha en la que se recibió la petición en el formato siguiente: [day/month/year:hour:minute:second zone]
 - day = 2*digit
 - month = 3*letter
 - year = 4*digit
 - hour = 2*digit
 - minute = 2*digit
 - second = 2*digit
 - zone = ('+' | '-') 4*digit
- "GET /apache_pb.gif HTTP/1.0" ("%"r"\"): la línea pedida por el cliente aparecerá entre comillas dobles. En primer lugar indicará el tipo de cliente empleado (GET/POST). A continuación el recurso solicitado y en tercer lugar el protocolo empleado.
- 200 (%>s): contiene el *estatus code* devuelto por el servidor al cliente.
 - 2XX: éxito.
 - 3XX: redirección.
 - 4XX: error causado por el cliente.
 - 5XX: error causado por el servidor.
 - 2326 (%b): tamaño del objeto devuelto sin incluir las cabeceras.

```
sudo mkdir /home/bigdata/ejemplosPig
```

```
sudo nano /home/bigdata/ejemplosPig/analisisLogsApache.pig
```

```
bigdata@bigdata:~/pig$ sudo mkdir /home/bigdata/ejemplosPig
bigdata@bigdata:~/pig$ sudo nano /home/bigdata/ejemplosPig/analisisLogsApache.pig
```



-- Ejemplo de llamada: pig -x mapreduce -f analisisLogsApache.pig -param LOGS='path_to_access_log_file.txt' --param OUTFILE='path_to_output_file'

-- El formato del fichero que estamos cargando es el siguiente:

-- <http://httpd.apache.org/docs/current/logs.html>

-- 127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326

-- 127.0.0.1 (%h): IP del cliente (host remoto) que ha realizado la petición

-- (%l) : el guion indica que esta información no está disponible. Esta información es poco fiable y no se debería emplear

-- frank (%u): Identifica el ID del usuario que está realizando la petición del documento determinado en una petición HTTP autenticada. Si el estatus code (más abajo) de la petición es 401, entonces este valor no debe tenerse en cuenta.

-- [10/Oct/2000:13:55:36 -0700] (%t): la fecha a la que se recibió la petición en el formato siguiente:

-- [day/month/year:hour:minute:second zone]

-- day = 2*digit

-- month = 3*letter

-- year = 4*digit

-- hour = 2*digit

-- minute = 2*digit

-- second = 2*digit

-- zone = ('+' | '-') 4*digit

-- "GET /apache_pb.gif HTTP/1.0" ("%r"): La línea pedida por el cliente aparecerá entre comillas dobles. En primer lugar indicará el tipo de cliente empleado (GET/POST). A continuación el recurso solicitado y en tercer lugar el protocolo empleado

-- 200 (%>s): contiene el estatus code devuelto por el servidor al cliente.

-- 2XX: éxito

-- 3XX: redirección

-- 4XX: error causado por el cliente

-- 5XX: error causado por el servidor

-- 2326 (%b): tamaño del objeto devuelto sin incluir las cabeceras.

-- 1. Registramos la librería

REGISTER /home/bigdata/pig/lib/piggybank.jar

-- Definimos la librería CommonLogLoader que vamos a emplear

```
DEFINE ApacheCommonLogLoader
org.apache.pig.piggybank.storage.apachelog.CommonLogLoader();
```

-- cargamos el fichero de logs indicado en la entrada al proceso

```

logs = LOAD '$LOGS' USING ApacheCommonLogLoader as (remoteHost, hyphen, user,
time, method, uri, protocol, statusCode, responseSize);

-- filtramos la entrada para quedarnos solo con las peticiones GET

logs = FILTER logs BY method == 'GET'

AND statusCode >= 200 AND statusCode < 300;

-- Proyectamos solo los campos que nos interesan

logs = FOREACH logs GENERATE uri, responseSize;

-- Agrupamos por URI y contamos el número de peticiones

groupedByUri = GROUP logs BY uri;

uriCounts = FOREACH groupedByUri GENERATE

group AS uri, COUNT(logs) AS numHits;

-- Ordenamos el resultado

uri_result = ORDER uriCounts BY uri DESC;

-- DUMP uri_result;

DESCRIBE uri_result;

-- Cargamos los datos a la salida

STORE uri_result INTO '$OUTFILE';

```

```

GNU nano 2.7.4                               Archivo: /home/bigdata/ejemplosPig/analisisLogsApache.pig                                Modificado: ■
.: Ejemplo de llanado: pig -x mapreduce -f analisisLogsApache.pig -param LOGS='path_to_access_log_file.txt' --param OUTFILE='path_to_output_file'
.: El formato del fichero que estamos cargando es el siguiente:
.: http://httpd.apache.org/docs/current/logs.html
.: El formato de la cabecera de la petición es el siguiente: "GET /apache_pb.gif HTTP/1.0" 200 2326
.: 127.0.0.1 ("h") IP del cliente (host remoto) que ha realizado la petición
.: (%) : el guión indica que esta información no está disponible. Esta información es poco fiable y no se debería emplear
.: La cabecera de fecha y hora indica que está realizando la petición del documento determinado en una petición HTTP autenticada. Si el estatus code (más abajo) de la petición es 401, entonces $[day/month/year:hour:minute:second zone]
.: year = 4#digit
.: month = 3#letter
.: month = 3#letter
.: year = 4#digit
.: month = 3#letter
.: minuate = 2#digit
.: second = 2#digit
.: "GET /apache_pb.gif HTTP/1.0" 4#digit
.: "GET /apache_pb.gif HTTP/1.0" ("%"": La linea pedida por el cliente aparecerá entre comillas dobles. En primer lugar indicará el tipo de cliente empleado (GET/POST). A continuación el recurso soñado
.: 200 (%$): contiene el estatus code devuelto por el servidor al cliente.
.: 301 (%$): redirección
.: 302 (%$): redirección
.: 3XX: error causado por el cliente
.: 4XX: error causado por el servidor
.: 5XX: error causado por el servidor
.: 2326 (%$): tenido del objeto devuelto sin incluir las cabeceras.
.: Registrando la librería /piggybank.jar
REGISTER file:///piggybank.jar
-- Definimos la librería CommonLogLoader que vamos a emplear
DEFINE ApacheCommonLogLoader org.apache.pig.piggybank.storage.ApacheLog.CommonLogLoader();
-- Cargamos los datos de los logs en un formato que podremos procesar
logs = LOAD '$LOGS' USING ApacheCommonLogLoader as (remoteHost, hyphen, user, time, method, uri, protocol, statusCode, responseSize);
-- filtramos la entrada para quedarnos solo con las peticiones GET
logs = FILTER logs BY method == 'GET';
-- AND statusCode >= 200 AND statusCode < 300;
-- Proyectamos solo los campos que nos interesan
logs = FOREACH logs GENERATE
-- Agrupamos por URI y contamos el numero de peticiones
groupedbyuri = GROUP logs BY uri;
uriCounts = FOREACH groupedbyuri GENERATE
group AS uri, COUNT(logs) AS numHits;
-- Ordenamos el resultado
uri_result = ORDER uriCounts BY numHits DESC;
-- DUMP uri_result;
DESCRIBE uri_result;
-- Cargamos los datos a la salida
STORE uri_result INTO '$OUTFILE';

```

sudo nano ejemplosPig/log_access_log.2015-10-26.txt

```

192.168.1.132 - - [26/Oct/2015:20:18:05 +0100] "GET /Cubenube/services/Service?wsdl HTTP/1.1" 404 1064

192.168.1.132 - - [26/Oct/2015:20:18:06 +0100] "GET /Cubenube/services/Service?wsdl HTTP/1.1" 404 1064

192.168.1.132 - - [26/Oct/2015:20:18:07 +0100] "GET /Cubenube/services/Service?wsdl HTTP/1.1" 404 1064

192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET / HTTP/1.1" 200 11444

192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET /tomcat.css HTTP/1.1" 200 5926

192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET /tomcat.png HTTP/1.1" 200 5103

192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET /bg-nav.png HTTP/1.1" 200 1401

192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET /bg-middle.png HTTP/1.1" 200 1918

192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET /bg-upper.png HTTP/1.1" 200 3103

192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET /bg-button.png HTTP/1.1" 200 713

192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET /ASF-Logo.png HTTP/1.1" 200 17811

192.168.1.132 - - [26/Oct/2015:20:18:15 +0100] "GET /manager/html HTTP/1.1" 401 2550

192.168.1.132 - - [26/Oct/2015:20:18:18 +0100] "GET /manager/html HTTP/1.1" 401 2550

192.168.1.132 - tomcat [26/Oct/2015:20:18:43 +0100] "GET /manager/html HTTP/1.1" 200 17790

192.168.1.132 - tomcat [26/Oct/2015:20:18:44 +0100] "GET /manager/images/tomcat.gif HTTP/1.1" 200 2066

192.168.1.132 - tomcat [26/Oct/2015:20:18:44 +0100] "GET /manager/images/ASF-Logo.gif HTTP/1.1" 200 7279

192.168.1.132 - - [26/Oct/2015:20:18:56 +0100] "GET /host-manager/html HTTP/1.1" 401 2112

192.168.1.132 - - [26/Oct/2015:20:20:47 +0100] "GET /axis2 HTTP/1.1" 404 980

```

```
192.168.1.132 - - [26/Oct/2015:20:21:11 +0100] "GET / HTTP/1.1" 200 11444
```

```
192.168.1.132 - - [26/Oct/2015:20:27:38 +0100] "GET / HTTP/1.1" 200 11444
```

```
| GNU nano 2.7.4                                     Archivo: ejemplos/log.access.log.2015-10-26.txt
192.168.1.132 - - [26/Oct/2015:20:18:05 +0100] "GET /Cubenube/services/Service?wsdl HTTP/1.1" 404 1064
192.168.1.132 - - [26/Oct/2015:20:18:06 +0100] "GET /Cubenube/services/Service?wsdl HTTP/1.1" 404 1064
192.168.1.132 - - [26/Oct/2015:20:18:07 +0100] "GET /Cubenube/services/Service?wsdl HTTP/1.1" 404 1064
192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET / HTTP/1.1" 200 11444
192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET /tomcat.css HTTP/1.1" 200 5926
192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET /tomcat.png HTTP/1.1" 200 5103
192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET /bg-nav.png HTTP/1.1" 200 1401
192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET /bg-middle.png HTTP/1.1" 200 1918
192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET /bg-upper.png HTTP/1.1" 200 3103
192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET /bg-button.png HTTP/1.1" 200 713
192.168.1.132 - - [26/Oct/2015:20:18:11 +0100] "GET /asf-logo.png HTTP/1.1" 200 17811
192.168.1.132 - - [26/Oct/2015:20:18:15 +0100] "GET /manager/html HTTP/1.1" 401 2550
192.168.1.132 - - [26/Oct/2015:20:18:18 +0100] "GET /manager/html HTTP/1.1" 401 2550
192.168.1.132 - - [26/Oct/2015:20:18:43 +0100] "GET /manager/html HTTP/1.1" 200 17790
192.168.1.132 - - [26/Oct/2015:20:18:44 +0100] "GET /manager/images/tomcat.gif HTTP/1.1" 200 2066
192.168.1.132 - - [26/Oct/2015:20:18:44 +0100] "GET /manager/images/asf-logo.gif HTTP/1.1" 200 7279
192.168.1.132 - - [26/Oct/2015:20:18:56 +0100] "GET /host-manager/html HTTP/1.1" 401 2112
192.168.1.132 - - [26/Oct/2015:20:20:47 +0100] "GET /axis2 HTTP/1.1" 404 980
192.168.1.132 - - [26/Oct/2015:20:21:11 +0100] "GET / HTTP/1.1" 200 11444
192.168.1.132 - - [26/Oct/2015:20:27:38 +0100] "GET / HTTP/1.1" 200 11444
```

```
hdfs dfs -mkdir /ejerciciosPig
```

```
hdfs dfs -mkdir /ejerciciosPig/logs
```

```
hdfs dfs -put ejemplos/log.access.log.2015-10-26.txt /ejerciciosPig/logs
```

```
bigdata@bigdata:~/pig$ hdfs dfs -mkdir /ejerciciosPig
bigdata@bigdata:~/pig$ hdfs dfs -mkdir /ejerciciosPig/logs
bigdata@bigdata:~/pig$ hdfs dfs -put ejemplos/log.access.log.2015-10-26.txt /ejerciciosPig/logs
```

Es necesario confirmar que todos los demonios incluidos en el jobHistory están arrancados y ejecutar el siguiente script.

```
pig -x local -local -param LOGS='hdfs://localhost:9000/ejerciciosPig/logs/log.access.log.2015-10-26.txt' -param OUTFILE='hdfs://localhost:9000/ejerciciosPig/logs/out' -f /home/bigdata/ejemplosPig/analisisLogsApache.pig
```

```
bigdata@bigdata:~/pig$ pig -x local -param LOGS='hdfs://localhost:9000/ejerciciosPig/logs/log.access.log.2015-10-26.txt' -param OUTFILE='hdfs://localhost:9000/ejerciciosPig/logs/out' -f /home/bigdata/ejemplosPig/analisisLogsApache.pig
17/08/27 17:38:48 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
17/08/27 17:38:48 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
17/08/27 17:38:48 INFO org.apache.pig.Main: Apache Pig version 0.16.0 (r1746539) compiled Jun 01 2016, 23:10:49
17/08/27 17:38:48,408 [main] INFO org.apache.pig.Main: Logging raw input and output to log file /tmp/pig_1503848328457.log
2017-08-27 17:38:49,408 [main] INFO org.apache.pig.Main: Apache Pig version 0.16.0 (r1746539) compiled Jun 01 2016, 23:10:49
2017-08-27 17:38:49,408 [main] INFO org.apache.pig.Pig: DEPRECATION WARNING: user.name is deprecated. Instead, use mapreduce.job.user.name
2017-08-27 17:38:49,408 [main] INFO org.apache.pig.Pig: DEPRECATION WARNING: user.name is deprecated. Instead, use mapreduce.job.user.name
2017-08-27 17:38:49,408 [main] INFO org.apache.hadoop.conf.Configuration.deprecation: user.name is deprecated. Instead, use mapreduce.job.user.name
2017-08-27 17:38:49,408 [main] INFO org.apache.hadoop.conf.Configuration.deprecation: mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-08-27 17:38:49,408 [main] INFO org.apache.hadoop.conf.Configuration.deprecation: fs.defaultFS is deprecated. Instead, use fs.defaultFs
2017-08-27 17:38:49,408 [main] INFO org.apache.hadoop.conf.Configuration.deprecation: mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-08-27 17:38:49,395 [main] INFO org.apache.pig.PigServer: Pig Script ID for the session: PIG-analisisLogsApache.pig-1ad5aef7-bf8e-461f-924c-a1db915f8709
```

```
hdfs dfs -cat /ejerciciosPig/logs/out/*
```

```
bigdata@bigdata:~/pig$ hdfs dfs -cat /ejerciciosPig/logs/out/*
/tomcat.png      1
/tomcat.css      1
manager/images/tomcat.gif      1
manager/images/asf-logo.gif    1
manager/html     1
/bg-upper.png   1
/bg-nav.png     1
/bg-middle.png  1
/bg-button.png  1
/asf-logo.png   1
/                 3
bigdata@bigdata:~/pig$ █
```

3.6. Lectura recomendada



Gates, Alan. *Programming Pig*. Sebastopol, CA: O'Reilly Media; 2011. (ISBN: 9781449317881)².

Esta guía es una herramienta de aprendizaje ideal y una referencia para Apache PIG, el lenguaje de programación que lo ayuda a describir y ejecutar grandes proyectos de datos en Hadoop. Con PIG, se pueden analizar datos sin tener que crear una aplicación completa, lo que permite experimentar con nuevos conjuntos de datos.

²Se puede consultar el índice de contenidos y adquirir la obra en la página web de la editorial: <https://www.safaribooksonline.com/library/view/programming-pig/9781449317881/>

4.1. Introducción y objetivos



Los objetivos de este apartado son:

- Conocer las características de HIVE.
- Aprender a instalarlo sobre un servidor Hadoop.
- Familiarizarse con los distintos tipos de estructuras que componen HIVE.
- Conocer la sintaxis de consulta de HIVE.
- Resolver ejercicios simples de importación de datos y de consulta sobre los mismos.
- Aprender comandos avanzados de definición de datos (DDL).
- Aprender comandos de manipulación de datos (load, insert, update, delete, import, export).
- Practicar con ejercicios de carga de datos.
- Conocer el sistema de almacenamiento de archivos.
- Introducción a SQOOP.



HIVE es una infraestructura de almacén de datos que proporciona métodos de agregación, queries *ad hoc* y análisis de grandes *datasets* almacenados en Hadoop empleando un dialecto SQL llamado Hive Query Language (HQL).

Características

- Sistemas desestructurados, pero con alguna estructura.
- Familiaridad y facilidad de aprendizaje: HQL está basado en SQL.
- Rapidez: respuesta *batch* frente a grandes volúmenes.
- Escalabilidad: posibilidad de añadir nuevos nodos.
- Metastore: Derby o un sistema relacional.

Desventajas

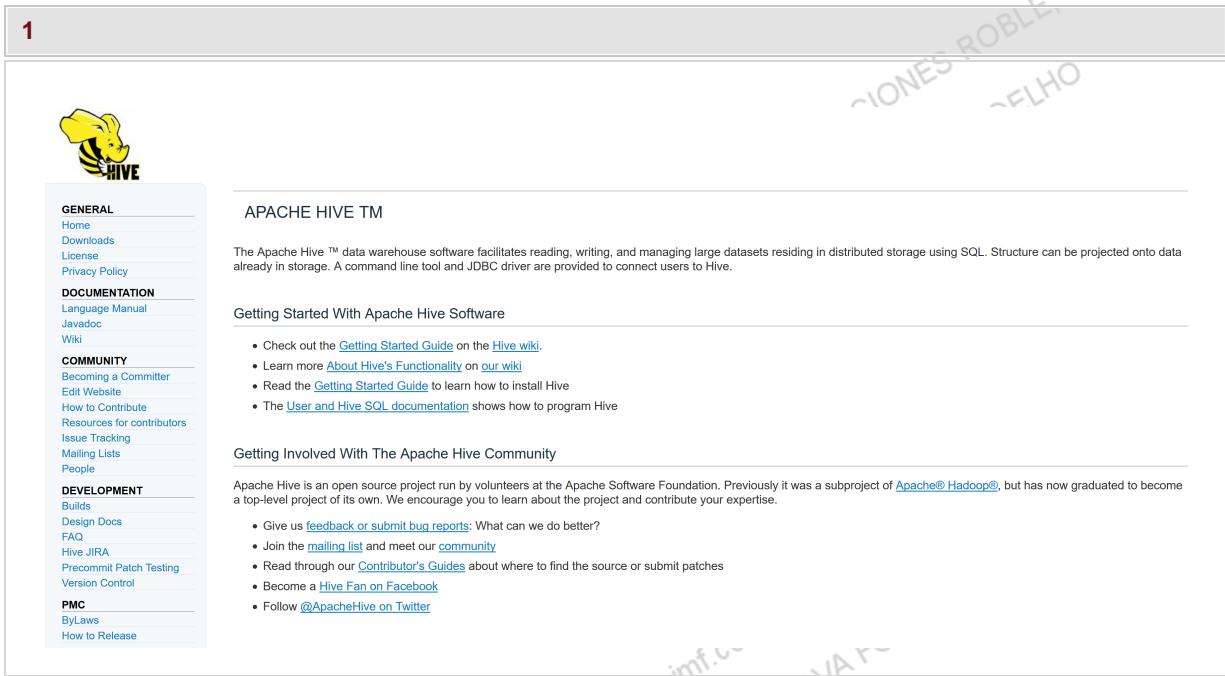
- Alta latencia: no está orientado a procesos *online*.
- No proporciona transaccionabilidad.

4.2. Instalación

Para instalar HIVE, hay que acceder a sus repositorios de descargas y seleccionar la última versión. Para este curso se ha descargado la última versión en el momento de redactar los contenidos y se ha dejado en el directorio home del usuario bigdata para que se pueda seguir correctamente todo el módulo.

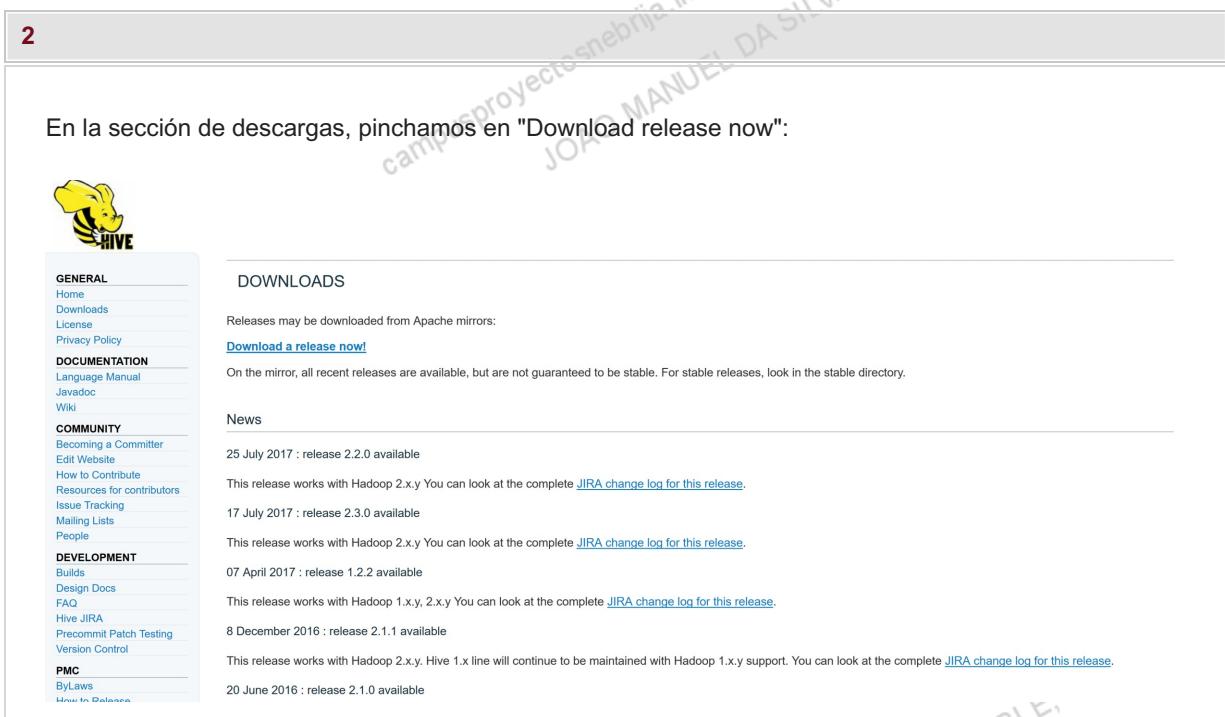
Para descargar HIVE, abrimos la página <http://hive.apache.org/> y, una vez ahí, la sección de descargas.

1



The screenshot shows the Apache Hive homepage. The left sidebar contains links for General, Documentation, Community, Development, and PMC. The main content area is titled "APACHE HIVE TM" and includes sections for "Getting Started With Apache Hive Software" and "Getting Involved With The Apache Hive Community".

2



The screenshot shows the "Downloads" page of the Apache Hive website. It features a sidebar with the same navigation links as the homepage. The main content area is titled "DOWNLOADS" and includes a "News" section with a list of recent releases and their corresponding JIRA change logs.

3

Elegimos el *mirror* que deseemos, en nuestro caso <http://apache.rediris.es/hive/>



We suggest the following mirror site for your download:

<http://apache.uvigo.es/hive/>

Other mirror sites are suggested below. Please use the backup mirrors only to download PGP and MD5 signatures to [verify your downloads](#) or if no other mirrors are working.



Google Custom	
The Apache Way	
Contribute	
ASF Sponsors	

HTTP

<http://apache.rediris.es/hive/>

<http://apache.uvigo.es/hive/>

<http://ftp.cixug.es/apache/hive/>

BACKUP SITES

Please use the backup mirrors only to download PGP and MD5 signatures to [verify your downloads](#) or if no other mirrors are working.

<http://www-eu.apache.org/dist/hive/>

<http://www-us.apache.org/dist/hive/>

4

Seleccionamos la versión deseada y procedemos a la descarga.

The screenshot shows a file listing titled "Index of /hive". The columns are "Name", "Last modified", "Size", and "Description". The "Size" and "Description" columns are mostly blank or show a dash (-). The "Last modified" column shows dates ranging from June 26, 2017, to July 18, 2017. The "Name" column lists various Hive version directories: Parent Directory, hive-1.2.2/, hive-2.1.1/, hive-2.2.0/, hive-2.3.0/, hive-parent-auth-hook/, hive-storage-2.2.1/, hive-storage-2.3.0/, hive-storage-2.3.1/, hive-storage-2.4.0/, ldap-fix/, and stable-2/. The RedIRIS logo is visible at the top left of the page.

5

Si lo que queremos es descargar HIVE directamente desde el terminal de Linux, emplearemos el comando:

```
 wget https://archive.apache.org/dist/hive/hive-2.2.0/apache-hive-2.2.0-bin.tar.gz
```

```
bigdata@bigdata:~$ wget http://apache.rediris.es/hive/hive-2.2.0/apache-hive-2.2.0-bin.tar.gz
--2017-08-27 18:02:13--  http://apache.rediris.es/hive/hive-2.2.0/apache-hive-2.2.0-bin.tar.gz
Resolviendo apache.rediris.es (apache.rediris.es)... 130.206.13.2
Conectando con apache.rediris.es (apache.rediris.es)[130.206.13.2]:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 218189420 (208M) [application/x-gzip]
Guardando como: "apache-hive-2.2.0-bin.tar.gz"

apache-hive-2.2.0-bin.tar 11%[==>] 22,93M 4,91MB/s eta 45s
```

4.3. Configuración

1

Lo primero que haremos es descomprimir el archivo descargado.

```
tar -xvf apache-hive-2.2.0-bin.tar.gz
```

```
bigdata@bigdata:~$ tar -xvf apache-hive-2.2.0-bin.tar.gz
```

2

Moveremos el directorio descomprimido a /home/bigdata/hive

```
mv apache-hive-2.2.0-bin /home/bigdata/hive
```

```
bigdata@bigdata:~$ mv apache-hive-2.2.0-bin /home/bigdata/hive
```

3

Ahora debemos incluir las variables del sistema necesarias para el correcto funcionamiento de HIVE; para ello usaremos el siguiente comando:

```
sudo nano ~/.bashrc
```

```
bigdata@bigdata:~$ sudo nano ~/.bashrc
```

4

Y añadiremos al final del archivo las siguientes líneas:

```
# HIVE VARIABLES START

export HIVE_HOME=/home/bigdata/hive
export PATH=$PATH:$HIVE_HOME/bin

# HIVE VARIABLES END

#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_HOME=/home/bigdata/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
#HADOOP VARIABLES END

#PIG VARIABLES START
export PIG_HOME=/home/bigdata/pig
export PATH=$PATH:$PIG_HOME/bin
#PIG VARIABLES end

# HIVE VARIABLES START
export HIVE_HOME=/home/bigdata/hive
export PATH=$PATH:$HIVE_HOME/bin
# HIVE VARIABLES END
```

5

Para que los cambios sean efectivos, ejecutaremos el comando:

```
source ~/.bashrc
```

```
bigdata@bigdata:~$ source ~/.bashrc
```

6

Antes de arrancar HIVE, deberemos crear los directorios necesarios en HDFS para el correcto funcionamiento de HIVE:

```
hdfs dfs -mkdir /tmp
hdfs dfs -mkdir /user/hive/
hdfs dfs -mkdir /user/hive/warehouse
hdfs dfs -chmod g+w /tmp
hdfs dfs -chmod g+w /user/hive/warehouse
```

```
bigdata@bigdata:~$ hdfs dfs -mkdir /tmp
bigdata@bigdata:~$ hdfs dfs -mkdir /user/hive/
bigdata@bigdata:~$ hdfs dfs -mkdir /user/hive/warehouse
bigdata@bigdata:~$ hdfs dfs -chmod g+w /tmp
bigdata@bigdata:~$ hdfs dfs -chmod g+w /user/hive/warehouse
bigdata@bigdata:~$
```

7

Una vez creados estos directorios y modificada su seguridad, solo nos queda iniciar el *schema* de derby para poder empezar a utilizar HIVE.

```
schematool -dbType derby -initSchema
```

```
bigdata@bigdata:~$ schematool -dbType derby -initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/:org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]
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 2.1.0
Initialization script hive-schema-2.1.0.derby.sql
Initialization script completed
schemaTool completed
bigdata@bigdata:~$
```

4.4. Comprobación

1

Para la comprobación de funcionamiento debemos arrancar HDFS y YARN y verificar que todos los demonios han arrancado correctamente.

```
jps
```

```
bigdata@bigdata:~$ jps
7252 Jps
```

```
start-dfs.sh
```

```
bigdata@bigdata:~$ start-dfs.sh
```

```
start-yarn.sh
```

```
bigdata@bigdata:~$ start-yarn.sh
```

```
$ jps
```

```
bigdata@bigdata:~$ jps
6562 Jps
24359 SecondaryNameNode
24152 DataNode
23979 NameNode
24651 NodeManager
24511 ResourceManager
bigdata@bigdata:~$
```

2

Una vez arrancados los demonios y desde el directorio /home/bigdata/hive ejecutaremos el siguiente comando que arrancará la consola de HIVE:

```
hive
```

```
bigdata@bigdata:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.3.0.jar!/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 engine (i.e. spark, tez) or using Hive 1.X releases.
hive>
```

3

Creamos una tabla test:

```
create table test (a INT);
```

```
hive> create table test (a INT);
OK
Time taken: 1.546 seconds
hive>
```

4

Recuperamos el contenido de la tabla creada anteriormente:

```
select * from test;
```

```
hive> select * from test;
OK
Time taken: 2.547 seconds
hive> ■
```

5

Y terminamos borrando la tabla creada:

```
drop table test;
```

```
hive> drop table test;
OK
Time taken: 2.079 seconds
hive> ■
```

6

Para salir, introduciremos uno de los comandos siguientes:

`exit; / quit;`

```
hive> exit;
bigdata@bigdata:~$
```

4.5. Comandos consola

La documentación sobre la consola esta disponible en [esta página de la Fundación Apache para HIVE](#)

The screenshot shows the Apache Hive LanguageManual Cli documentation. The left sidebar includes links for Páginas, Blog, ACCESES DIRECTOS DE ESPACIO, How-to articles, and PÁGINAS HIJAS. The main content area is titled "LanguageManual Cli" and was last modified by Grant Sohn on December 03, 2016. It contains sections on "Hive CLI" (with sub-links for Hive CLI, Deprecation in favor of Beeline CLI, Hive Command Line Options, Examples, The hiverc File, Logging, Tool to Clear Dangling Scratch Directories, Hive Batch Mode Commands, Hive Interactive Shell Commands, and Hive Resources), "HCatalog CLI", and "Deprecation in favor of Beeline CLI". A note at the bottom states that \$HIVE_HOME/bin/hive is a shell utility for running Hive queries.

1

En cualquier momento, podemos acceder a la ayuda de HIVE a través del comando:

`hive -help`

```
bigdata@bigdata:~$ hive -help
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]
usage: hive
-d,--define <key=value>           Variable substitution to apply to hive
                                     commands. e.g. -d A=B or --define A=B
--database <database>              Specify the database to use
-e <quoted-query-string>          SQL from command line
-f <filename>                     SQL from files
-H,--help                          Print help information
--hiveconf <property=value>        Use value for given property
--hivevar <key=value>              Variable substitution to apply to hive
                                     commands. e.g. --hivevar A=B
-i <filename>                     Initialization SQL file
-S,--silent                        Silent mode in interactive shell
-v,--verbose                       Verbose mode (echo executed SQL to the
                                     console)
bigdata@bigdata:~$
```

2

Accederemos a la consola a través del comando:

```
hive
```

3

Una vez en la consola, podemos crear tablas con el comando create:

```
create table test (data String);
```

```
hive> create table test (data String);
OK
Time taken: 1.696 seconds
hive>
```

4

Con el comando insert, insertaremos varias cadenas en la tabla test:

```
insert into table test values
('cadena1'),('cadena2'),('cadena3'),('cadena que no se vera');
```

```
hive> Insert into table test values ('cadena1'),('cadena2'),('cadena3'),('cadena que no se vera');
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = bigdata_20170827201336_ddba459a-214d-4131-baf8-be277e535a0c
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job 1: job_1503850422017_0001. Tracking URL: http://bigdata:8088/proxy/application_1503850422017_0001/
Kill Command = /home/bigdata/hadoop/bin/hadoop job -kill job_1503850422017_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2017-08-27 20:13:53,977 Stage-1 map = 0%, reduce = 0%
2017-08-27 20:14:03,165 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.47 sec
MapReduce Total cumulative CPU time: 1 seconds 470 msec
End Job : Job 1503850422017_0001
Stage-0 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://localhost:9000/user/hive/warehouse/test/.hive-staging_hive_2017-08-27_20-13-36_957_1301658263538253051-1/-ext-10000
Loading data to table default.test
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Cumulative CPU: 1.47 sec   HDFS Read: 3816 HDFS Write: 114 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 470 msec
OK
Time taken: 28.003 seconds
hive>
```

5

Saldremos de la consola con el siguiente comando:

```
quit;
```

```
hive> quit;
bigdata@bigdata:~$
```

6

En la ayuda de HIVE, observamos que la opción -e nos permite escribir sentencias SQL a través de la línea de comandos:

-e <quoted-query-string>	SQL from command line
-f <filename>	SQL from files

7

A través del parámetro -e ejecutaremos la siguiente consulta:

```
hive -e "select * from test limit 3"
```

```
bigdata@bigdata:~$ hive -e "select * from test limit 3"
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true
OK
cadena1
cadena2
cadena3
Time taken: 3.211 seconds, Fetched: 3 row(s)
bigdata@bigdata:~$
```

8

Para continuar con los ejemplos, crearemos el directorio /home/bigdata/ejemplosHive con el siguiente comando:

```
mkdir /home/bigdata/ejemplosHive
```

Y crearemos el archivo queries.hql:

```
sudo nano /home/bigdata/ejemplosHive/queries.hql
```

```
bigdata@bigdata:~$ sudo nano /home/bigdata/ejemplosHive/queries.hql
```

Con el contenido:

```
select x.* from test x;
```

```
GNU nano 2.7.4                               Archivo: /home/bigdata/ejemplosHive/queries.hql
select x.* from test x;■
```

Con el parámetro -f podemos indicarle a HIVE un archivo donde están las consultas que ha de ejecutar:

```
hive -f /home/bigdata/ejemplosHive/queries.hql
```

```
bigdata@bigdata:~$ hive -f /home/bigdata/ejemplosHive/queries.hql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true
OK
cadena1
cadena2
cadena3
cadena que no se vera
Time taken: 3.059 seconds, Fetched: 4 row(s)
bigdata@bigdata:~$
```

9

Desde la consola de HIVE también podemos ejecutar consultas indicando la ubicación del archivo que las contiene.

```
hive
```

```
source /home/bigdata/ejemplosHive/queries.hql;
```

```
bigdata@bigdata:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/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 engine (i.e. spark, tez) or using Hive 1.X releases
hive> source /home/bigdata/ejemplosHive/queries.hql;
OK
cadena1
cadena2
cadena3
cadena que no se vera
Time taken: 3.517 seconds, Fetched: 4 row(s)
hive> █
```

```
quit;
```

```
hive> quit;
bigdata@bigdata:~$
```



Importante: Cerrar sesiones

Local MetaStoreAdmin solo permite una conexión a Derby usado en entornos locales³.

³Apache Hive: “AdminManual MetastoreAdmin”. [En línea] URL disponible en:
<https://cwiki.apache.org/confluence/display/Hive/AdminManual+MetastoreAdmin>

¶

```
Another instance of Derby may have already booted  
rby.iapi.error.StandardException.newException(Unk  
rby.impl.store.raw.data.BaseDataFileFactory.privG  
rby.impl.store.raw.data.BaseDataFileFactory.su(Up
```

10

Ahora, crearemos el archivo datos con el contenido que figura a continuación:

```
sudo nano /home/bigdata/ejemplosHive/datos
```

```
bigdata@bigdata:~$ sudo nano /home/bigdata/ejemplosHive/datos
```

```
dato1
```

```
dato2
```

```
dato3
```

```
dato4
```

```
dato5
```

```
GNU nano 2.7.4
```

```
Archivo: /home/bigdata/ejemplosHive/datos
```

```
dato1  
dato2  
dato3  
dato4  
dato5
```

11

Una vez creado, ejecutaremos el siguiente comando para insertar el contenido de datos en la tabla test:

```
hive -e "LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/datos' INTO TABLE test;"
```

```
bigdata@bigdata:~$ hive -e "LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/datos' INTO TABLE test;"  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]  
  
Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true  
Loading data to table default.test  
OK  
Time taken: 2.68 seconds  
bigdata@bigdata:~$
```

Si ahora ejecutamos la consulta del archivo queries.hql, recuperaremos todos los datos contenidos en la tabla test.

```
hive -f /home/bigdata/ejemplosHive/queries.hql
```

```
bigdata@bigdata:~$ hive -f /home/bigdata/ejemplosHive/queries.hql  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]  
  
Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true  
OK  
cadena1  
cadena2  
cadena3  
cadena que no se vera  
dato1  
dato2  
dato3  
dato4  
dato5  
Time taken: 3.934 seconds, Fetched: 9 row(s)  
bigdata@bigdata:~$
```

Con el parámetro -S evitaremos las salidas por pantalla. En este caso redirigimos la salida de la consulta a result.txt

-S,--silent

Silent mode in interactive shell

```
hive -S -e "select * from test limit 3;" > /home/bigdata/ejemplosHive/result.txt
```

```
bigdata@bigdata:~$ hive -S -e "select * from test limit 3;" > /home/bigdata/ejemplosHive/result.txt  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]  
bigdata@bigdata:~$
```

Para comprobar el funcionamiento, consultaremos el contenido de result.txt con el comando siguiente:

```
cat /home/bigdata/ejemplosHive/result.txt
```

```
bigdata@bigdata:~$ cat /home/bigdata/ejemplosHive/result.txt  
cadena1  
cadena2  
cadena3  
bigdata@bigdata:~$
```



Para terminar, en el siguiente ejemplo vemos cómo podemos hacer uso de los comandos de HDFS desde la propia consola de HIVE:

`hive`

`dfs -ls /;`

```
hive> dfs -ls /;
Found 6 items
drwxr-xr-x  - bigdata supergroup          0 2017-08-27 14:26 /ejemplo
drwxr-xr-x  - bigdata supergroup          0 2017-08-12 00:27 /ejercicio1
drwxr-xr-x  - bigdata supergroup          0 2017-08-27 17:37 /ejerciciosPig
drwxr-xr-x  - bigdata supergroup          0 2017-08-12 00:28 /out
drwxrwxr-x  - bigdata supergroup          0 2017-08-27 20:13 /tmp
drwxr-xr-x  - bigdata supergroup          0 2017-08-27 19:49 /user
hive> █
```

`dfs -help;`

```
hive> dfs -help;
Usage: hadoop fs [generic options]
      [-appendToFile <localsrc> ... <dst>]
      [-cat [-ignoreCrc] <src> ...]
      [-checksum <src> ...]
      [-chgrp [-R] GROUP PATH...]
      [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
      [-chown [-R] [OWNER][:[GROUP]] PATH...]
      [-copyFromLocal [-f] [-p] [-l] [-d] <localsrc> ... <dst>]
      [-copyToLocal [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
      [-count [-q] [-h] [-v] [-t [<storage type>]] [-u] [-x] <path> ...]
      [-cp [-f] [-p | -p[topax]] [-d] <src> ... <dst>]
      [-createSnapshot <snapshotDir> [<snapshotName>]]
      [-deleteSnapshot <snapshotDir> <snapshotName>]
      [-df [-h] [<path> ...]]
      [-du [-s] [-h] [-x] <path> ...]
      [-expunge]
```

4.6. Arquitectura

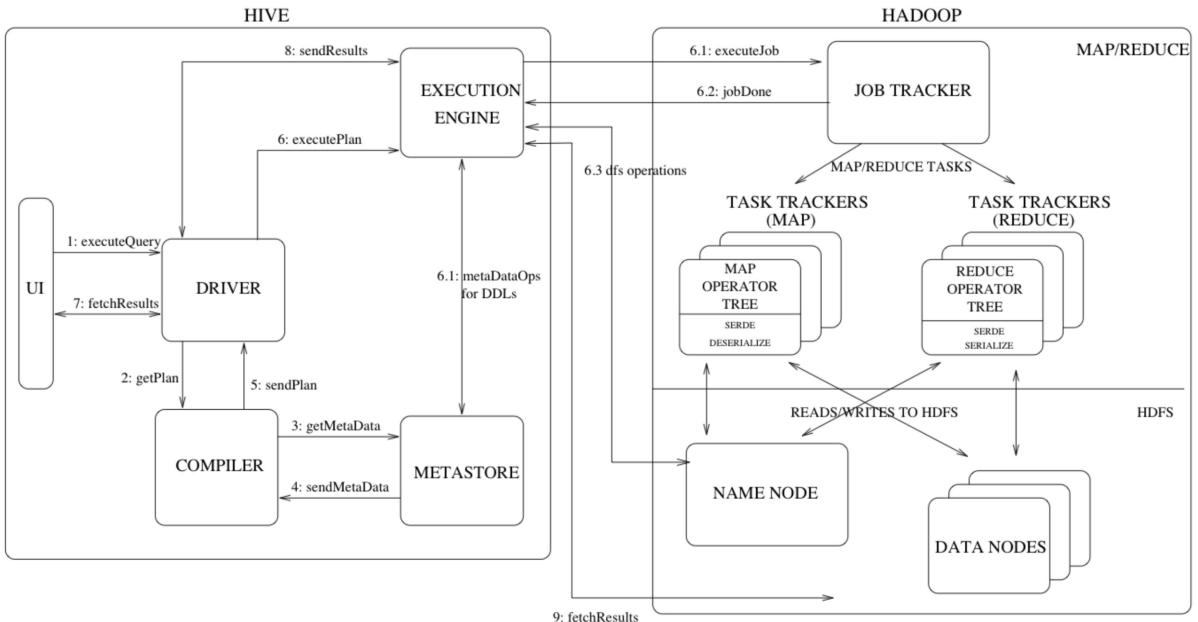


Figura 1. Arquitectura HIVE. Fuente: Fundacion Apache.

La figura 1 muestra los principales componentes de HIVE y sus interacciones con Hadoop. Como se ve aquí, los componentes principales de HIVE son⁴:

IU

La interfaz de usuario para que los usuarios envíen consultas y otras operaciones al sistema. A partir de 2011, el sistema tenía una interfaz de línea de comandos y se estaba desarrollando una GUI web.

Driver

El componente que recibe las consultas. Este componente implementa mecanismos de control de sesión y proporciona API JDBC / ODBC.

Compilador

El componente que analiza la consulta, realiza un análisis semántico en los diferentes bloques de la consulta y expresiones y, finalmente, genera un plan de ejecución con la ayuda de los metadatos de partición buscados en el Metastore.

Metastore

El componente que almacena toda la información de estructura de las distintas tablas y particiones del almacén, incluida la información del tipo columna y columna, los serializadores y deserializadores necesarios para leer y escribir datos y los archivos HDFS correspondientes donde se almacenan los datos.

Motor de ejecución

El componente que ejecuta el plan de ejecución creado por el compilador. El plan es un DAG de etapas. El motor de ejecución administra las dependencias entre estas diferentes etapas del plan y ejecuta estas etapas en los componentes del sistema apropiados.

⁴Más información en la página de la comunidad: <https://cwiki.apache.org/confluence/display/Hive/Design>

4.7. Metastore

Todos los metadatos de las tablas y las particiones de HIVE son accesibles desde el Metastore de HIVE. El Metastore contiene información persistente mediante el uso de JPOX (*Java Persistent Objects*), por lo que cualquier base de datos puede ser soportada. La mayoría de las bases de datos relacionales libres y comerciales son soportadas.

Hay dos tipos de configuraciones del servidor Metastore y de la base de datos:

- Embedido o local.
- Remoto.

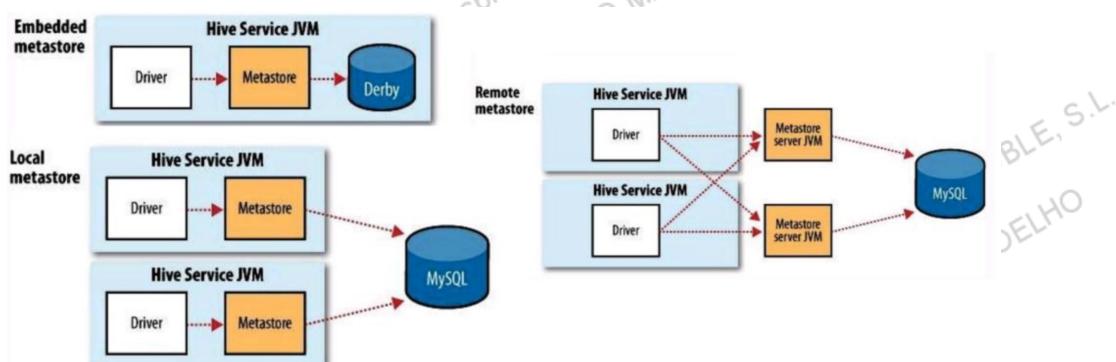


Figura 2. Metastore. Fuente: Fundacion Apache.

4.8. Tipos de datos

HIVE maneja los tipos de datos⁵ más comunes utilizados por otros motores de bases de datos, lenguajes de programación, etc.

⁵Se puede encontrar más información en: <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types>

4.8.1. Tipos simples

Tipos simples numéricos:

- tinyint: entero con signo de 1 byte (-128 a 127).

- smallint: entero con signo de 2 byte (-32.768 a 32.767).
- int: entero con signo de 4 bytes (-2.147.483.648 a 2.147.483.647).
- Bigint: entero con signo de 8 bytes (-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807)
- float: número en punto flotante de 4 bytes.
- double: número en punto flotante de 8 bytes.

Tipos simples Date/Time:

- Timestamp: permite formato timestamp de Unix con milisegundos de forma opcional.
- Date: almacena formatos fechas del estilo {YYYY-MM-DD} sin incluir la hora del día.

Tipos simples cadenas:

- String: se pueden expresar con comillas simples o dobles.
- Varchar: disponible desde la versión 0.12. (1..65355).
- Char: disponible desde la versión 0.13 (1..255).

Otros tipos de datos simples:

- Boolean.
- Binary: desde la versión 0.8.

4.8.2. Tipos complejos

También maneja estructuras complejas como los arrays, structs y maps.

Array:

- Definición: secuencias ordenadas del mismo tipo de datos que son indexables por enteros.
- Sintaxis: [value <, value ...>]
- Posición inicial: 0.
- Ejemplo:
 - ['Juan','Perez']
 - Usuario[0]

Struct:

- Definición: similares a las estructuras de C. Se puede acceder a sus campos con “.”
- Sintaxis: (field TYPE[, field TYPE...])
- Ejemplo:
 - STRUCT {nombre STRING, apellidos STRING}
 - struct ('Juan','Perez')
 - usuario.nombre

Map:

- Definición: es una colección de claves-valores a los que se accede con la notación array.
- Sintaxis: [key,value <, key,value ...>]

- Ejemplo:
 - map ('nombre', 'Juan', 'apellido', 'Perez')
 - usuario['nombre']

4.9. Almacenamiento de información

A la hora de almacenar la información en HDFS, HIVE utiliza una codificación basada en separadores donde:

\n = separa cada línea o registro en el fichero de texto.

^A = se corresponde con el valor octal 01 y se emplea para separar los campos de las columnas.

^B = se emplea para separar tipos de datos complejos con el octal 02.

^C = se emplea para separar las claves de los MAP Key-Value con el octal 03.



De este modo, podríamos almacenar la siguiente estructura (ejemplo):

- Nombre: John Doe.
- Salario: 50.000.
- Subordinados: Mary Smith y Peter Jones.
- Deducciones:
 - IRPF 20 %.
 - Seg Social 5 %.
 - Seguros 10 %.
- Address:
 - Calle Maria de Molina 20.
 - Ciudad Madrid.
 - Comunidad Madrid.
 - CP: 28012.

Del modo siguiente:

John Doe^A50000.0^AMary Smith^BPeter Jones

1

Si creamos la siguiente tabla:

```
create table empleados (
    nombre STRING,
    salario FLOAT,
    subordinados ARRAY<STRING>,
    deducciones MAP<STRING, FLOAT>,
    dirección STRUCT<calle:STRING, ciudad:STRING, comunidad:STRING, codpostal:INT>
);
```

2

Internamente, la información se almacenaría:

Para ver el código que necesitas, pulsa [aquí](#).

4.10. Operadores

HIVE soporta la mayoría de los operadores más comunes.

Operadores aritméticos

- A + B
- A - B
- A * B
- A / B
- A % B

Operadores de comparación

- A = B
- A != B
- A < B
- A > B
- A <= B
- A >= B
- A like B
- A Rlike B o A REGEXP B

Operadores lógicos

- A & B
- A | B
- A ^ B
- ~ A
- A AND B → A && B
- A OR B → A || B
- NOT A → !A

Operadores sobre nulos

- A is null
- A is not null

4.11. Otras funciones

HIVE ofrece funciones de cálculo, tratamiento de cadenas, fechas y otros tipos de datos.

Funciones numéricas

- round(double a)
- floor (double a)
- ceil (double a)
- rand()
- rand (int semilla)
- size (Map <k,v>)
- size (Array <t>)

Funciones de fechas

- from_unixtime (int unixtime)
- to_date (string timestamp)
- year (string date)
- month (string date)
- day (string date)

Funciones de cadenas

- concat (string A, string B, ...)
- substr (string A, int start)
- substr (string A, int start, int length)
- upper (string A)
- ucase (string A)
- lower (string A)
- lcase (string A)
- trim (string A)
- rtrim (string A)
- regexp_replace (string A, string B, string C)

Funciones de agregación

- count(*)
- count(expr)
- count(DISTINCT expr [,expr])
- sum (col)
- sum (DISTINCT col)
- avg (col)
- avg (DISTINCT col)
- min (col)
- max (col)

Otras funciones

- cast (<expr> as <type>)
- Get_json_object(string json_string, string path)

4.12. Ejemplo: word count

A continuación, se desarrolla un ejemplo que permitirá contar las palabras de dos ficheros de texto a través de HIVE.

1

1. Crear documentos de texto:

```
echo -e "Hola\nmundo.\nEjemplo\nde\fichero\nde\nHive" > /home/bigdata/ejemplosHive/ejemplo1
```

```
echo -e "Hola Hive. Segundo fichero" > /home/bigdata/ejemplosHive/ejemplo2
```

```
cat /home/bigdata/ejemplosHive/ejemplo1
```

```
bigdata@bigdata:~$ echo -e "Hola\nmundo.\nEjemplo\nde\fichero\nde\nHive" > /home/bigdata/ejemplosHive/ejemplo1
bigdata@bigdata:~$ echo -e "Hola Hive. Segundo fichero" > /home/bigdata/ejemplosHive/ejemplo2
bigdata@bigdata:~$ cat /home/bigdata/ejemplosHive/ejemplo1
Hola
mundo.
Ejemplo
de
fichero
de
Hive
bigdata@bigdata:~$
```

2

2. Arrancar HIVE:

```
hive
```

```
bigdata@bigdata:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impi-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/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 engine (i.e. spark, tez) or using Hive 1.X releases
hive>
```

3

3. Crear tabla y salir de HIVE:

```
CREATE TABLE docs (line STRING);
```

```
quit; o exit;
```

```
hive> CREATE TABLE docs (line STRING);
OK
Time taken: 1.888 seconds
hive> quit;
bigdata@bigdata:~$
```

4

4. Cargar ficheros en la tabla:

```
hive -e "LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/ejemplo1' INTO TABLE docs";
```

```
hive -e "LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/ejemplo2' INTO TABLE docs";
```

```
bigdata@bigdata:~$ hive -e "LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/ejemplo1' INTO TABLE docs";
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true
Loading data to table default.docs
OK
Time taken: 2.567 seconds
bigdata@bigdata:~$ hive -e "LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/ejemplo2' INTO TABLE docs";
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true
Loading data to table default.docs
OK
Time taken: 2.233 seconds
bigdata@bigdata:~$
```

5

5. Acceder a HIVE y ejecutar un consulta sencilla para verificar que hay datos:

```
hive
```

```
select * from docs;
```

```
bigdata@bigdata:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/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 engine (i.e. spark, tez) or using Hive 1.X releases.
hive> select * from docs;
OK
Hola
mundo.
Ejemplo
de
fichero
de
Hive
Hola Hive. Segundo fichero
Time taken: 3.78 seconds, Fetched: 8 row(s)
hive>
```

6. Trocear cada una de las líneas de los documentos en palabras separadas por espacios en blanco y luego agruparlas para contar el número de palabras. IMPORTANTE: deben estar arrancados los demonios de Map Reduce:

```
CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
(SELECT explode(split(line, ' ')) AS word FROM docs) w
GROUP BY word
ORDER BY word;
```

```
hive> CREATE TABLE word_counts AS
-> SELECT word, count(1) AS count FROM
-> (SELECT explode(split(line, ' ')) AS word FROM docs) w
-> GROUP BY word
-> ORDER BY word;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = b1gdata_2017082703917_0979c18b-64f4-4595-8117-2699859e8290
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 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_1503850422017_0002, Tracking URL = http://b1gdata:8088/proxy/application_1503850422017_0002/
Kill Command = /home/b1gdata/hadoop/bin/hadoop job -kill job_1503850422017_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-08-27 20:39:29,056 Stage-1 map = 0%, reduce = 0%
2017-08-27 20:39:36,027 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.38 sec
2017-08-27 20:39:36,027 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 2.43 sec
MapReduce Total cumulative CPU time: 2 seconds 430 msec
Ended Job = job_1503850422017_0002
Launching Job 2 out of 2
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_1503850422017_0003, Tracking URL = http://b1gdata:8088/proxy/application_1503850422017_0003/
Kill Command = /home/b1gdata/hadoop/bin/hadoop job -kill job_1503850422017_0003
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2017-08-27 20:40:04,201 Stage-2 map = 0%, reduce = 0%
2017-08-27 20:40:10,859 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 0.91 sec
2017-08-27 20:40:10,859 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 2.0 sec
MapReduce Total cumulative CPU time: 2 seconds 6 msec
Ended Job = job_1503850422017_0003
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/word_counts
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1  Cumulative CPU: 2.43 sec   HDFS Read: 8520 HDFS Write: 290 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1  Cumulative CPU: 2.0 sec   HDFS Read: 5277 HDFS Write: 141 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 430 msec
ok
Time taken: 55.323 seconds
hive>
```

7

7. Consultar el resultado:

```
select * from word_counts;
```

```
hive> select * from word_counts;
OK
Ejemplo 1
Hive    1
Hive.   1
Hola    2
Segundo 1
de      2
fichero 2
mundo.  1
Time taken: 0.346 seconds, Fetched: 8 row(s)
hive>
```

8

8. Mostrar todas las tablas existentes:

```
show tables;
```

```
hive> show tables;
OK
docs
test
word_counts
Time taken: 0.096 seconds, Fetched: 3 row(s)
hive>
```

9

9. Mostrar información de una tabla:

```
describe word_counts;
```

```
exit;
```

```
hive> describe word_counts;
OK
word                      string
count                     bigint
Time taken: 0.158 seconds, Fetched: 2 row(s)
hive> exit;
bigdata@bigdata:~$
```

4.13. Estructuras de datos

La información en HIVE se agrupa en estructuras de datos complejas. Las estructuras de datos disponibles son:

- Bases de datos.
- Tablas.
- Particiones (filas).
- Buckets (columnas).
- Vistas.
- Índices.

Estas estructuras se comentarán en los puntos sucesivos.

En el archivo `hive-site.xml` se define la propiedad `hive.metastore.warehouse.dir`. Esta propiedad nos indicará la ruta utilizada para el Metastore. El Metastore guardará los metadatos de cómo HIVE almacenará la información en HDFS.

Estructura	Ejemplo	Ruta
Base de datos	testDB	/hive_warehouse/testDB.db
Tablas	tabla1	/hive_warehouse/testDB.db/tabla1
Partición	date='20141220'	/hive_warehouse/testDB.db/tabla1/date=20140120
Buckets	userid	/hive_warehouse/testDB.db/tabla1/date=20140120/0000021_0

Tabla 1. Estructuras de datos. *Fuente:* Fundación Apache.

Sintaxis:

```
describe formatted <table_name>;
```

```
hive -S -e "describe formatted <table_name> ;" | grep 'Location' | awk '{ print $NF }'
```



Ejemplo:

```
hive -S -e "describe formatted docs;" | grep 'Location' | awk '{ print $NF }'
```

```
bigdata@bigdata:~$ hive -S -e "describe formatted docs;" | grep 'Location' | awk '{ print $NF }'
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-1.2.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
hdfs://localhost:9000/user/hive/warehouse/docs
bigdata@bigdata:~$
```

```
hdfs dfs -ls -R /user/hive/warehouse/docs
```

```
bigdata@bigdata:~$ hdfs dfs -ls -R /user/hive/warehouse/docs
-rwxrwxr-x 1 bigdata supergroup 39 2017-08-27 20:36 /user/hive/warehouse/docs/ejemplo1
-rwxrwxr-x 1 bigdata supergroup 27 2017-08-27 20:36 /user/hive/warehouse/docs/ejemplo2
bigdata@bigdata:~$
```

```
hdfs dfs -cat /user/hive/warehouse/docs/ejemplo1
```

```
bigdata@bigdata:~$ hdfs dfs -cat /user/hive/warehouse/docs/ejemplo1
Hola
mundo.
Ejemplo
de
fichero
de
Hive
bigdata@bigdata:~$
```

A continuación, vamos a ver en detalle cada una de las estructuras complejas enumeradas en el punto anterior.

4.13.1. Bases de datos

Como en otros motores, el nivel de agregación más alto de información es la base de datos. En este enlace se puede ver la [documentación técnica de la comunidad](#):

The screenshot shows the Apache Hive LanguageManual DDL page. The left sidebar includes links for Páginas, Blog, and PÁGINAS HIJAS (LanguageManual, LanguageManual DDL, LanguageManual DDL Bucketing, Exchange Partition). The main content area shows the LanguageManual DDL page with a table of contents for Hive Data Definition Language, including sections like Overview, Keywords, Non-reserved Keywords and Reserved Keywords, Create/Drop/Alter Database, Create/Drop/Truncate Table, Alter Table/Partition/Column, Create/Drop/Alter View, Create/Drop/Alter Index, Create/Drop/Reload Function, Create/Drop/Grant/Revoke Roles and Privileges, Show, Describe, HCatalog and WebHCat DDL.

LanguageManual DDL

Creado por Confluence Administrator, modificado por última vez por pengcheng xiong el dic 23, 2015

Hive Data Definition Language

- Hive Data Definition Language
 - Overview
 - Keywords, Non-reserved Keywords and Reserved Keywords
 - Create/Drop/Alter Database
 - Create/Drop/Truncate Table
 - Alter Table/Partition/Column
 - Create/Drop/Alter View
 - Create/Drop/Alter Index
 - Create/Drop/Reload Function
 - Create/Drop/Grant/Revoke Roles and Privileges
 - Show
 - Describe
- HCatalog and WebHCat DDL

Overview

HiveQL DDL statements are documented here, including:

- CREATE DATABASE|SCHEMA, TABLE, VIEW, FUNCTION, INDEX
- DROP DATABASE|SCHEMA, TABLE, VIEW, INDEX
- TRUNCATE TABLE
- ALTER DATABASE|SCHEMA, TABLE, VIEW
- MSCK REPAIR TABLE (or ALTER TABLE RECOVER PARTITIONS)
- SHOW DATABASES|SCHEMAS, TABLES, TBLPROPERTIES, PARTITIONS, FUNCTIONS, INDEX|ES, COLUMNS, CREATE TABLE
- DESCRIBE DATABASE|SCHEMA, table_name, view_name

Algunos comandos disponibles para el manejo de bases de datos son:

Create Database

Sintaxis:

`CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name`

`[COMMENT database_comment]`

`[LOCATION hdfs_path]`

`[WITH DBPROPERTIES (property_name=property_value, ...)];`

Ejemplos:

`CREATE DATABASE IF NOT EXISTS banca;`

`SHOW DATABASES;`

`CREATE DATABASE banca COMMENT 'Almacena todas las tablas de banca'`

`LOCATION '/mi/rutapordefecto' WITH DBPROPERTIES ('creator' = 'John Doe', 'date' = '2014-12-20');`

Drop Database

Sintaxis:

DROP (DATABASE/SCHEMA) [IF EXISTS] database_name [RESTRICT/CASCADE];

Ejemplo:

DROP DATABASE IF EXISTS banca;

Alter Database

Sintaxis:

*ALTER(DATABASE/SCHEMA) database_name SET
DBPROPERTIES(property_name=property_value,...);*

ALTER (DATABASE/SCHEMA) database_name SET OWNER [USER/ROLE] user_or_role;

Ejemplo:

ALTER database BANCA set DBPROPERTIES ('edited-by'= 'Mary Smith');

Use Database

Sintaxis:

USE database_name;

Ejemplo:

USE DEFAULT;

Describe Database

Describe

Sintaxis:

DESCRIBE DATABASE [EXTENDED] db_name;

DESCRIBE SCHEMA [EXTENDED] db_name;

Ejemplo:

DESCRIBE DATABASE banca;

Show Database

Sintaxis:

SHOW (DATABASES/SCHEMAS) [LIKE identifier_with_wildcards];

Ejemplo:

Show databases like 'ban';*

4.13.2. Tablas

Las bases de datos contienen tablas, como en otros motores. Algunos comandos disponibles para el manejo de bases de datos son:

Create Table

```

CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name --
(Note: TEMPORARY available in Hive 0.14.0 and later)
[(col_name data_type [COMMENT col_comment], ...)]
[COMMENT table_comment]
[PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
[CLUSTERED BY (col_name, col_name, ...) [SORTED BY (col_name [ASC|DESC], ...)] INTO
num_buckets BUCKETS]
[SKEWED BY (col_name, col_name, ...) ON ((col_value, col_value, ...), ...|col_value, col_value,
...)]
[STORED AS DIRECTORIES] -- (Note: Available in Hive 0.10.0 and later)
[
[ROW FORMAT row_format]
[STORED AS file_format]
[ STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)] -- (Note: Available
in Hive 0.6.0 and later)
]
[LOCATION hdfs_path]
[TBLPROPERTIES (property_name=property_value, ...)] -- (Note: Available in Hive 0.6.0 and
later)
[AS select_statement]; -- (Note: Available in Hive 0.5.0 and later; not supported for external
tables)

```

Ejemplo:

```

CREATE EXTERNAL TABLE IF NOT EXISTS valores_bolsa (activo STRING,simbolo
STRING,ymd     STRING,precio_apertura   FLOAT,precio_maximo   FLOAT,precio_minimo
FLOAT,precio_cierre FLOAT,volumen INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/data/bolsa';

```

DropTable

Sintaxis:

```
DROP TABLE [IF EXISTS] table_name;
```

Ejemplo:

```
drop table valores_bolsa;
```

Características:

- hdfs dfs –rm: elimina metadatos y datos de la tabla.
- Recuperación de datos si está configurado el directorio Trash.

- Los metadatos se pierden por completo.
- Cuando se elimina una tabla externa, los datos de la tabla no se borran del sistema de archivos.
- Cuando se elimina una tabla referenciada por vistas, no se muestra ningún mensaje

Otros comandos

TRUNCATE TABLE table_name [PARTITION partition_spec]

ALTER TABLE table_name RENAME TO new_table_name;

ALTER TABLE table_name SET TBLPROPERTIES table_properties;

ALTER TABLE table_name SET TBLPROPERTIES ('comment' = new_comment);

ALTER TABLE table_name SET SERDE serde_class_name [WITH SERDEPROPERTIES serde_properties];

*ALTER TABLE table_name CLUSTERED BY (col_name, col_name, ...) [SORTED BY (col_name, ...)]
INTO num_buckets BUCKETS;*

*ALTER TABLE table_name [PARTITION partition_spec] CHANGE [COLUMN] col_old_name
col_new_name column_type
[COMMENT col_comment] [FIRST/AFTER column_name] [CASCADE/RESTRICT];*

*ALTER TABLE table_name [PARTITION partition_spec]
ADD/REPLACE COLUMNS (col_name data_type [COMMENT col_comment], ...)
[CASCADE/RESTRICT]*

4.13.3. Tablas particionadas

Dado que nos encontramos en un ecosistema destinado al manejo de grandes volúmenes de datos, existe la posibilidad de particionar tablas muy grandes para aumentar el rendimiento de las consultas y evitar tener que recorrerlas enteras para hacer una determinada búsqueda.

A continuación veremos cómo se crea una tabla particionada, en este caso, por comunidad.

`hdfs://master_server/user/bigdata/warehouse/my.db/empleados`

CREATE TABLE empleados (

nombre STRING,

salario FLOAT,

subordinados ARRAY<STRING>,

deducciones MAP<STRING, FLOAT>,

```
direccion STRUCT<calle:STRING, ciudad:STRING, provincia:STRING, comunidad: STRING,
cod_postal:INT>
)
PARTITIONED BY (comunidad STRING, provincia STRING);
...
hdfs://master_server/user/bigdata/warehouse/my.db/empleados/comunidad=Andalucia/provincia=-
hdfs://master_server/user/bigdata/warehouse/my.db/empleados/comunidad=Andalucia/provincia=-
...
hdfs://master_server/user/bigdata/warehouse/my.db/empleados/comunidad=Galicia/provincia=Lug
hdfs://master_server/user/bigdata/warehouse/my.db/empleados/comunidad=Galicia/provincia=Ore
...
SELECT * FROM empleados
WHERE comunidad = 'Andalucia' AND provincia = 'Sevilla';
SHOW PARTITIONS empleados;
SHOW PARTITIONS employees PARTITION(comunidad='Andalucia');
```

4.13.4. Tablas internas y externas

Tablas internas:

- HIVE controla los metadatos y el ciclo de vida de los datos.
- Los datos se almacenan en la ubicación indicada en `hive.metastore.warehouse.dir`
- Al eliminar las tablas, se eliminan también los datos.

Tablas externas:

- Se definen con las palabras clave `EXTERNAL` y `LOCATION`.
- Las tablas se almacenan fuera de HIVE.
- Pueden usarlas otras aplicaciones o herramientas.
- Al eliminar la tabla, solo se eliminan los metadatos, no los datos, al no asumir HIVE la propiedad de las tablas.

```

CREATE EXTERNAL TABLE IF NOT EXISTS mensajes_log (
  hms INT,
  severidad STRING,
  servidor STRING,
  id_proceso INT,
  mensaje STRING)
COMMENT 'Tabla de mensajes de logs de los servidores'
PARTITIONED BY (anio INT, mes INT, dia INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ''
STORED AS TEXTFILE
LOCATION '<hdfs_location>';

```

4.13.5. Buckets

Por lo general, las tablas particionadas ofrecen una forma de separar datos en múltiples archivos, pero el particionado da resultados efectivos cuando:

- Hay un número limitado de particiones.
- Las particiones son de tamaño parecido.

Pero esto puede no ser posible en todos los escenarios, por ejemplo si particionamos nuestras tablas por país, algunos países pueden tener particiones más grandes que aporten el 70-80 % del total de datos. En estos casos, el particionamiento no será suficiente. Para superar este problema, HIVE proporciona el concepto de Bucket, otra técnica para descomponer las particiones en partes más manejables.

- Particiones: cada valor clave es troceado o particionado en su propio espacio.
- Clúster o bucket: se emplea el *hash* sobre el valor y después se distribuye en los cubos *obuckets*.

```

CREATE TABLE user_info_bucketed(user_id BIGINT, department STRING, firstname STRING,
lastname STRING)
COMMENT 'A bucketed copy of user_info'
PARTITIONED BY(department STRING)
CLUSTERED BY(user_id) INTO 256 BUCKETS;

```

```

set hive.enforce.bucketing = true;
FROM user_id
INSERT OVERWRITE TABLE user_info_bucketed
PARTITION (ds='IT')
SELECT userid, firstname, lastname WHERE ds='IT';

```

4.13.6. Índices

Los índices permiten el aumento del rendimiento de determinadas consultas, como en otros motores.

Create Index

Sintaxis:

```

CREATE INDEX index_name
ON TABLE base_table_name (col_name, ...)

```

AS index_type

[WITH DEFERRED REBUILD]

[IDXPROPERTIES (property_name=property_value, ...)]

[IN TABLE index_table_name]

[

[ROW FORMAT ...] STORED AS ...

| STORED BY ...

]

[LOCATION hdfs_path]

[TBLPROPERTIES (...)]

[COMMENT "index comment"];



```
CREATE TABLE employees (
    name STRING,
    salary FLOAT,
    subordinates ARRAY<STRING>,
    deductions MAP<STRING, FLOAT>,
    address STRUCT<street:STRING, city:STRING,
    state:STRING, zip:INT>
) PARTITIONED BY (country STRING, state STRING);

CREATE INDEX employees_index ON TABLE employees (country)
AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler'
WITH DEFERRED REBUILD
IDXPROPERTIES ('creator = 'me', 'created_at' = 'some_time')
IN TABLE employees_index_table
PARTITIONED BY (country, name)
COMMENT 'Employees indexed by country and name';
```

Otros comandos

```
DROP INDEX [IF EXISTS] index_name ON table_name;
```

```
ALTER INDEX index_name ON table_name [PARTITION partition_spec] REBUILD;
```

4.13.7. Vistas

El uso de vistas en HIVE es el mismo que el de entornos relacionales.

Create View

Sintaxis:

```
CREATE VIEW [IF NOT EXISTS] view_name [(column_name [COMMENT column_comment],
...) ]
[COMMENT view_comment]
[TBLPROPERTIES (property_name = property_value, ...)]
AS SELECT ...;
```

Ejemplo:

```
CREATE VIEW origen_visitas_eoi(url COMMENT 'URL of Referring page')
COMMENT 'Referencia a las visitas de la web del EOI'
AS
SELECT DISTINCT origen_url
FROM page_view
WHERE page_url='http://www.eoi.es';
```

Otros comandos

```
DROP VIEW [IF EXISTS] view_name;
ALTER VIEW view_name SET TBLPROPERTIES table_properties;
ALTER VIEW view_name AS select_statement;
```

4.14. Ejemplo: estructuras complejas

A continuación, presentaremos un ejemplo que ilustrará la creación de algunas de las estructuras complejas vistas en el punto anterior.

```
create table empleados (
    nombre STRING,
    salario FLOAT,
    subordinados ARRAY<STRING>,
    deducciones MAP<STRING, FLOAT>,
    dirección STRUCT<calle:STRING, ciudad:STRING, comunidad:STRING, codpostal:INT>
);
```

1. Crear tabla que permita gestionar tipos de datos básicos

```
create table empleados1 (
    nombre STRING,
    salario FLOAT
)ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

```
hive> create table empleados1 (
    > nombre STRING,
    > salario FLOAT
    > )ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LINES TERMINATED BY '\n'
    > STORED AS TEXTFILE;
```

OK

Time taken: 1.599 seconds

hive>

hive> quit;

\$ sudo nano /home/bigdata/ejemplosHive/empleados1

```
bigdata@bigdata:~$ sudo nano /home/bigdata/ejemplosHive/empleados1
juan perez, 30000
```

maria jimenez, 25000

jesus sanchez, 20000

GNU nano 2.7.4

Archivo: /home/bigdata/ejemplosHive/empleados1

```
juan perez, 30000
maria jimenez, 25000
jesus sanchez, 20000
```

```
hive -e "LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/empleados1' INTO TABLE empleados1";
```

```
bigdata@bigdata:~$ hive -e "LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/empleados1' INTO TABLE empleados1";
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true
Loading data to table default.empleados1
OK
Time taken: 2.293 seconds
bigdata@bigdata:~$
```

```
hive -e "select nombre from empleados1;"
```

```
bigdata@bigdata:~$ hive -e "select nombre from empleados1"
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true
OK
juan perez
maria jimenez
jesus sanchez
Time taken: 3.476 seconds, Fetched: 3 row(s)
bigdata@bigdata:~$
```

2. Crear una tabla que permita gestionar los tipos de datos ARRAY

```
create table empleados2 (
    nombre STRING,
    salario FLOAT,
    subordinados ARRAY<STRING>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '#'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

```

hive> create table empleados2 (
  > nombre STRING,
  > salario FLOAT,
  > subordinados ARRAY<STRING>
  > )
  > ROW FORMAT DELIMITED
  > FIELDS TERMINATED BY ','
  > COLLECTION ITEMS TERMINATED BY '#'
  > LINES TERMINATED BY '\n'
  > STORED AS TEXTFILE;
OK
Time taken: 1.582 seconds
hive> █

```

hive>quit;

sudo nano /home/bigdata/ejemplosHive/empleados2

```

hive> quit;
bigdata@bigdata:~$ sudo nano /home/bigdata/ejemplosHive/empleados2

```

juan perez, 30000, maria jimenez#jesus sanchez

maria jimenez, 25000, jesus sanchez

jesus sanchez, 20000

GNU nano 2.7.4

```

juan perez, 30000, maria jimenez#jesus sanchez
maria jimenez, 25000, jesus sanchez
jesus sanchez, 20000

```

hive -e "LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/empleados2' INTO TABLE empleados2";

```

bigdata@bigdata:~$ hive -e "LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/empleados2' INTO TABLE empleados2";
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true
Loading data to table default.empleados2
OK
Time taken: 2.334 seconds
bigdata@bigdata:~$ █

```

hive -e "select salario from empleados2;"

```
bigdata@bigdata:~$ hive -e "select salario from empleados2;"  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]  
  
Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true  
OK  
30000.0  
25000.0  
20000.0  
Time taken: 3.332 seconds, Fetched: 3 row(s)  
bigdata@bigdata:~$
```

hive -e "select nombre, subordinados from empleados2";

```
bigdata@bigdata:~$ hive -e "select nombre, subordinados from empleados2;"  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]  
  
Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true  
OK  
juan perez      [" maria jimenez","jesus sanchez"]  
maria jimenez   [" jesus sanchez"]  
jesus sanchez    NULL  
Time taken: 3.443 seconds, Fetched: 3 row(s)  
bigdata@bigdata:~$
```

3. Crear una tabla que permita gestionar los tipos de datos MAP

```
create table empleados3 (  
    nombre STRING,  
    salario FLOAT,  
    subordinados ARRAY<STRING>,  
    deducciones MAP<STRING, FLOAT>  
)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ','  
COLLECTION ITEMS TERMINATED BY '#'  
MAP KEYS TERMINATED BY ':'  
LINES TERMINATED BY '\n'  
STORED AS TEXTFILE;
```

```

hive> create table empleados3 (
    > nombre STRING,
    > salario FLOAT,
    > subordinados ARRAY<STRING>,
    > deducciones MAP<STRING, FLOAT>
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > COLLECTION ITEMS TERMINATED BY '#'
    > MAP KEYS TERMINATED BY ':'
    > LINES TERMINATED BY '\n'
    > STORED AS TEXTFILE;
OK
Time taken: 2.017 seconds
hive> 

```

hive>quit;

sudo nano /home/bigdata/ejemplosHive/empleados3

```

hive> quit;
bigdata@bigdata:~$ sudo nano /home/bigdata/ejemplosHive/empleados3

```

juan perez, 30000,maria jimenez#jesus sanchez,IRPF:.25#SEG SOC:.05

maria jimenez, 25000,jesus sanchez,IRPF:.2#SEG SOC:.05

jesus sanchez, 20000,,IRPF:.22#SEG SOC:.05

GNU nano 2.7.4

```

juan perez, 30000,maria jimenez#jesus sanchez,IRPF:.25#SEG SOC:.05
maria jimenez, 25000,jesus sanchez,IRPF:.2#SEG SOC:.05
jesus sanchez, 20000,,IRPF:.22#SEG SOC:.05

```

hive -e "LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/empleados3' INTO TABLE empleados3";

```
bigdata@bigdata:~$ hive -e "LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/empleados3' INTO TABLE empleados3";
SLF43: Class path contains multiple SLF4J bindings.
SLF43: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF43: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF43: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF43: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true
Loading data to table default.empleados3
OK
Time taken: 2.412 seconds
bigdata@bigdata:~$
```

campusproyectosnebrija.imf.com © EJECIONES ROBLE, S.L.

```
hive -e "select nombre from empleados3 where salario > 20000;"
```

```
bigdata@bigdata:~$ hive -e "select nombre from empleados3 where salario > 20000;"
SLF43: Class path contains multiple SLF4J bindings.
SLF43: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF43: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF43: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF43: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true
OK
juan perez
maria jimenez
Time taken: 4.164 seconds, Fetched: 2 row(s)
bigdata@bigdata:~$
```

campusproyectosnebrija.imf.com © EJECIONES ROBLE, S.L.

```
hive -e "select nombre, deducciones from empleados3";
```

```
bigdata@bigdata:~$ hive -e "select nombre, deducciones from empleados3";
SLF43: Class path contains multiple SLF4J bindings.
SLF43: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF43: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF43: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF43: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true
OK
juan perez {"IRPF":0.25,"SEG SOC":0.05}
maria jimenez {"IRPF":0.2,"SEG SOC":0.05}
jesus sanchez {"IRPF":0.22,"SEG SOC":0.05}
Time taken: 3.273 seconds, Fetched: 3 row(s)
bigdata@bigdata:~$
```

4. Crear una tabla que permita gestionar los distintos tipos de datos complejos

```
create table empleados4 (
    nombre STRING,
    salario FLOAT,
    subordinados ARRAY<STRING>,
    deducciones MAP<STRING, FLOAT>,
    direccion STRUCT<calle:STRING, ciudad:STRING, comunidad:STRING, codpostal:INT>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '#'
MAP KEYS TERMINATED BY ':'
LINES TERMINATED BY '\n'
```

STORED AS TEXTFILE;

```
hive> create table empleados4 (
    > nombre STRING,
    > salario FLOAT,
    > subordinados ARRAY<STRING>,
    > deducciones MAP<STRING, FLOAT>,
    > direccion STRUCT<calle:STRING, ciudad:STRING, comunidad:STRING, codpostal:INT>
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > COLLECTION ITEMS TERMINATED BY '#'
    > MAP KEYS TERMINATED BY ':'
    > LINES TERMINATED BY '\n'
    > STORED AS TEXTFILE;
OK
Time taken: 1.802 seconds
hive> [REDACTED]
```

sudo nano /home/bigdata/ejemplosHive/empleados4

juan perez, 30000,maria jimenez#jesus sanchez,IRPF:.25#SEG SOC:.05,Gran Via#Madrid#Madrid#28001

maria jimenez, 25000,jesus sanchez,IRPF:.2#SEG SOC:.05,Calle Mayor 12#Alpedrete#Madrid#28512

jesus sanchez, 20000,,IRPF:.2#SEG SOC:.05,Calle Arenal 3#Madrid#Madrid#28001

GNU nano 2.7.4	Archivo: /home/bigdata/ejemplosHive/empleados4	Modificado
juan perez, 30000,maria jimenez#jesus sanchez,IRPF:.25#SEG SOC:.05,Gran Via#Madrid#Madrid#28001 maria jimenez, 25000,jesus sanchez,IRPF:.2#SEG SOC:.05,Calle Mayor 12#Alpedrete#Madrid#28512 jesus sanchez, 20000,,IRPF:.2#SEG SOC:.05,Calle Arenal 3#Madrid#Madrid#28001		

hive -e "LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/empleados4' INTO TABLE empleados4";

```
bidata@bitdata:~$ hive -e "LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/empleados4' INTO TABLE empleados4";
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/:org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true
Loading data to table default.empleados4
OK
Time taken: 2.618 seconds
bitdata@bitdata:~$ [REDACTED]
```

hive

select nombre from empleados4 where salario > 20000 and salario < 30000;

```
bitdata@bitdata:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/:org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/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 engine (i.e. spark, tez) or using Hive 1.X releases
hive> select nombre from empleados4 where salario > 20000 and salario < 30000;
OK
maria jimenez
Time taken: 4.353 seconds, Fetched: 1 row(s)
hive> [REDACTED]
```

```
select nombre, direccion.calle from empleados4;
```

```
hive> select nombre, direccion.calle from empleados4;
OK
juan perez      Gran Via
maria jimenez   Calle Mayor 12
jesus sanchez   Calle Arenal 3
Time taken: 0.416 seconds, Fetched: 3 row(s)
hive> █
```

4.15. Trabajando con Json

Ahora, se debe crear una tabla que permita gestionar un blog en Json con los siguientes campos:

- idBlog.
- Fecha.
- Usuario que comenta.
- Comentario.
- Datos contacto.
 - E-mail.
 - Tlf.

1

```
sudo nano /home/bigdata/ejemplosHive/comments.txt
```

```
bigdata@bigdata:~$ sudo nano /home/bigdata/ejemplosHive/comments.txt
```

2

```
{ "idBlog" : "75584", "date" : "2015-12-20", "user" : "user1", "comment" : "me gusta el blog",
"contact" : { "email" : "user1@gmail.com", "tlfn" : "91 661 12 33" } }
```

```
{ "idBlog" : "75584", "date" : "2015-12-21", "user" : "user2", "comment" : "otro comentario en el
mismo blog", "contact" : { "email" : "user2@gmail.com" } }
```

```
{ "idBlog" : "75585", "date" : "2015-12-20", "user" : "user1", "comment" : "no me gusta el blog",
"contact" : { "email" : "user1@gmail.com", "tlfn" : "91 661 12 33" } }
```

```
{ "idBlog" : "75585", "date" : "2015-12-22", "user" : "user3", "comment" : "comentario en el otro
blog" }
```

```
GNU nano 2.7.4                               Archivo: /home/bigdata/ejemplosHive/comments.txt
{ "idBlog" : "75584", "date" : "2015-12-20", "user" : "user1", "comment" : "me gusta el blog", "contact" : { "email" : "user1@gmail.com", "tlfn" : "91 661 12 33" } }
{ "idBlog" : "75584", "date" : "2015-12-21", "user" : "user2", "comment" : "otro comentario en el mismo blog", "contact" : { "email" : "user2@gmail.com" } }
{ "idBlog" : "75585", "date" : "2015-12-20", "user" : "user1", "comment" : "no me gusta el blog", "contact" : { "email" : "user1@gmail.com", "tlfn" : "91 661 12 33" } }
{ "idBlog" : "75585", "date" : "2015-12-22", "user" : "user3", "comment" : "comentario en el otro blog" } █
```

3

hive

```
bigdata@bigdata:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/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 engine (i.e. spark, tez) or using Hive 1.X releases.
hive>
```

4

CREATE EXTERNAL TABLE comments(info STRING);

```
hive> CREATE EXTERNAL TABLE comments(info STRING);
OK
Time taken: 1.926 seconds
hive>
```

5

LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/comments.txt' OVERWRITE INTO TABLE comments;

```
hive> LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/comments.txt' OVERWRITE INTO TABLE comments
Loading data to table default.comments
OK
Time taken: 1.598 seconds
hive>
```

6

*select * from comments;*

```
hive> select * from comments;
OK
[{"idBlog": "75584", "date": "2015-12-20", "user": "user1", "comment": "me gusta el blog", "contact": {"email": "user1@gmail.com", "tlfn": "91 661 12 33"}}, {"idBlog": "75584", "date": "2015-12-21", "user": "user2", "comment": "otro comentario en el mismo blog", "contact": {"email": "user2@gmail.com"}}, {"idBlog": "75585", "date": "2015-12-20", "user": "user1", "comment": "no me gusta el blog", "contact": {"email": "user1@gmail.com", "tlfn": "91 661 12 33"}}, {"idBlog": "75585", "date": "2015-12-22", "user": "user3", "comment": "comentario en el otro blog"}]
Time taken: 1.904 seconds, Fetched: 4 row(s)
hive>
```

7

select idBlog from comments;

```
hive> select idBlog from comments;
FAILED: SemanticException [Error 10004]: Line 1:7 Invalid table alias or column reference 'idBlog': (possible column names are: info)
hive> ■
```



Como vemos, no es posible consultar de esa manera. Para trabajar con Json, debemos utilizar la función `get_json_object`, cuya documentación podemos consultar en este enlace: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF#LanguageManualUDF-get_json_object

8

SELECT GET_JSON_OBJECT(comments.info,'\$.idBlog') FROM comments;

```
hive> SELECT GET_JSON_OBJECT(comments.info,'$.idBlog') FROM comments;
OK
75584
75584
75585
75585
Time taken: 0.703 seconds, Fetched: 4 row(s)
hive> ■
```

9

SELECT GET_JSON_OBJECT(c.info,'\$.contact.email') FROM comments c;

```
hive> SELECT GET_JSON_OBJECT(c.info,'$.contact.email') FROM comments c;
OK
user1@gmail.com
user2@gmail.com
user1@gmail.com
NULL
Time taken: 0.45 seconds, Fetched: 4 row(s)
hive> ■
```



Y en el siguiente enlace se puede ampliar información sobre la función lateral view [http://cwiki.apache.org/confluence/display/Hive/LanguageManual+LateralView#LanguageManualLateralView-LateralViewSyntax](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+LateralView#LanguageManualLateralView-LateralViewSyntax)

10

```

SELECT b.idBlog, c.email
FROM comments a
LATERAL VIEW json_tuple(a.info, 'idBlog', 'contact') b AS idBlog, contact
LATERAL VIEW json_tuple(b.contact, 'email', 'tlfn') c AS email, tlfn
WHERE b.idBlog='75584';

```

```

hive> SELECT b.idBlog, c.email
> FROM comments a
> LATERAL VIEW json_tuple(a.info, 'idBlog', 'contact') b AS idBlog, contact
> LATERAL VIEW json_tuple(b.contact, 'email', 'tlfn') c AS email, tlfn
> WHERE b.idBlog='75584';
OK
75584    user1@gmail.com
75584    user2@gmail.com
Time taken: 0.327 seconds, Fetched: 2 row(s)
hive>

```

4.16. Data Definition Language (DDL)

Para la definición y consulta de nuestras bases de datos, tablas, seguridad de acceso, etc., HIVE nos ofrece algunos comandos útiles⁶.

⁶Puedes ampliar la información en la página de la comunidad:
<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL>

Roles y privilegios

Funciones que permiten la gestión de permisos y seguridad⁷

- CREATE ROLE
- GRANT ROLE
- REVOKE ROLE
- DROP ROLE
- SHOW ROLES
- SHOW ROLE GRANT
- SHOW CURRENT ROLES
- SET ROLE
- SHOW PRINCIPALS

⁷Se puede ampliar información en Wiki Apache. *LanguageManual DDL*. “Create/Drop/Grant/Revoke Roles and Privileges”. [En línea] URL disponible en: <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-Create/Drop/Gra>

<p>campusprojectsnabria.imf.com © EDICIONES ROBLE, S.L.</p> <p>JOAO MANUEL DA SILVA FONTES COELHO</p> <p>Show</p> <p>Consulta de características de estructuras de datos, permisos, etc.⁸</p> <ul style="list-style-type: none">→ Show Databases→ Show Tables/Partitions/Indexes→ Show Tables/Partitions/IndexesShow Columns→ Show Tables→ Show Partitions→ Show Table/Partition Extended→ Show Table Properties→ Show Create Table→ Show Indexes→ Show Functions→ Show Granted Roles and Privileges→ Show Locks→ Show Conf→ Show Transactions→ Show Compactions <p>⁸Se puede ampliar información en Wiki Apache. <i>LanguageManual DDL</i>. “Show”. [En línea] URL disponible en: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-Show</p> <p>Describe⁹</p> <ul style="list-style-type: none">→ Describe Database→ Describe Table/View/Column→ Describe Table/View/ColumnDescribe Partition <p>⁹Se puede ampliar información en Wiki Apache. <i>LanguageManual DDL</i>. “Describe”. [En línea] URL disponible en: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-Describe</p>

4.17. Data Manipulation Language (DML)

A continuación, revisaremos los distintos comandos de manipulación de datos.

- Carga de datos:
 - Load.
 - Insert.
 - Export / Import.
- Actualización de información:
 - Update.
- Borrado:
 - Delete.

Cargar ficheros

Sintaxis:

LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)]

LOCAL, si se indica:

- Buscará la ruta de *filepath* en el sistema de archivos local. Si se especifica una ruta relativa, se interpretará como una ruta desde el directorio de trabajo del usuario.
- Intentará copiar todos los archivos indicados en la ruta al sistema de archivos destino. El sistema de archivos destino se deducirá buscando la ubicación del atributo de tabla. Los ficheros copiados se moverán a la tabla.

LOCAL, si no se indica:

- Si el *path* no es absoluto, entonces se interpretará el relativo al usuario, es decir,/user/<username>
- HIVE moverá los archivos indicados en la ruta de *filepath* a la tabla o partición.

Filepath

- Una ruta relativa como ejemplos/ejemplo1.
- Una ruta absoluta como /home/bigdata/ejemplos/ejemplo1.
- Una URI completa como por ejemplo hdfs://namenode:9000/user/bigdata/ejemplos/ejemplo1.
- Se puede referir a un fichero (en cuyo caso HIVE lo moverá transformándolo en tabla) o un directorio (en cuyo caso moverá todos los ficheros dentro de la tabla).
- *Filepath* no puede contener subdirectorios.

Overwrite

- Los contenidos de la tabla o partición objetivo serán eliminados y reemplazados por los ficheros indicados en el *filepath*.
- En caso de no indicarse, los ficheros serán añadidos a la tabla.

Insert

Sintaxis estándar:

```
INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] [IF NOT EXISTS] select_statement1 FROM from_statement;
```

Sintaxis extendida (múltiples insert):

```
FROM from_statement1 INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] [IF NOT EXISTS] select_statement1  
[INSERT OVERWRITE TABLE tablename2 [PARTITION ... [IF NOT EXISTS]] select_statement2]  
[INSERT INTO TABLE tablename2 [PARTITION ...] select_statement2] ...;
```

Inserción de particiones

```
INSERT OVERWRITE TABLE tablename PARTITION (partcol1[=val1], partcol2[=val2] ...) select_statement  
FROM from_statement;
```

```
INSERT INTO TABLE tablename PARTITION (partcol1[=val1], partcol2[=val2] ...) select_statement FROM  
from_statement;
```

- INSERT OVERWRITE sobreescribirá cualquier dato existente en la tabla o partición.
- INSERT INTO añadirá a la tabla o partición los datos, manteniendo los datos existentes intactos.
- La propiedad immutable permite bloquear escrituras en tablas mediante el comando INSERT INTO. Si permite la inserción con el comando OVERWRITE. Por defecto, esta propiedad vale *false*, pero se puede asignar con el comando TBLPROPERTIES ("immutable"="true").
- Los inserts pueden realizarse sobre una tabla o una partición. Si la tabla está particionada, debe indicarse la partición de la tabla y los valores de las columnas particionadas.
- Se pueden realizar múltiples inserts en una misma query, minimizando el número de posicionamientos requeridos para insertar los datos.
- La salida de cada una de las sentencias está escrita en la tabla o partición elegida.
- El formato de salida y la clase de serialización de datos se determina por los metadatos de la tabla (definidos con los comandos DDL).

Insert en particiones dinámicas

En los inserts en particiones dinámicas, los usuarios pueden indicar parte de las especificaciones de la partición, lo que significa que solo es necesario especificar los nombres de las columnas en la sentencia partición. Los valores de las columnas son opcionales. Si el valor está indicado, entonces se trata de una partición estática. En caso contrario, es una partición dinámica. Cada columna particionada dinámicamente tiene una columna de entrada correspondiente indicada en la sentencia select. Esto significa que la creación de las particiones dinámicas se determina por el valor de la columna de entrada. Las columnas particionadas dinámicamente deben especificarse en el mismo orden en que aparecen en la partición.

Por defecto, los inserts en particiones dinámicas están deshabilitados. Estas son las propiedades de configuración más relevantes para las particiones dinámicas:

Propiedad	Valor por defecto	Nota
hive.exec.dynamic.partition	false	Debe establecerse en TRUE para habilitar las inserciones de particiones dinámicas
hive.exec.dynamic.partition.mode	strict	En modo "strict", el usuario debe especificar al menos una partición estática, en el modo no estricto todas las particiones pueden ser dinámicas
hive.exec.max.dynamic.partitions.pernode	100	Número máximo de particiones dinámicas permitidas para crear en cada nodo mapper/reducir
hive.exec.max.dynamic.partitions	1000	Número máximo de particiones dinámicas permitidas para crearse en total
hive.exec.max.created.files	100000	Número máximo de archivos HDFS creados por todos nodos mapper/reducir en un job de MapReduce
hive.error.on.empty.partition	false	Se lanza una excepción si la inserción de partición dinámica genera resultados vacíos

Tabla 2. Propiedades de particiones dinámicas.

Fuente: Fundación Apache.



A continuación, se va a crear una partición llamada comunidad partiendo de una subselect de páginas vistas.

```
INSERT OVERWRITE TABLE page_view PARTITION(dt='2015-12-20', ccaa)
```

```
SELECT pvs.viewTime, pvs.userid, pvs.page_url, pvs.referrer_url, null, null,
pvs.ip, pvs.ccaa
```

```
FROM page_view_stg pvs
```

Escribir datos en el sistema de archivos desde queries

Los resultados de las queries también se pueden insertar en el sistema de archivos empleando una sentencia ligeramente distinta a la anterior.

Sintaxis estándar:

```
INSERT OVERWRITE [LOCAL] DIRECTORY directory1
```

```
[ROW FORMAT row_format] [STORED AS file_format]
```

```
SELECT ... FROM ...
```

Sintaxis extendida para múltiples inserciones:

FROM from_statement

INSERT OVERWRITE [LOCAL] DIRECTORY directory1 select_statement1

[INSERT OVERWRITE [LOCAL] DIRECTORY directory2 select_statement2] ...

row_format

: DELIMITED [FIELDS TERMINATED BY char [ESCAPED BY char]] [COLLECTION ITEMS TERMINATED BY char]

[MAP KEYS TERMINATED BY char] [LINES TERMINATED BY char]

[NULL DEFINED AS char]

Descripción:

- Directory puede ser una URI completa. Si el esquema o la autorización no se indican, entonces se empleará la configurada por Hadoop en la variable `fs.default.name` que especifica la URI del NameNode.
- Si se emplea la palabra LOCAL, HIVE escribirá los datos en el directorio perteneciente al sistema de datos local.
- Los datos escritos en el sistema de archivos son serializados como texto separados por ^A y filas separadas por nuevas líneas. Si alguna de las columnas no es un tipo primitivo, entonces estas columnas son serializadas con el formato Json.
- Las sentencias INSERT OVERWRITE sobre directorios, directorios locales y tablas o particiones se pueden mezclar en la misma query.
- Las sentencias INSERT OVERWRITE sobre directorios del sistema de archivos de HDFS son el mejor modo de extraer grandes cantidades de información de HIVE, ya que HIVE puede escribir en directorios en paralelo empleando un job de Map Reduce.

Insert valores en tablas SQL

El comando insert también se puede usar para volcar datos a tablas de SQL.

Sintaxis estándar:

INSERT INTO TABLE tablename [PARTITION (partcol1[=val1], partcol2[=val2] ...)] VALUES values_row [, values_row ...]

Donde values_row es: (value [, value ...]), donde cada valor es o nulo o cualquier valor válido SQL.

Características:

- Cada fila incluida en el listado de VALUES se inserta en la tabla llamada tablename.
- Los valores deben ser proporcionados para cada columna de la tabla. La sintaxis estándar SQL para permitir insertar valores solo en algunas consultas todavía no se ha desarrollado, pero sí se permite insertar nulos en las columnas a las que el usuario no desee asignar valor.
- Las particiones dinámicas se gestionan del mismo modo que INSERT SELECT.



Ejemplos:

```
CREATE TABLE estudiantes (nombre VARCHAR(64), edad INT, nota  
DECIMAL(3, 2))  
  
CLUSTERED BY (edad) INTO 2 BUCKETS STORED AS ORC;  
  
INSERT INTO TABLE estudiantes  
  
VALUES ('John Smith', 35, 8.28), ('Jane Brown', 32, 7.32);  
  
CREATE TABLE pageviews (userid VARCHAR(64), link STRING, from  
STRING)  
  
PARTITIONED BY (datestamp STRING) CLUSTERED BY (userid) INTO 256  
BUCKETS STORED AS ORC;  
  
INSERT INTO TABLE pageviews PARTITION (datestamp = '2014-12-23')  
  
VALUES ('jsmith', 'mail.com', 'sports.com'), ('jdoe', 'mail.com', null);  
  
INSERT INTO TABLE pageviews PARTITION (datestamp)  
  
VALUES ('tjohnson', 'sports.com', 'finance.com', '2014-12-23'), ('tlee',  
'finance.com', null, '2014-12-21');
```

Update/Delete/Import-Export

UPDATE está disponible desde la versión 0.14.

Sintaxis:

UPDATE tablename SET column = value [, column = value ...] [WHERE expression]

Características:

- La columna referenciada debe ser una columna de la tabla que se va a actualizar.
- El valor asignado debe ser una expresión que soporte HIVE en la cláusula select. Por lo tanto, se permiten operadores aritméticos, UDF, operaciones de casteo y literales. No se permiten subqueries.
- Solo se actualizarán las filas que se ajusten a la cláusula Where.
- Las columnas particionadas o en *buckets* no pueden ser actualizadas.

Delete/Truncate

DELETE está disponible desde la versión 0.14.

Sintaxis:

DELETE FROM tablename [WHERE expression]

TRUNCATE FROM tablename

Import/Export

Sintaxis:

EXPORT TABLE tablename [[PARTITION (part_column="value"[, ...])]] TO 'export_target_path'

IMPORT [[EXTERNAL] TABLE new_or_original_tablename [[PARTITION (part_column="value"[, ...])]]]
FROM 'source_path'
[LOCATION 'import_target_path']

Ejemplo básico de exportación e importación:

```
export table departmento to 'hdfs_exports_location/departmento';
import from 'hdfs_exports_location/departmento';
```

Exportación con cambio de nombre de la tabla renombrada:

```
export table departamento to 'hdfs_exports_location/departamento';
import table departamento_importado from 'hdfs_exports_location/departamento';
```

Exportación de particiones:

```
export table empleados partition (ccaa="andalucia", provincia="sevilla") to
'hdfs_exports_location/empleados';
import from 'hdfs_exports_location/empleados';
```

Exportación de tabla e importación de partición:

```
export table empleados to 'hdfs_exports_location/empleados';
import table empleados partition (ccaa="andalucia", provincia="sevilla") from
'hdfs_exports_location/empleados';
```

Exportación especificando la ubicación de la importación:

```
export table departamento to 'hdfs_exports_location/departamento';
import table departamento from 'hdfs_exports_location/departamento'
location 'import_target_location/departamento';
```

Importando una tabla externa:

```
export table departamento to 'hdfs_exports_location/departamento';
import external table departamento from 'hdfs_exports_location/departamento';
```

4.18. Ejemplo de almacenamiento

Partiendo de los datos estadísticos del gobierno, realizaremos una carga de datos de un fichero en HIVE y, después, efectuaremos una serie de consultas sobre él.

1. Obtener el fichero con los datos

Accederemos a la siguiente página web y descargaremos el fichero del año 2012 <http://www.minhap.gob.es/es-ES/Areas%20Tematicas/Administracion%20Electronica/OVEELL/Paginas/DeudaViva.aspx>

The screenshot shows the official website of the Spanish Ministry of Finance (Hacienda y Función Pública). The header includes the Spanish flag, the text "GOBIERNO DE ESPAÑA", and "MINISTERIO DE HACIENDA Y FUNCIÓN PÚBLICA". The main navigation menu includes links for "Ministerio", "Áreas temáticas", "Prensa", "Central de Información", "Normativa", "Publicaciones", and "Sobre el Ministerio". Below the menu, a breadcrumb trail shows "Inicio / Central de Información / Sistemas de Financiación y Deuda Pública / InformacionEELs". A section titled "Deuda Viva de las Entidades Locales" is displayed, with two subsections: "2016:" and "2015:". Each subsection contains five links, each preceded by a small blue square icon:

- [Deuda viva de los Ayuntamientos a 31/12/2016.](#)
- [Deuda viva de las Diputaciones de Régimen Común, Diputaciones de Régimen Foral, Consejos y Cabildos insulares a 31/12/2016.](#)
- [Deuda viva de las Entidades de ámbito territorial inferior al municipal, Comarcas u otras Entidades que agrupen varios municipios, Áreas Metropolitanas, Mancomunidades de Municipios a 31/12/2016.](#)
- [Resumen de la deuda viva de las Entidades Locales a 31/12/2016.](#)
- [Informe sobre la deuda viva de las Entidades Locales a 31/12/2016.](#)

Below the 2015 section, there is a note in Spanish: "La otra opción es ejecutar directamente el comando wget con la URL asociada a la ubicación del fichero y guardarlo en /home/bigdata/ejemplosHive:"

```
wget
```

```
http://www.minhap.gob.es/Documentacion/Publico/DGCSEL/DeudaViva/DeudaVivaAyuntamientos2012.xls
```

Se abrirá una página de descarga y seleccionaremos la opción guardar.

Abriendo DeudaVivaAyuntamientos2012.xls

Ha elegido abrir:

DeudaVivaAyuntamientos2012.xls
que es: Hoja de cálculo de Microsoft Excel
de: <http://www.minhafp.gob.es>

¿Qué debería hacer Firefox con este archivo?

Abrir con LibreOffice Calc (predeterminada) Guardar archivo Hacer esto automáticamente para estos archivos a partir de ahora.

Cancelar **Aceptar**

A continuación, abriremos el fichero:

DeudaVivaAyuntamientos2012.xls Completada

Mostrar todas las descargas

Se nos abrirá un fichero como el siguiente, con información por municipios y deudas asociadas. Nos centraremos en la solapa Datos_Format.

	A	B	C	D	E	
1						
2	Deuda Viva Entidades Locales a 31/12/2012					
3	(Ayuntamientos)					
4						
5	Código CA	Código Provin	Código Municipio	Municipio	Deuda 31/12/2012 (Miles de €)	
6	01	04	001	Abla	726	
7	01	04	002	Abrucena	235	
8	01	04	003	Adra	24.027	
9	01	04	004	Albánchez	146	
10	01	04	005	Alboloduy	77	
11	01	04	006	Albox	4.453	
12	01	04	007	Alcolea	323	
13	01	04	008	Alcóntar	284	
14	01	04	009	Alcudia de Monteagud	34	
15	01	04	010	Alhabia	1.613	
16	01	04	011	Alhama de Almería	1.349	
17	01	04	012	Alicún	264	
18	01	04	013	Almería	92.854	
19	01	04	014	Almócita	119	
20	01	04	015	Alsodux	17	
21	01	04	016	Antas	2.378	
22	01	04	017	Arboleas	842	
23	01	04	018	Armuña de Almanzora	0	
24	01	04	019	Bacares	257	
25	01	04	020	Bayárcal	75	
26	01	04	021	Bayarque	32	
27	01	04	022	Béjar	0	
28	01	04	023	Beires	10	
29	01	04	024	Benahadux	0	
30	01	04	026	Benitagla	0	
31	01	04	027	Benízalón	28	
32	01	04	028	Bentarique	419	
33	01	04	029	Berja	9.072	
34	01	04	030	Canjáyar	111	

2. Preparar el fichero para su procesamiento

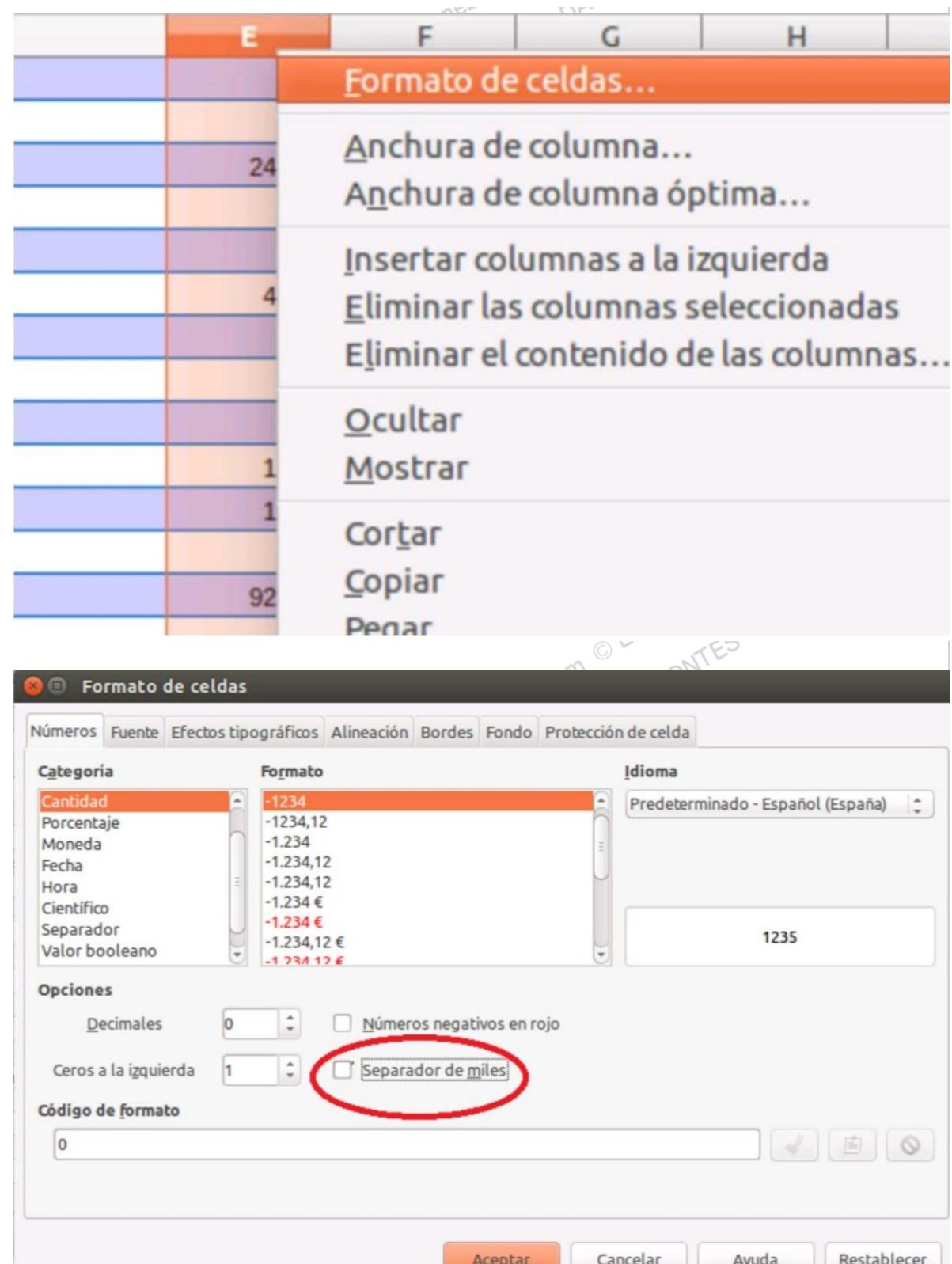
Lo siguiente que haremos es preparar el fichero para poder procesarlo. En la mayoría de los ejercicios de procesamiento de datos es necesario realizar primero una fase de limpieza y preparación de los datos.

En este caso solo tendremos que eliminar las cinco primeras líneas para quedarnos con las filas que poseen datos de análisis.

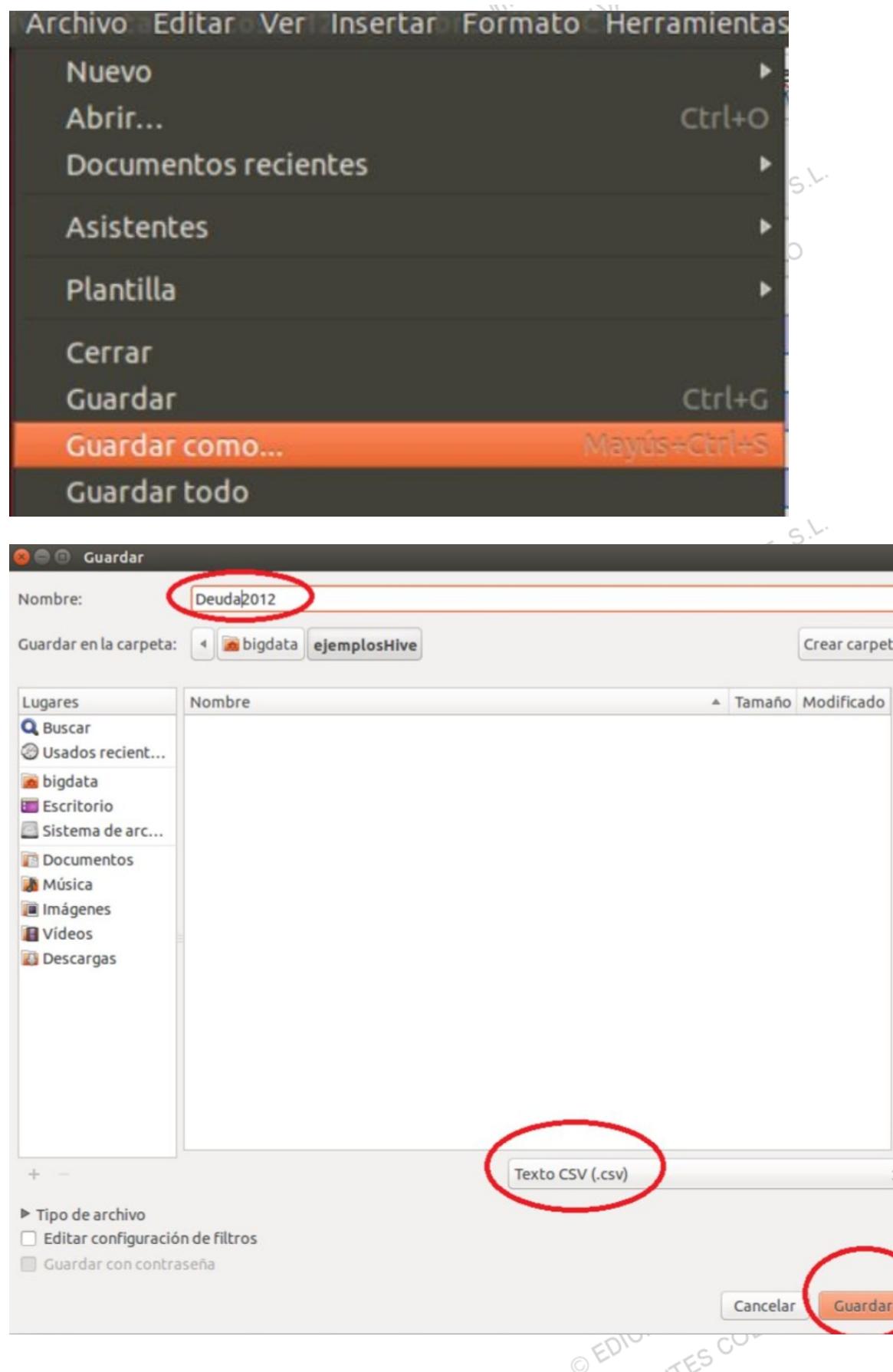
También las últimas filas (pulsar CTRL + FIN).

	A	B	C	D	E
8101	17	46	253	Vallés.	58
8102	17	46	254	Venta del Moro	310
8103	17	46	255	Villalonga	3.073
8104	<u>Formato de celdas...</u>				2.456
8105	<u>Altura de fila...</u>			stellón	1.992
8106	<u>Altura óptima de fila...</u>			o	1.993
8107	<u>Insertar filas arriba</u>			briel	0
8108	Eliminar las filas seleccionadas				1.056
8109	<u>Eliminar contenido de las filas...</u>				500
8110	<u>Ocultar</u>				96
8111	<u>Mostrar</u>				136
8112	<u>Cortar</u>				435
8113	<u>Copiar</u>				5.967
8114	<u>Pegar</u>				0
8115	<u>Pegado especial...</u>				269.541
8116	rcios, Fundaciones, asociaciones, etc) que no				113.554
8117	ntrol efectivo				59.786
8118	Diferencias de conciliacion con el Banco de España				67.032
8119					TOTAL 35.290.059

Formatearemos los importes de deudas eliminando los separadores de miles y seleccionando toda la columna.



Lo guardamos con la opción “Guardar como”:

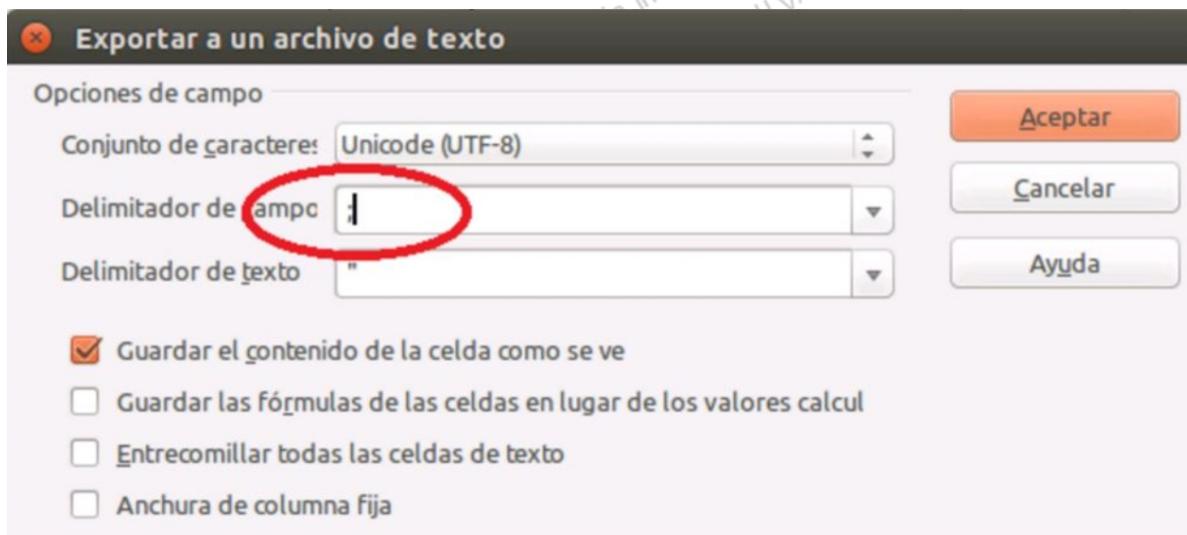


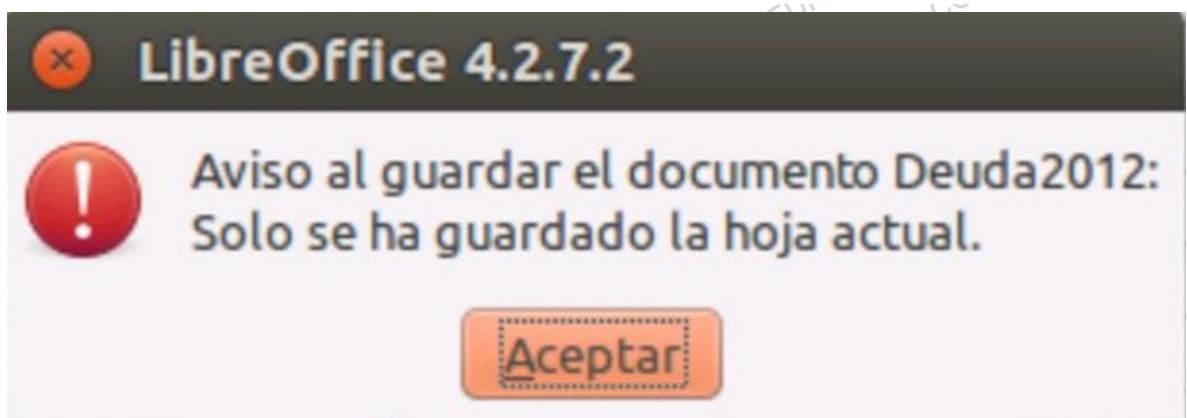
Como fichero csv en la ruta /home/bigdata/ejemplosHive con el nombre Deuda2012.csv.

Si la carpeta no existe, la creamos pulsando en "Crear Carpeta".



Lo guardaremos en modo CSV y el delimitador empleado lo cambiaremos por un ;





3. Arrancamos HIVE

Para ello, arrancaremos los demonios de HDFS y MapReduce y, después, escribiremos el comando:

```
$./bin/hive o $hive
```

```
bigdata@bigdata:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]

Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/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 engine (i.e. spark, tez) or using Hive 1.X releases
hive> [REDACTED]
```

4. Crear la tabla deuda_2012, en la que almacenaremos la información del fichero

- cod_comunidad: String
- cod_prov: String
- cod_municipio: String
- municipio: String
- deuda: int

```
CREATE TABLE deuda_2012(cod_comunidad STRING, cod_prov STRING, cod_municipio STRING, municipio STRING, deuda INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY
'(';
```

```
hive> CREATE TABLE deuda_2012(cod_comunidad STRING, cod_prov STRING, cod_municipio STRING, municipio STRING, deuda INT)
      > ROW FORMAT DELIMITED FIELDS TERMINATED BY ';';
OK
Time taken: 2.001 seconds
hive> [REDACTED]
```

5. Cargar el fichero Deuda2012.csv en la tabla anterior

```
LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/Deuda2012.csv' INTO TABLE deuda_2012;
```

```
hive> LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/Deuda2012.csv' INTO TABLE deuda_2012;
Loading data to table default.deuda_2012
OK
Time taken: 1.408 seconds
hive>
```

6. Ejecutar una select sin criterios de filtrado para consultar información de la tabla

```
SELECT deuda_2012.* FROM deuda_2012;
```

```
hive> SELECT deuda_2012.* FROM deuda_2012;
OK
01      04      001      Abla                      726
01      04      002      Abrucena                  235
01      04      003      Adra                     24027
01      04      004      Albánchez                 146
01      04      005      Alboloduy                77
01      04      006      Albox                     4453
01      04      007      Alcolea                   323
01      04      008      Alcóntar                  284
01      04      009      Alcudia de Monteagud    34
01      04      010      Alhabia                  1613
01      04      011      Alhama de Almería       1349
01      04      012      Alicún                    264
01      04      013      Almería                  92854
01      04      014      Almócita                 119
01      04      015      Alsodux                  17
01      04      016      Antas                     2378
01      04      017      Arboleas                  842
01      04      018      Armuña de Almanzora     0
01      04      019      Bacares                  257
01      04      020      Bayárcal                  75
01      04      021      Bayarque                  32
01      04      022      Bédar                     0
01      04      023      Beires                    10
01      04      024      Benahadux                 0
01      04      026      Benitagla                 0
01      04      027      Benizalón                 28
01      04      028      Bentarique                 419
01      04      029      Berja                     9072
01      04      030      Canjáyar                  111
01      04      031      Cantoria                 1412
01      04      032      Carboneras                 10935
01      04      033      Castro de Filabres      0
01      04      034      Cóbdar                     0
01      04      035      Cuevas del Almanzora    10023
01      04      036      Chercos                    0
01      04      037      Chirivel                  937
01      04      038      Dalias                     4028
```

7. Ejecutar una sentencia que permita filtrar algunos datos

```
SELECT * FROM deuda_2012 WHERE municipio LIKE 'Madri%';
```

```
hive> SELECT * FROM deuda_2012 WHERE municipio LIKE 'Madri%';
OK
07    05      114    Madrigal de las Altas Torres          13
07    09      196    Madrigal del Monte                  0
07    09      197    Madrigalejo del Monte                 0
07    49      103    Madridanos                      18
08    02      045    Madrigueras                     966
08    45      087    Madridejos                      5156
10    10      111    Madrigal de la Vera                  0
10    10      112    Madrigalejo                     0
12    28      079    Madrid                         7429664
Time taken: 0.915 seconds, Fetched: 9 row(s)
hive>
```

4.19. Almacenamiento de datos

Modos de almacenamiento:

- Almacenamiento en ficheros.
- Almacenamiento en registros.

4.19.1 Almacenamiento en ficheros

TEXTFILE (por defecto)

Almacena cada registro como una línea de texto. Es muy útil para compartir ficheros con otras aplicaciones y editar ficheros manualmente. Es menos eficiente que los ficheros de texto en formato binario al no emplear ningún tipo de estructura.

SEQUENCEFILE

Almacena el fichero como una secuencia única de pares claves-valores en binario que se guardan en Hadoop. Es bueno para compartir ficheros en el ecosistema de Hadoop.

RCFILE (HIVE 0.6.0 o superior)

Almacena las columnas de la tabla como registros en forma de columna. Permite buena compresión.

ORC (optimized row columnar file) (HIVE 0.11.0 o superior)

Nuevo formato optimizado que podrá sustituir a RCFILE.

PARQUET

PARQUET¹⁰ (HIVE 0.13.0 o superior): formato de almacenamiento en columnas disponible para cualquier proyecto del ecosistema de Hadoop.

¹⁰Apache Hive: “Parquet”. [En línea] URL disponible en: <https://cwiki.apache.org/confluence/display/Hive/Parquet>

AVRO

AVRO (HIVE 0.14.0 o superior): sistema de serialización de datos de Hadoop que pretende convertirse en un estándar. Se apoya en la definición de un esquema para almacenar los datos.

INPUTFORMAT

input_format_classname.

OUTPUTFORMAT

output_format_classname.



CREATE TABLE ejemplo1

STORED AS SEQUENCEFILE

CREATE TABLE ejemplo2

STORED AS

INPUTFORMAT org.apache.hadoop.mapred.SequenceFileInputFormat

OUTPUTFORMAT
org.apache.hadoop.hive.contrib.fileformat.base64.Base64TextInputFormat

Sintaxis	Equivalencia
STORED AS AVRO / STORED AS AVROFILE Hive 0.9.1	ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe' STORED as INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat' OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
STORED AS ORC / STORED AS ORCFILE Hive 0.11	ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.orc.OrcSerde' STORED as INPUTFORMAT 'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat' OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat'
STORED AS PARQUET / STORED AS PARQUETFILE Hive 0.13	ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerde' STORED as INPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.ParquetInputFormat' OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.ParquetOutputFormat'
STORED AS RCFILE	STORED as INPUTFORMAT 'org.apache.hadoop.hive.ql.io.RCFileInputFormat' OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.RCFileOutputFormat'
STORED AS TEXTFILE	STORED as INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat' OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
CSV Hive 0.14	'org.apache.hadoop.hive.serde2.OpenCSVSerde'
JSON Hive 0.12	'org.apache.hive.hcatalog.data.JsonSerDe'

Tabla 3. Almacenamiento en ficheros. Fuente: Fundación Apache.

4.19.2. Almacenamiento en registros

Para el almacenamiento de los datos en registros se emplea el SerDe¹¹ de HIVE. SerDe significa “serializador y deserializador”. Básicamente consiste en una clase Java que permite convertir los datos de entrada en datos de salida.

¹¹Para ampliar información sobre SerDe, consulta la documentación del proyecto HIVE: <https://cwiki.apache.org/confluence/display/Hive/DeveloperGuide#DeveloperGuide-HiveSerDe>. Acceso a la documentación sobre la API en: <https://hive.apache.org/javadocs/r1.2.2/api/>



Recuperar ficheros de HDFS en formato clave-valor y emplear un deserializador para devolver un objeto fila con el que podríamos trabajar.

Dado un objeto fila en HIVE, podemos emplear un serializador para convertir los pares claves-valor en un formato de salida que almacenaremos en HDFS.

Package org.apache.hadoop.hive.serde2

Interface Summary

Interface	Description
ByteStream.RandomAccessOutput	
Deserializer	Deprecated
SerDe	Deprecated
SerDeStatsStruct	
Serializer	Deprecated
StructObject	

Class Summary

Class	Description
AbstractDeserializer	Abstract class for implementing Deserializer.
AbstractEncodingAwareSerDe	AbstractEncodingAwareSerDe aware the encoding for serialize, and transform data from UTF-8 to specified encoding.
AbstractSerDe	Abstract class for implementing SerDe.
AbstractSerializer	Abstract class for implementing Serializer.
BaseStructObjectInspector	
BaseStructObjectInspector.MyField	
ByteStream	Extensions to bytearrayinput/output streams.
ByteStream.Input	Input.
ByteStream.Output	

1

HIVE aprovecha el uso de clases Java para formatear los datos de entrada y salida SerDe:

- `TextInputFormat` / `HiveIgnoreKeyTextOutputFormat`: estas clases se emplean para leer y escribir datos en formato de texto plano.
- `SequenceFileInputFormat` / `SequenceFileOutputFormat`: estas dos clases se emplean para leer y escribir datos en el formato SEQUENCEFILE.
- `MetadataTypedColumnsetSerDe`: se emplea para leer y escribir registros separados por delimitadores, como CSV o tabuladores.
- `LazySimpleSerDe`: se emplea para leer el mismo formato que `MetadataTypedColumnsetSerDe` y `TCTLSeparatedProtocol`, pero crea objetos en un modo "vago", lo que proporciona un mejor rendimiento. Desde la versión 0.14 permite indicar el tipo de encoding:

2

```
ALTER TABLE person SET SERDEPROPERTIES ('serialization.encoding'='ISO-8859-1');
```

- ThriftSerDe: Se emplea para leer y escribir objetos serializados en Thrift.
- DynamicSerDe: se emplea para leer y escribir objetos serializados en Thrift, pero a diferencia del anterior, permite comprender el formato del esquema de Thrift. También proporciona conversión con otros protocolos incluyendo TBinaryProtocol, TJSONProtocol y TCTLSeparatedProtocol.
- Amazon proporciona un SerDe para leer JSON s3://elasticmapreduce/samples/hive-ads/libs/jsonserde.jar.
- AvroSerDe desde la versión 0.9
- ORC desde la versión 0.11
- Parquet desde la versión 0.13
- CSV desde la versión 0.14

3

Conversión de datos:

```
CREATE TABLE nuevaTablaEnFormatoSEQUENCE (.....)
```

```
STORED AS SEQUENCEFILE;
```

```
INSERT OVERWRITE TABLE nuevaTablaEnFormatoSEQUENCE
```

```
SELECT * FROM tablaAntiguaFormatoTEXTO;
```

4

Compresión¹²:

- GZIP o BZip2.
- LZO: nuevo formato de compresión que favorece la velocidad frente a la tasa de compresión.

```
CREATE TABLE raw (line STRING)
```

```
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n';
```

```
LOAD DATA LOCAL INPATH '/tmp/weblogs/20141220-access.log.gz' INTO TABLE raw;
```

```
CREATE TABLE raw_sequence (line STRING) STORED AS SEQUENCEFILE;
```

```
SET hive.exec.compress.output=true;
```

```
SET io.seqfile.compression.type=BLOCK;
```

```
INSERT OVERWRITE TABLE raw_sequence SELECT * FROM raw;
```

¹²Documentación sobre compresión del proyecto HIVE:

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+LZO#LanguageManualLZO-LZOCompression>

4.20. HIVE y UDF¹³

¹³Documentacion sobre UDF: <https://cwiki.apache.org/confluence/display/Hive/HivePlugins>

Caso de estudio: “Escibiendo UDAF genéricas”: <https://cwiki.apache.org/confluence/display/Hive/GenericUDAFCaseStudy>

1

1. Crear estructura de directorios y clase java:

```
mkdir -p /home/bigdata/ejemplosHive/udf/com/ejemplo/hive/udf
```

```
sudo nano /home/bigdata/ejemplosHive/udf/com/ejemplo/hive/udf/Lower.java
```

```
bigdata@bigdata:~$ mkdir -p /home/bigdata/ejemplosHive/udf/com/ejemplo/hive/udf
bigdata@bigdata:~$ sudo nano /home/bigdata/ejemplosHive/udf/com/ejemplo/hive/udf/Lower.java
```

2

2. Crear código de UDF:

Para ver el código que necesitas, pulsa [aquí](#).

```
GNU nano 2.2.6 Archivo: ...jemplo/hive/udf/Lower.java
package com.ejemplo.hive.udf;
import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.io.Text;
public final class Lower extends UDF {
    public Text evaluate(final Text s) {
        if (s == null) { return null; }
        return new Text(s.toString().toLowerCase());
    }
}
```

3

3. Compilar fichero:

```
cd /home/bigdata/ejemplosHive/udf/  
sudo javac -classpath $HADOOP_HOME/share/hadoop/common/hadoop-common-  
2.8.0.jar:$HIVE_HOME/lib/hive-exec-2.2.0.jar com/ejemplo/hive/udf/Lower.java  
sudo jar cf /home/bigdata/ejemplosHive/udf/Lower.jar com/ejemplo/hive/udf/*.class
```

ls

```
bigdata@bigdata:~$ cd /home/bigdata/ejemplosHive/udf/  
bigdata@bigdata:~/ejemplosHive/udf$ sudo javac -classpath /home/bigdata/hadoop/share/hadoop/commo-  
n/hadoop-common-2.8.0.jar:/home/bigdata/hive/lib/hive-exec-2.2.0.jar com/ejemplo/hive/udf/Lower.j-  
ava  
bigdata@bigdata:~/ejemplosHive/udf$ sudo jar cf /home/bigdata/ejemplosHive/udf/Lower.jar com/ejem-  
plo/hive/udf/*.class  
bigdata@bigdata:~/ejemplosHive/udf$ ls  
com_Lower.jar  
bigdata@bigdata:~/ejemplosHive/udf$ █
```

4

4. Acceder a HIVE:

\$hive

```
bigdata@bigdata:~$ hive  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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]  
Logging initialized using configuration in jar:file:/home/bigdata/hive/lib/hive-common-2.2.0.jar!/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 engine (i.e. spark, tez) or using Hive 1.X releases  
hive> █
```

5

5. Registrar el jar:

```
hive> add jar /home/bigdata/ejemplosHive/udf/Lower.jar;
```

```
hive> add jar /home/bigdata/ejemplosHive/udf/Lower.jar;  
Added [/home/bigdata/ejemplosHive/udf/Lower.jar] to class path  
Added resources: [/home/bigdata/ejemplosHive/udf/Lower.jar]
```

6

6. Crear una tabla temporal y asignarle un par de registros:

```
hive> CREATE TABLE ejemploLower(cadena String);
```

```
hive> CREATE TABLE ejemploLower(cadena String);
OK
Time taken: 0.462 seconds
```

```
hive> INSERT INTO ejemploLower(cadena) values
```

```
('CADENA1'),('Cadena2');
```

```
hive> CREATE TABLE ejemploLower(cadena String);
OK
Time taken: 1.747 seconds
hive> INSERT INTO ejemploLower(cadena) values ('CADENA1'),('Cadena2');
Warning: live-on-HR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hive_20170903123333_015124e0-3bbc-4c29-a33b-ff47e50e5762
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1503850422017_0004, Tracking URL = http://bigdata:8088/proxy/application_1503850422017_0004/
Kill Command = /home/bigdata/hadoop/bin/hadoop job -kill job_1503850422017_0004
Hadoop Job Information for Stage-1: number of mappers: 1; number of reducers: 0
2017-09-03 12:33:47,248 Status: map = 0%, reduce = 0%
2017-09-03 12:33:56,262 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.11 sec
MapReduce Total cumulative CPU time: 1 seconds 110 msec
Ended Job = job_1503850422017_0004
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-2 is filtered out by condition resolver.
Moving data to table default.ejemploLower
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Cumulative CPU: 1.11 sec  HDFS Read: 3811 HDFS Write: 92 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 110 msec
OK
Time taken: 25.293 seconds
hive>
```

7

7. Registrar la función:

```
hive> create temporary function my_lower as 'com.ejemplo.hive.udf.Lower';
```

```
hive> create temporary function my_lower as 'com.ejemplo.hive.udf.Lower';
OK
Time taken: 0.01 seconds
```

8

8. Usar la nueva función:

```
hive> select my_lower(cadena), cadena from ejemploLower;
```

```
hive> select my_lower(cadena), cadena from ejemploLower;
OK
cadena1 CADENA1
cadena2 Cadena2
Time taken: 0.228 seconds, Fetched: 2 row(s)
```

4.21. Lectura recomendada



Rutherford, Jason; Wampler, Dean; Capriolo, Edward. *Programming Hive*. Sebastopol (CA): O'Reilly Media; 2012 (ISBN: 9781449326944)¹⁴.

Esta guía completa presenta Apache Hive, la infraestructura de *data warehouse* de Hadoop. Con ella aprenderás rápidamente a usar el dialecto SQL de Hive-HiveQL para resumir, consultar y analizar grandes conjuntos de datos almacenados en el sistema de archivos distribuido de Hadoop. La guía está basada en ejemplos y muestra cómo configurar HIVE, proporciona una descripción detallada de Hadoop y MapReduce y demuestra cómo funciona HIVE dentro del ecosistema de Hadoop.

¹⁴Se puede consultar el índice de contenidos y adquirir la obra en la página web de la editorial: <https://www.safaribooksonline.com/library/view/programming-hive/9781449326944/>

V. SQUOOP

1



SQuoop es una herramienta de transferencia de datos entre sistemas estructurados como los sistemas de bases de datos relacionales y sistemas de Hadoop con datos semiestructurados (Cassandra o HBase) o desestructurados (HDFS).

```
$ sqoop help
usage: sqoop COMMAND [ARGS]

Available commands:
  codegen          Generate code to interact with database records
  create-hive-table Import a table definition into Hive
  eval             Evaluate a SQL statement and display the results
  export           Export an HDFS directory to a database table
  help             List available commands
  import           Import a table from a database to HDFS
  import-all-tables Import tables from a database to HDFS
  list-databases   List available databases on a server
  list-tables      List available tables in a database
  version          Display version information

See 'sqoop help COMMAND' for information on a specific command.
```

2

SQuoop emplea JDBC para realizar las conexiones a base de datos. Incorpora algunas librerías de conexión pero se pueden añadir nuevas incorporándolas a la carpeta \$SQUOOP_HOME/lib.

The screenshot shows the top navigation bar of the Apache Sqoop website. It includes links for "Snoop", "Project Information", "Releases", "Documentation", "ASF", "External Links", and a "Google Custom Search" bar. The URL in the address bar is "sqoop.apache.org".



The Apache Software Foundation <http://www.apache.org/>

Apache / Sqoop /

Last Published: 2015-09-01

Apache Sqoop

Apache Sqoop(TM) is a tool designed for efficiently transferring bulk data between [Apache Hadoop](#) and structured datastores such as relational databases.

Sqoop successfully graduated from the Incubator in March of 2012 and is now a Top-Level Apache project: [More information](#)

Latest stable release is 1.4.6 ([download](#), [documentation](#)). Latest cut of Sqoop2 is 1.99.6 ([download](#), [documentation](#)). Note that 1.99.6 is not compatible with 1.4.6 and not feature complete, it is not intended for production deployment.



We suggest the following mirror site for your download:

<http://apache.uvigo.es/sqoop/>

Other mirror sites are suggested below. Please use the backup mirrors only to download PGP and MDS signatures to verify your downloads or if no other mirrors are working.

HTTP

<http://apache.rediris.es/sqoop/>

<http://apache.uvigo.es/sqoop/>

<http://ftp.cixug.es/apache/sqoop/>

Index of /sqoop

Name	Last modified	Size	Description
	Parent Directory	-	
	1.4.6/	-	2017-06-26 18:19
	1.99.7/	-	2017-06-26 18:19

3

Descargaremos Sqoop mediante el comando:

```
wget https://archive.apache.org/dist/sqoop/1.4.6/sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz
```

Index of /sqoop/1.4.6

Name	Last modified	Size	Description
 Parent Directory		-	
 sqoop-1.4.6.bin__hadoop-0.23.tar.gz	2015-05-08 10:28	16M	
 sqoop-1.4.6.bin__hadoop-1.0.0.tar.gz	2015-05-08 10:28	16M	
 sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz	2015-05-08 10:28	16M	
 sqoop-1.4.6.tar.gz	2015-05-08 10:28	2.1M	

```
bigdata@bigdata:~$ wget http://apache.rediris.es/sqoop/1.4.6/sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz
--2015-05-08 10:28:23-- http://apache.rediris.es/sqoop/1.4.6/sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz
Recibiendo apache.rediris.es (apache.rediris.es)[130.206.13.2]:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 16870735 (16M) [application/x-gzip]
Guardando como: "sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz"
sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz    70%[=====] 11,29M  2,57MB/s  eta 2s
```

4

Una vez descargado, procederemos a su descompresión:

```
$ cd /home/bigdata
```

```
$ tar -xvf sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz
```

```
bigdata@bigdata:~$ tar -xvf sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz
```

Lo moveremos a otra ruta para independizarlo de la versión:

```
mv sqoop-1.4.6-bin_hadoop2.0.4-alpha /home/bigdata/sqoop
```

```
rm sqoop-1.4.6-bin_hadoop2.0.4-alpha.tar.gz
```

```
bigdata@bigdata:~$ mv sqoop-1.4.6-bin_hadoop2.0.4-alpha /home/bigdata/sqoop  
bigdata@bigdata:~$ rm sqoop-1.4.6-bin_hadoop2.0.4-alpha.tar.gz
```

Editamos el fichero `~/.bashrc`:

```
sudo nano ~/.bashrc
```

```
bigdata@bigdata:~$ sudo nano ~/.bashrc
```

Y añadimos al final:

```
#Sqoop START  
  
export SQOOP_HOME=/home/bigdata/sqoop  
  
export PATH=$PATH:$SQOOP_HOME/bin  
  
#Sqoop END
```

```
#Sqoop START  
export SQOOP_HOME=/home/bigdata/sqoop  
export PATH=$PATH:$SQOOP_HOME/bin  
#Sqoop END
```

```
source ~/.bashrc
```

```
bigdata@bigdata:~$ source ~/.bashrc
```

El comando de ayuda de Sqoop es:

```
sqoop help
```

```

bigdata@bigdata:~$ sqoop help
Warning: /home/bigdata/sqoop/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /home/bigdata/sqoop/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/bigdata/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/bigdata/sqoop/../zookeeper does not exist! Zookeeper imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
17/09/03 12:45:18 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6
usage: sqoop COMMAND [ARGS]

Available commands:
  codegen          Generate code to interact with database records
  create-hive-table Import a table definition into Hive
  eval             Evaluate a SQL statement and display the results
  export            Export an HDFS directory to a database table
  help              List available commands
  import            Import a table from a database to HDFS
  import-all-tables Import tables from a database to HDFS
  import-mainframe  Import datasets from a mainframe server to HDFS
  job               Work with saved jobs
  list-databases   List available databases on a server
  list-tables       List available tables in a database
  merge             Merge results of incremental imports
  metastore         Run a standalone Sqoop metastore
  version           Display version information

See 'sqoop help COMMAND' for information on a specific command.

```

cd sqoop/conf

ls -lrt

```

bigdata@bigdata:~$ cd sqoop/conf
bigdata@bigdata:~/sqoop/conf$ ls -lrt
total 28
-rw-r--r-- 1 bigdata bigdata 5531 abr 27 2015 sqoop-site.xml
-rw-r--r-- 1 bigdata bigdata 5531 abr 27 2015 sqoop-site-template.xml
-rwxr-xr-x 1 bigdata bigdata 1345 abr 27 2015 sqoop-env-template.sh
-rw-r--r-- 1 bigdata bigdata 1404 abr 27 2015 sqoop-env-template.cmd
-rw-r--r-- 1 bigdata bigdata 3895 abr 27 2015 oraoop-site-template.xml
bigdata@bigdata:~/sqoop/conf$

```

cp sqoop-env-template.sh sqoop-env.sh

```
[bigdata@bigdata:~/sqoop/conf$ cp sqoop-env-template.sh sqoop-env.sh
```

sudo nano sqoop-env.sh

```
bigdata@bigdata:~/sqoop/conf$ sudo nano sqoop-env.sh
```

Cambiar:

```
#Set path to where bin/hadoop is available
#export HADOOP_COMMON_HOME=

#Set path to where hadoop-* core.jar is available
#export HADOOP_MAPRED_HOME=

#set the path to where bin/hbase is available
#export HBASE_HOME=
```

Por:

```
#Set path to where bin/hadoop is available
export HADOOP_COMMON_HOME=$HADOOP_HOME

#Set path to where hadoop-* core.jar is available
export HADOOP_MAPRED_HOME=$HADOOP_HOME

#set the path to where bin/hbase is available
export HBASE_HOME=$HBASE_HOME
```

```
#Set path to where bin/hadoop is available
export HADOOP_COMMON_HOME=$HADOOP_HOME

#Set path to where hadoop-* core.jar is available
export HADOOP_MAPRED_HOME=$HADOOP_HOME

#set the path to where bin/hbase is available
export HBASE_HOME=$HBASE_HOME
```

sudo nano ~/.bashrc

bigdata@bigdata:~\$ sudo nano ~/.bashrc

```
#HCAT START

export HCAT_HOME=$HIVE_HOME/hcatalog/
export PATH=$HCAT_HOME/bin:$PATH

#HCAT END
```

```
#HCAT START
export HCAT_HOME=$HIVE_HOME/hcatalog/
export PATH=$HCAT_HOME/bin:$PATH
#HCAT END
```

source ~/.bashrc

```
bigdata@bigdata:~$ source ~/.bashrc
```

5

Antes de ejecutar el ejemplo, necesitamos instalar varios componentes de software. El primero, el *driver* necesario para conectarnos a MySQL; para ello nos situamos en:

```
cd /home/bigdata/sqoop/lib
```

```
bigdata@bigdata:~$ cd /home/bigdata/sqoop/lib
bigdata@bigdata:~/sqoop/lib$
```

<http://mvnrepository.com/artifact/mysql/mysql-connector-java/5.1.38>

The screenshot shows the Maven Repository page for the MySQL Connector/J 5.1.38 artifact. It includes the MySQL logo, the artifact name, version, and type. Below the artifact details, there are links for 'Download (JAR)' and 'View'.

Artifact	Download (JAR) (961 KB)
POM File	View
Date	(Dec 02, 2015)

Y descargamos el *driver* en ese directorio:

```
wget http://central.maven.org/maven2/mysql/mysql-connector-java/5.1.38/mysql-connector-java-5.1.38.jar
```

```
bigdata@bigdata:~/sqoop/lib$ wget http://central.maven.org/maven2/mysql/mysql-connector-java/5.1.38/mysql-connector-java-5.1.38.jar
--2017-09-03 12:50:56-- http://central.maven.org/maven2/mysql/mysql-connector-java/5.1.38/mysql-connector-java-5.1.38.jar
Resolving central.maven.org (central.maven.org)... 151.101.132.209
Connecting to central.maven.org (central.maven.org)|151.101.132.209|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 983911 (961K) [application/java-archive]
Saving to: ‘mysql-connector-java-5.1.38.jar’

100%[=====] 960,85K 2,27MB/s   in 0,4s
2017-09-03 12:51:00 (2,27 MB/s) - “mysql-connector-java-5.1.38.jar” guardado [983911/983911]
bigdata@bigdata:~/sqoop/lib$
```

6

En este punto, y antes de conectarnos, deberemos instalar MySQL en nuestra máquina virtual. Para ello ejecutaremos los siguientes comandos (usar como clave para el usuario root “1234”):

```
sudo apt-get update
```

```
bigdata@bigdata:~$ sudo apt-get update
```

sudo apt-get install mysql-server-5.7

```
bigdata@bigdata:~$ sudo apt-get install mysql-server-5.7
```

Ahora ya podremos conectarnos con nuestro servidor de MySQL.

mysql -u root -p

Password vacía



NOTA: Si el acceso a mysql no funciona probar las siguientes opciones:

- Opción 1: sudo mysql -u root
- Opción 2
 - sudo mysql -u root
 - UPDATE user SET plugin='mysql_native_password' WHERE User='root';
 - FLUSH PRIVILEGES;
 - exit;
 - service mysql restart (Entrar desde sudo mysql -u root)

```
bigdata@bigdata:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.7.19-0ubuntu0.17.04.1 (Ubuntu)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

mysql > use mysql;

mysql > update user set authentication_string=PASSWORD("1234") where User='root';

mysql > flush privileges;

mysql > quit

```
service mysql restart
```

```
mysql -u root -p
```

```
mysql> create database sqoop;
```

```
mysql> create database sqoop;
Query OK, 1 row affected (0.00 sec)
```

```
mysql> use sqoop;
```

```
mysql> use sqoop
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
```

```
mysql> create table test (cadena varchar(50));
```

```
mysql> create table test (cadena varchar(50));
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> insert into test (cadena) values ('Cadena1'), ('cadena2');
```

```
mysql> insert into test (cadena) values ('Cadena1'), ('cadena2');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

```
mysql> select * from test;
```

```
mysql> select * from test;
+-----+
| cadena |
+-----+
| Cadena1 |
| cadena2 |
+-----+
2 rows in set (0.00 sec)
```

```
mysql> exit
```

```
mysql> exit;
Bye
biqdata@biqdata:~/sqoop/lib$
```

Ayuda sobre el comando import de Sqoop:

```
bigdata@bigdata:~$ sqoop help import
Warning: /home/bigdata/sqoop/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /home/bigdata/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/bigdata/sqoop/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.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.slf4j.impl.Log4jLoggerFactory]
17/09/03 13:05:16 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6
usage: sqoop import [GENERIC-ARGS] [TOOL-ARGS]

Common arguments:
  --connect <jdbc-uri>                      Specify JDBC connect
                                                string
  --connection-manager <class-name>            Specify connection manager
                                                class name
  --connection-param-file <properties-file>    Specify connection
                                                parameters file
  --driver <class-name>                         Manually specify JDBC
                                                driver class to use
  --hadoop-home <dir>                          Override
                                                $HADOOP_MAPRED_HOME_ARG
  --hadoop-mapred-home <dir>                  Override
                                                $HADOOP_MAPRED_HOME_ARG
  --help                                         Print usage instructions
  -P                                           Read password from console
  --password <password>                        Set authentication
                                                password
  --password-alias <password-alias>             Credential provider
                                                password alias
  --password-file <password-file>              Set authentication
                                                password file path
  --relaxed-isolation                           Use read-uncommitted
```

Para ver el código que necesitas, pulsa [aquí](#).

5.1. Ejemplo

Una vez realizadas todas las configuraciones, podremos ejecutar el comando import. Esto dará lugar a una tarea Map Reduce que dejará la información de la tabla test en un archivo dentro de HDFS:

```
sqoop import --connect jdbc:mysql://localhost/sqoop --username
root --password 1234 --table test -m 1
```

```
bigdata@bigdata:~/sqoop/lbt$ sqoop import --connect jdbc:mysql://localhost/sqoop --username root --password 1234 --table test -m 1
Warning: /home/bigdata/sqoop/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /home/bigdata/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/bigdata/sqoop/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hive/lib/log4j-slf4j-impl-2.6.2.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.slf4j.impl.Log4jLoggerFactory]
17/09/03 13:08:09 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6
```

```

17/09/03 13:08:36 INFO mapreduce.Job: Job job_1503850422017_0005 completed successfully
17/09/03 13:08:36 INFO mapreduce.Job: Counters: 30
  File System Counters
    FILE: Number of bytes read=0
    FILE: Number of bytes written=154203
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=87
    HDFS: Number of bytes written=16
    HDFS: Number of read operations=4
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
  Job Counters
    Launched map tasks=1
    Other local map tasks=1
    Total time spent by all maps in occupied slots (ms)=4185
    Total time spent by all reduces in occupied slots (ms)=0
    Total time spent by all map tasks (ms)=4185
    Total vcore-milliseconds taken by all map tasks=4185
    Total megabyte-milliseconds taken by all map tasks=4285440
  Map-Reduce Framework
    Map input records=2
    Map output records=2
    Input split bytes=87
    Spilled Records=0
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ms)=55
    CPU time spent (ms)=620
    Physical memory (bytes) snapshot=128954368
    Virtual memory (bytes) snapshot=1944322048
    Total committed heap usage (bytes)=62980096
  File Input Format Counters
    Bytes Read=0
  File Output Format Counters
    Bytes Written=16
17/09/03 13:08:36 INFO mapreduce.ImportJobBase: Transferred 16 bytes in 20,7971 seconds (0,7693 bytes/sec)
17/09/03 13:08:36 INFO mapreduce.ImportJobBase: Retrieved 2 records.
bigdata@bigdata:~/sqoop/lib$ 

```

hdfs dfs -ls -R /user/bigdata/test

```

bigdata@bigdata:~/sqoop/lib$ hdfs dfs -ls -R /user/bigdata/test
-rw-r--r-- 1 bigdata supergroup 0 2017-09-03 13:08 /user/bigdata/test/_SUCCESS
-rw-r--r-- 1 bigdata supergroup 16 2017-09-03 13:08 /user/bigdata/test/part-m-00000

```

hdfs dfs -cat /user/bigdata/test/part-m-00000

```

bigdata@bigdata:~/sqoop/lib$ hdfs dfs -cat /user/bigdata/test/part-m-00000
Cadena1
cadena2

```

5.2. Sqoop: conexión

Podemos automatizar las tareas de carga de datos a través de un *script* como en el ejemplo siguiente:

```

sqoop import --connect jdbc:mysql://localhost/mydb --username root --password 1234 --table test

sqoop --options-file /home/bigdata/sqoop_imports/import.txt --password 1234 --table TEST

cat /home/bigdata/sqoop_imports/import.txt

```

```
bigdata@bigdata:~$ cat /home/bigdata/sqoop_imports/import.txt
#
# Opciones de importacion de Sqoop
#
#Especificamos la herramienta invocada
import

#Parametros de conexion y valor
--connect
jdbs:mysql://localhost/test
--username
root

#El resto de parametros deben ser indicado por linea de comandos
```

```
sqoop import --connect jdbc:mysql://localhost/mydb --username usuario --password-file ${user.home}/.password --table TEST
```

5.3. Sqoop: migración

Parámetros que nos permitirán parametrizar las migraciones:

Argumento	Descripción
--append	Anexa datos a un conjunto de datos existente en HDFS
--as-avrodatafile	Importa datos a los archivos de datos Avro
--as-sequencefile	Importa datos a SequenceFiles
--as-textfile	Importa datos como texto sin formato (predeterminado)
--boundary-query <statement>	Consulta de límite para crear divisiones
--columns <col,col,col...>	Columnas para importar de la tabla
--direct	Utiliza la ruta rápida de importación directa
--direct-split-size <n>	Divide la secuencia de entrada cada n bytes al importar en modo directo
--inline-lob-limit <n>	Establece el tamaño máximo para un LOB en línea
-m,--num-mappers <n>	Usa n tareas map para importar en paralelo
-e,--query <statement>	Importa los resultados de statement
--split-by <column-name>	Columna de la tabla utilizada para dividir las unidades de trabajo
--table <table-name>	Tabla para leer
--target-dir <dir>	Dir de destino HDFS
--warehouse-dir <dir>	Padre de HDFS para el destino de la tabla
--where <where clause>	WHERE; cláusula para usar durante la importación
-z,--compress	Habilita la compresión
--compression-codec <c>	Usa el códec de Hadoop (gzip predeterminado)
--null-string <null-string>	La cadena que se escribirá para un valor nulo para columnas de cadena
--null-non-string <null-string>	La cadena que se escribirá para un valor nulo para columnas que no sean cadenas

Tabla 4. Parámetros de migración. *Fuente:* Fundación Apache.

Para ver el código que necesitas, pulsa [aquí](#).

5.4. Sqoop: migraciones a otros sistemas

Parámetros que nos permitirán parametrizar las migraciones a otros sistemas como HIVE o HBase:

HIVE

Argumento	Descripción
--hive-home <dir>	Anula \$ HIVE_HOME
--hive-import	Importa tablas en Hive (usa los delimitadores por defecto de Hive si no hay ninguno configurado)
--hive-overwrite	Sobrescribe los datos existentes en la tabla Hive
--create-hive-table	Si se establece, el trabajo fallará si existe. Por defecto, esta propiedad es falsa
--hive-table <table-name>	Establece el nombre de la tabla que se utilizará al importar en Hive
--hive-drop-import-delims	Quita \n, \r, y \01 de los campos string al importar en Hive
--hive-delims-replacement	Reemplaza \n, \r, y \01 de los campos string por una cadena definida por el usuario al importar en Hive
--hive-partition-key	El nombre de un campo para particionar
--hive-partition-value <v>	String-value que sirve como clave de partición en este trabajo
--map-column-hive <map>	Anula la asignación predeterminada de tipo SQL a tipo Hive para las columnas configuradas

Tabla 5. Párametro de migración a HIVE. Fuente: Fundación Apache

HBase

Argumento	Descripción
--column-family <family>	Establece la familia de columnas de destino para la importación
--hbase-create-table	Si se especifica, crea tablas HBase faltantes
--hbase-row-key <col>	Especifica qué columna de entrada usar como clave de fila
--hbase-table <table-name>	Especifica una tabla HBase para usar como destino en lugar de HDFS

Tabla 6. Parámetros de migración a HBase. Fuente: Fundación Apache.

VI. HBASE

6.1. Introducción y objetivos

 Los objetivos de este epígrafe son:

- ➔ Conocer las características de HBase.
- ➔ Aprender a instalarlo sobre un servidor Hadoop y conocer su estructura de directorios.
- ➔ Familiarizarse con la terminología asociada a sus estructuras de datos.
- ➔ Conocer los comandos básicos disponibles.
- ➔ Conocer la existencia de APIs de comunicación o el modo de integración con otros sistemas como HIVE.
- ➔ Resolver ejercicios simples.



Apache HBase es una base de datos que se emplea cuando se necesita acceso aleatorio de lecturas y escrituras en tiempo real en Big Data. El objetivo del proyecto HBase es alojar grandes tablas (miles de millones de registros por millones de columnas) sobre servidores de bajo coste.

Más específicamente, Apache HBase es una base de datos distribuida desarrollada como un subproyecto de Hadoop y que usa HDFS como sistema de almacenamiento de archivos, que tiene escasa tolerancia a fallos cuando estamos almacenando grandes volúmenes de datos.

Con esta arquitectura es posible lograr no solo altos niveles de rendimiento, sino además altos niveles de redundancia en datos, ya que a cierto nivel los datos están distribuidos de manera redundante.

Aunque HBase representa una amenaza a largo plazo tanto para las bases de datos comerciales como Oracle, DB2 o MS SQL Server, como para las de código libre como Postgres o MySQL, hay que seguir teniendo en cuenta que hay situaciones en las que es mejor aún tener una base de datos relacional.

Sus principales características son:

Open source

Puede utilizarse libremente y además su código puede modificarse y mejorarse.

Implementado en Java

Puede funcionar en cualquier plataforma que tenga una máquina virtual Java, accesible desde distintos entornos como Unix, Linux, Windows y OSX.

Distribuida

Los datos se partitionan y fragmentan sobre múltiples servidores.

Escalable

Se pueden añadir nuevos servidores al clúster de manera automática en función de las necesidades.

Tolerancia a fallos

Al ser un subproyecto de Hadoop, permite la configuración de mecanismos de tolerancia a fallos.

Redundancia de datos

Asociado a lo anterior, incorpora redundancia de datos como mecanismo de control frente a errores.

HDFS

HBase permite el uso de HDFS como sistema de archivos distribuido.

Orientado a columnas

La información se almacena en celdas agrupadas en columnas que a su vez se agrupan en familias de columnas, y estas columnas pueden crearse en tiempo de ejecución. Además, los registros se identifican mediante una clave que relaciona una o varias columnas, dando lugar a una representación de datos en forma de mapas.

Fragmentación automática por regiones

Las tablas de datos se distribuyen de forma dinámica dentro del clúster en regiones que contienen valores entre una clave de inicio y una clave de fin. Además, incorpora recuperación automática frente a fallos de las regiones.

Map Reduce

HBase permite el procesamiento masivo de forma paralela empleando Map Reduce tanto como entrada como salida de los jobs.

Cacheo de bloques

Permite el cacheado de bloques de datos para la ejecución de consultas en tiempo real.

6.2. Terminología del modelo de datos de HBase

Antes de explicar el proceso de instalación de HBase y su sintaxis, vamos a explicar los conceptos básicos de su terminología.

NameSpace

Es la estructura empleada para agrupar tablas de modo lógico, similar a una base de datos en los sistemas de base de datos tradicionales. Por defecto, HBase define dos namespaces:

- hbase: se emplea para almacenar las tablas internas de HBase.
- default: se emplea para almacenar las tablas en las que no se indica su namespace.

Tabla

Una tabla de HBase está compuesta por múltiples filas.

Fila

Una fila en HBase consiste en una clave de fila y una o más columnas con valores asociados entre ellas. Las filas se ordenan alfabéticamente por la clave de la fila a medida que se van almacenando. Por esta razón, es importante diseñar la clave de la fila. El objetivo de este modo de almacenamiento del dato es que cada fila se relacione con las que tiene cerca.

Columna

Una columna en HBase consiste en una “Column Family” y una “Column Qualifier”, delimitadas por el carácter ‘;’.

Column Family

Almacena información de columnas y sus valores, a menudo por razones de rendimiento. Cada Column Family almacena un conjunto de propiedades, como por ejemplo si sus valores deben almacenarse en caché, si sus datos se comprimen o si las claves de la fila están codificadas. Cada fila en una tabla tiene las mismas Column Families, pero no todas las Column Family de una fila tienen que tener almacenada información.

Las Column Families se especifican en la creación de la tabla e influyen en el modo en el que los datos se almacenan en el sistema de archivos. Por ello las Column Families deben analizarse en detalle al diseñar el esquema.

Column Qualifier

Una Column Qualifier se agrega a una Column Family para proporcionar el índice para un determinado grupo de datos. Ejemplos de Column Qualifier: content:html, content:pdf...

Cell

Una celda es una combinación de fila, Column Family y Column Qualifier, y contiene un valor (array de bytes) y un timestamp que representa el valor de la versión del dato.

Timestamp

Con cada valor, se escribe un timestamp que identifica la versión del dato. Por defecto, el timestamp representa la hora de la región en la que el dato fue escrito, pero se puede especificar otro timestamp.

Región¹⁵

Son los elementos básicos de disponibilidad y distribución de las tablas. En general HBase está diseñado para correr sobre un número pequeño (entre 20 y 200) de regiones por servidor (de entre 5 y 20 Gb).

¹⁵Para más información sobre las regiones consultar:
<http://hbase.apache.org/book/regions.arch.html>

6.3. Instalación

A continuación se describe el proceso de descarga de HBase, así como los ficheros que se deben modificar para configurar correctamente la aplicación.

El primer paso de la descarga no es necesario realizarlo pues el fichero ya se encuentra descargado en la raíz del usuario bigdata en la máquina virtual.

1

Para poder instalar HBase, procederemos a seleccionar la última versión accesible desde sus ventanas de descarga:

The screenshot shows the Apache HBase Project homepage. At the top, there is a navigation bar with links to 'Apache HBase Project', 'Project Information', 'Documentation and API', and 'ASF'. On the right side of the header is a search bar labeled 'Google Custom Search' with a magnifying glass icon. Below the header, there is a banner for 'HBASECON ASIA 2017' and the text '2017 HBASE 亚洲技术大会'. To the right of the banner is the Apache HBase logo, which features the word 'APACHE' in red and 'HBASE' in larger red letters, with a silhouette of an orca swimming next to it.

Welcome to Apache HBase™

Apache™ HBase™ is the Hadoop™ database, a distributed, scalable, big data store.

Use Apache HBase™ when you need random, realtime read/write access to your Big Data. This project's goal is the hosting of very large tables – billions of rows X millions of columns – atop clusters of commodity hardware. Apache HBase is an open-source, distributed, versioned, non-relational database modeled after Google's [Bigtable: A Distributed Storage System for Structured Data](#) by Chang et al. Just as Bigtable leverages the distributed data storage provided by the Google File System, Apache HBase provides Bigtable-like capabilities on top of Hadoop and HDFS.

Download

Click [here](#) to download Apache HBase™.

Features

- Linear and modular scalability.
- Strictly consistent reads and writes.
- Automatic and configurable sharding of tables
- Automatic failover support between RegionServers.
- Convenient base classes for backing Hadoop MapReduce jobs with Apache HBase tables.
- Easy to use Java API for client access.
- Block cache and Bloom Filters for real-time queries.
- Query predicate push down via server side Filters
- Thrift gateway and a RESTful Web service that supports XML, Protobuf, and binary data encoding options
- Extensible ruby-based (JRuby) shell
- Support for exporting metrics via the Hadoop metrics subsystem to files or Ganglia; or via JMX

<http://www.apache.org/dyn/closer.cgi/hbase/>



We suggest the following mirror site for your download:

<http://ftp.cixug.es/apache/hbase/>

Other mirror sites are suggested below. Please use the backup mirrors only to download PGP and MD5 signatures to verify your downloads or if no other mirrors are working.

HTTP

<http://apache.rediris.es/hbase/>

<http://apache.uvigo.es/hbase/>

<http://ftp.cixug.es/apache/hbase/>



A vertical navigation menu for the Apache Software Foundation. It includes a 'Google Custom' search bar at the top, followed by four menu items: 'The Apache Way', 'Contribute', and 'ASF Sponsors'.

2

De las versiones estables, seleccionaremos un binario (descartamos las de código fuente para no compilarlas) y como estamos trabajando con Hadoop 2 cogeremos la versión hadoop2-bin.

HBase Releases

Please make sure you're downloading from a [nearby mirror site](#), not from www.apache.org.

We suggest downloading the current [stable](#) release.

The 1.2.x series is the current stable release line, it supercedes earlier release lines (the 1.1.x line is still seeing a regular cadence of bug fix releases for those who are not easily able to update). Note that: 0.96 was EOM'd September 2014; 1.0 was EOM'd January 2016; 0.94 and 0.98 were EOM'd April 2017.

For older versions, check the [apache archive](#).

Name	Last modified	Size	Description
Parent Directory			
1.1.12/	2017-08-22 06:06	-	
1.2.6/	2017-06-26 21:07	-	
1.3.1/	2017-06-26 21:07	-	
2.0.0-alpha-2/	2017-08-21 20:08	-	
hbase-thirdparty-1.0.1/	2017-08-28 21:17	-	
stable/	2017-06-26 21:07	-	

3

Accedemos a nuestra máquina Ubuntu y ejecutamos el siguiente comando:

```
 wget https://archive.apache.org/dist/hbase/1.2.6/hbase-1.2.6-bin.tar.gz
```

```
bigdata@bigdata:~$ wget http://apache.rediris.es/hbase/stable/hbase-1.2.6-bin.tar.gz
--2017-08-28 13:25:23-- http://apache.rediris.es/hbase/stable/hbase-1.2.6-bin.tar.gz
Resolviendo apache.rediris.es (apache.rediris.es) [130.206.13.2]
Conectando con apache.rediris.es (apache.rediris.es)[130.206.13.2]:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 104059474 (100M) [application/x-gzip]
Guardando como: "hbase-1.2.6-bin.tar.gz"

hbase-1.2.6-bin.tar.gz          1%>
```

Extraemos el fichero y cambiamos al nuevo directorio:

```
 tar xzvf hbase-1.2.6-bin.tar.gz
```

```
bigdata@bigdata:~$ tar xzvf hbase-1.2.6-bin.tar.gz
```

```
 mv hbase-1.2.6 hbase
```

```
 cd hbase
```

```
 ls
```

```

hbase-1.2.6/lib/hadoop-mapreduce-client-shuffle-2.5.1.jar
hbase-1.2.6/lib/hadoop-mapreduce-client-jobclient-2.5.1.jar
hbase-1.2.6/lib/commons-daemon-1.0.13.jar
hbase-1.2.6/lib/libthrift-0.9.3.jar
hbase-1.2.6/lib/jruby-complete-1.6.8.jar
hbase-1.2.6/lib/spymemcached-2.11.6.jar
bigdata@bigdata:~$ mv hbase-1.2.6 hbase
bigdata@bigdata:~$ cd hbase
bigdata@bigdata:~/hbase$ ls
bin  CHANGES.txt  conf  docs  hbase-webapps  LEGAL  lib  LICENSE.txt  NOTICE.txt  README.txt
bigdata@bigdata:~/hbase$ 

```

6.4. Configuración de ficheros

Una vez descargado y descomprimido, procederemos a editar los ficheros de configuración que se encuentran dentro de la carpeta conf. Básicamente editaremos el fichero en el que indicaremos la ruta a HBase y el directorio en el que se almacenarán los datos.

```
cd conf
```

```
ls
```

```

bigdata@bigdata:~/hbase$ cd conf
bigdata@bigdata:~/hbase/conf$ ls
hadoop-metrics2-hbase.properties  hbase-env.sh      hbase-site.xml    regionservers
hbase-env.cmd                      hbase-policy.xml  log4j.properties
bigdata@bigdata:~/hbase/conf$ 

```

```
sudo nano hbase-site.xml
```

```
bigdata@bigdata:~/hbase/conf$ sudo nano hbase-site.xml
```

Añadimos las propiedades:

```
<configuration>
</configuration>
```

```

<property>
<name>hbase.rootdir</name>
<value>file:///home/bigdata/hbase</value>
</property>
<property>
<name>hbase.zookeeper.property.dataDir</name>
<value>/home/bigdata/zookeeper</value>

```

```
</property>
```

```
bigdata@bigdata: ~/hbase/conf
GNU nano 2.2.6          Archivo: hbase-site.xml          Modificado

* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
-->
<configuration>
<property>
  <name>hbase.rootdir</name>
  <value>file:///home/bigdata/hbase</value>
</property>
<property>
  <name>hbase.zookeeper.property.dataDir</name>
  <value>/home/bigdata/zookeeper</value>
</property>
</configuration>
```

No es necesario crear el directorio de datos de zookeeper, ya que HBase lo creará cuando se arranque.

Incluimos en el classpath la información de las rutas en las que se encuentran las librerías de HBase, para después poder emplearlas.

```
sudo nano ~/.bashrc
```

```
bigdata@bigdata:~/hbase$ sudo nano ~/.bashrc
```

```
# Rutas de HBase

export HBASE_HOME=/home/bigdata/hbase

export PATH=$PATH:$HBASE_HOME/lib

export PATH=$PATH:$HBASE_HOME/bin

# Fin Rutas de HBase
```

```
# Rutas de HBase
export HBASE_HOME=/home/bigdata/hbase
export PATH=$PATH:$HBASE_HOME/lib
export PATH=$PATH:$HBASE_HOME/bin
# Fin Rutas de HBase
```

```
source ~/.bashrc
```

```
bigdata@bigdata:~/hbase/conf$ source ~/.bashrc
```

6.5. Arrancar HBase

Para poder arrancar Hbase, hay que ejecutar el siguiente comando:

```
bin/start-hbase.sh
```

```
bigdata@bigdata:~/hbase$ start-hbase.sh
starting master, logging to /home/bigdata/hbase/logs/hbase-bigdata-master-bigdata.out
OpenJDK 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0
OpenJDK 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.
```

Como resultado, se generará un *log* de errores en el que podremos consultar la información de los procesos que se han ejecutado.

Tras arrancarlo comprobaremos con el comando jps que el proceso está corriendo.

```
$ jps
```

```
bigdata@bigdata:~/hbase$ jps
31200 NodeManager
30657 DataNode
31058 ResourceManager
30904 SecondaryNameNode
31802 Jps
31661 HMaster
30511 NameNode
```

6.6. Arrancar *shell* de comandos

A continuación, accederemos a la *shell* y ejecutaremos el comando status:

```
./bin/hbase shell
```

```
status
```

```
bigdata@bigdata:~/hbase$ hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/hbase/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

hbase(main):001:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 2.0000 average load
hbase(main):002:0>
```

Para salir de la consola, ejecutaremos el comando exit pulsando intro.

A continuación, abriremos el *log* de errores y veremos las trazas que se han generado.

```
sudo nano logs/hbase-bigdata-master-bigdata.out
```

```
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/home/bigdata/hbase/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
```

Como vemos, aparece un *warning* debido a clases duplicadas en Hadoop common y en HBase. Para resolver este problema, eliminamos el jar en HBase y rearrancamos.

```
rm /home/bigdata/hbase/lib/slf4j-log4j12-1.7.5.jar
```

```
stop-hbase.sh
```

```
start-hbase.sh
```

```
hbase shell
```

```
exit
```

```
bigdata@bigdata:~/hbase$ rm /home/bigdata/hbase/lib/slf4j-log4j12-1.7.5.jar  
bigdata@bigdata:~/hbase$ stop-hbase.sh  
stopping hbase.....  
bigdata@bigdata:~/hbase$ start-hbase.sh  
starting master, logging to /home/bigdata/hbase/logs/hbase-bigdata-master-bigdata.out  
OpenJDK 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0  
OpenJDK 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0  
bigdata@bigdata:~/hbase$ hbase shell  
HBase Shell; enter 'help<RETURN>' for list of supported commands.  
Type "exit<RETURN>" to leave the HBase Shell  
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017  
  
hbase(main):001:0> exit  
bigdata@bigdata:~/hbase$
```

Podemos ver que, tras haber el eliminado el jar, ya no aparecen los *warnings* asociados a log4j.

Por último, comprobamos que se ha creado el directorio en el que se almacenará la información:

```
ls /home/bigdata/zookeeper
```

```
bigdata@bigdata:~/hbase$ ls /home/bigdata/zookeeper  
zookeeper_0  
bigdata@bigdata:~/hbase$
```

6.7. Zookeeper



En un sistema distribuido, es necesario mantener:

- Información de configuración.
- Nombres de máquinas.
- Sincronización entre los propios sistemas distribuidos.
- Servicios de grupos.

Características de Zookeeper:

- Interfaz sencilla para gestionar este servicio centralizado de coordinación.
- Servicio distribuido.
- Alta disponibilidad al estar disponible en varias máquinas.
- Posee un sistema de consenso o de *pooling* para determinar los nodos vivos.
- Incluye herramientas de gestión de grupos.
- Libera a las aplicaciones que gestionan de implementar su propio sistema de replicación y simplifica los cambios de configuración.

6.8. Estructura de directorios

A continuación, se describen brevemente los principales directorios que componen HBase

ls

```
bigdata@bigdata:~/hbase$ ls
bin  CHANGES.txt  conf  data  docs  hbase.id  hbase.version  hbase-webapps  LEGAL  lib
LICENSE.txt  logs  MasterProcWALS  NOTICE.txt  oldWALS  README.txt  WALS
bigdata@bigdata:~/hbase$
```

bin

Contiene los *scripts* facilitados por HBase para arrancar y parar HBase, así como otros ejecutables para lanzar demonios o arrancar nodos maestros adicionales.

```
bigdata@bigdata:~/hbase$ ls bin
draining_servers.rb  hbase-config.sh          region_mover.rb      stop-hbase.cmd
get-active-master.rb  hbase-daemon.sh         regionservers.sh    stop-hbase.sh
graceful_stop.sh     hbase-daemons.sh        region_status.rb    test
hbase                 hbase-jruby             replication           thread-pool.rb
hbase-cleanup.sh     hirb.rb                rolling-restart.sh  zookeepers.sh
hbase.cmd            local-master-backup.sh   shutdown_regionserver.rb
hbase-common.sh      local-regionservers.sh start-hbase.cmd
hbase-config.cmd     master-backup.sh       start-hbase.sh
```

conf

Contiene los ficheros que definen cómo está configurado HBase.

```
bigdata@bigdata:~/hbase$ ls conf
hadoop-metrics2-hbase.properties    hbase-env.sh      hbase-site.xml   regionservers
hbase-env.cmd                         hbase-policy.xml  log4j.properties
```

docs

Contiene una copia del proyecto web de HBase, incluyendo la documentación de las herramientas, API y el propio proyecto. Puedes consultar la información desde un navegador accediendo a: docs/index.html

```
bigdata@bigdata:~/hbase$ ls docs
acid-semantics.html      dependency-convergence.html metrics.html
apidocs                  dependency-info.html        modules.html
asciidocor.css            dependency-management.html old_news.html
book                     devapidocs                plugin-management.html
book.html                 distribution-management.html plugins.html
book.pdf                  doap_Hbase.rdf          project-info.html
book.pdfmarks             export_control.html       project-reports.html
bulk-loads.html           hbase.css                 project-summary.html
_chapters                images                   pseudo-distributed.html
checkstyle-aggregate.html index.html              replication.html
checkstyle.html            integration.html        resources.html
checkstyle.rss             issue-tracking.html   source-repository.html
coderay-asciidocor.css    js                      sponsors.html
css                      license.html           team-list.html
cygwin.html               mail-lists.html         xref
dependencies.html          ...                     xref-test
```

hbase-webapps

Contiene las interfaces de usuario implementadas como aplicaciones web Java.

```
bigdata@bigdata:~/hbase$ ls hbase-webapps/
master regionserver rest static thrift
bigdata@bigdata:~/hbase$
```

Por ejemplo, podemos acceder a la interfaz web de “master” accediendo al puerto 16010:

localhost:16010

The screenshot shows the Apache HBase master status interface. At the top, it displays the URL `localhost:16010/master-status`. Below the header, there's a navigation bar with links: Home, Table Details, Local Logs, Log Level, Debug Dump, Metrics Dump, and HBase Configuration. The main content area starts with "Master bigdata". Under "Region Servers", there's a table with columns: ServerName, Start time, Version, Requests Per Second, and Num. Regions. One entry is shown: `bigdata,40935,1504456766102` with start time `Sun Sep 03 18:39:26 CEST 2017`, version `1.2.6`, requests per second `0`, and num. regions `2`. A summary row "Total:1" has requests per second `0` and num. regions `2`. Below this is a section for "Backup Masters" with a table showing "Total:0". The final section is "Tables" with tabs for User Tables, System Tables, and Snapshots. The "User Tables" tab is selected.

lib

Librerías Java auxiliares necesarias para las distintas aplicaciones Java.

```
bigdata@bigdata:~/hbase$ ls lib
activation-1.1.jar
aopalliance-1.0.jar
apacheds-i18n-2.0.0-M15.jar
apacheds-kerberos-codec-2.0.0-M15.jar
api-asn1-api-1.0.0-M20.jar
api-util-1.0.0-M20.jar
asm-3.1.jar
avro-1.7.4.jar
commons-beanutils-1.7.0.jar
commons-beanutils-core-1.8.0.jar
commons-cli-1.2.jar
hbase-procedure-1.2.6.jar
hbase-protocol-1.2.6.jar
hbase-resource-bundle-1.2.6.jar
hbase-rest-1.2.6.jar
hbase-server-1.2.6.jar
hbase-server-1.2.6-tests.jar
hbase-shell-1.2.6.jar
hbase-thrift-1.2.6.jar
htrace-core-3.1.0-incubating.jar
httpclient-4.2.5.jar
httpcore-4.4.1.jar
```

logs

logs de los distintos demonios que lanza HBase.

```
bigdata@bigdata:~/hbase$ ls logs
hbase-bigdata-1-regionserver-bigdata.log      hbase-bigdata-master-bigdata.out.1
hbase-bigdata-1-regionserver-bigdata.out       hbase-bigdata-master-bigdata.out.2
hbase-bigdata-1-regionserver-bigdata.out.1     hbase-bigdata-master-bigdata.out.3
hbase-bigdata-1-regionserver-bigdata.out.2     hbase-bigdata-master-bigdata.out.4
hbase-bigdata-1-regionserver-bigdata.out.3     hbase-bigdata-master-bigdata.out.5
hbase-bigdata-master-bigdata.log               SecurityAuth.audit
hbase-bigdata-master-bigdata.out
```

src

Si se ha descargado el paquete binario, estarán disponibles los códigos fuente del proyecto HBase.

6.9. Modos de funcionamiento

HBase tiene tres modos de funcionamiento:

Standalone

HBase no usará el sistema de archivos HDFS sino que empleará el sistema de archivos local y se ejecutarán todos los demonios HBase y el Zookeeper en la misma JVM.

Pseudodistribuido

Es un modo distribuido simple para correr en un único servidor. Esta configuración se emplea para probar y para hacer prototipos de HBase. Los cambios en el fichero de configuración son los siguientes:

```
<configuration>
...
<property>
<name>hbase.rootdir</name>
<value>hdfs://localhost:9000/hbase</value>
</property>
<property>
<name>dfs.replication</name>
<value>1</value>
```

```
</property>
```

Completamente distribuido

Los demonios se distribuyen en diferentes nodos del clúster. Esta es la configuración empleada en producción o para evaluar el rendimiento de HBase.

```
<configuration>
```

```
...
```

```
<property>
```

```
<name>hbase.rootdir</name>
```

```
<value>hdfs://namenode.foo.com:9000/hbase</value>
```

```
</property>
```

```
<property>
```

```
<name>hbase.cluster.distributed</name>
```

```
<value>true</value>
```

```
</property>
```

```
...
```

```
</configuration>
```

6.10. Comandos disponibles en HBase shell

Como hemos hecho en otros casos, vamos a dividir los comandos¹⁶ en varios grupos:

- Comandos de uso general.
- Comandos de definición de datos (DDL).
- Comandos de manipulación de datos (DML).
- Comandos avanzados.

¹⁶Se puede encontrar información más detallada sobre los comandos en el siguiente enlace:<http://hbase.apache.org/book.html#shell>

6.10.1. Comandos generales

- status: muestra el estado del servidor.

```
hbase(main):001:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 2.0000 average load
hbase(main):002:0>
```

- **version**: muestra la versión de HBase.

```
hbase(main):002:0> version
1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017
```

```
hbase(main):003:0> 
```

- **help**: muestra información asociada a un comando.

```
hbase(main):003:0> help
HBase Shell, version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017
Type 'help "COMMAND"', (e.g. 'help "get"' -- the quotes are necessary) for help on a specific command.
Commands are grouped. Type 'help "COMMAND_GROUP"', (e.g. 'help "general"') for help on a command group.

COMMAND GROUPS:
  Group name: general
    Commands: status, table_help, version, whoami

  Group name: ddl
    Commands: alter, alter_async, alter_status, create, describe, disable, disable_all, drop, drop_all, enable, en

  Group name: namespace
    Commands: alter_namespace, create_namespace, describe_namespace, drop_namespace, list_namespace, list_namespac

  Group name: dml
    Commands: append, count, delete, deleteall, get, get_counter, get_splits, incr, put, scan, truncate, truncate_

  Group name: tools
    Commands: assign, balance_switch, balancer, balancer_enabled, catalogjanitor_enabled, catalogjanitor_run, cata
ve, normalize, normalizer_enabled, normalizer_switch, split, trace, unassign, wal_roll, zk_dump

  Group name: replication
    Commands: add_peer, append_peer_tableCFs, disable_peer, disable_table_replication, enable_peer, enable_table_r
tableCFs, show_peer_tableCFs

  Group name: snapshots
    Commands: clone_snapshot, delete_all_snapshot, delete_snapshot, list_snapshots, restore_snapshot, snapshot
```

6.10.2. Comandos DDL (definición de datos)

A continuación, repasaremos los comandos disponibles para la definición de las distintas estructuras de datos que podemos manejar con HBase:

- Namespace
- Tablas

Operaciones sobre Namespaces

- **create**: permite crear namespaces.

create_namespace 'master'

```
hbase(main):001:0> create_namespace 'master'
0 row(s) in 0.3150 seconds
```

- **alter_namespace:** permite añadir o eliminar una propiedad de un namespace.

```
alter_namespace 'master',
{METHOD=>'set','Propiedad1'=>'Valor1'}
```

```
hbase(main):002:0> alter_namespace 'master', {METHOD=>'set','Propiedad1'=>'Valor1'}
0 row(s) in 0.0610 seconds
```

```
alter_namespace 'master',
{METHOD=>'set','Propiedad2'=>'Valor2'}
```

```
hbase(main):003:0> alter_namespace 'master', {METHOD=>'set','Propiedad2'=>'Valor2'}
0 row(s) in 0.0740 seconds
```

```
alter_namespace 'master',
{METHOD=>'unset',NAME=>'Propiedad2'}
```

```
hbase(main):004:0> alter_namespace 'master', {METHOD=>'unset',NAME=>'Propiedad2'}
0 row(s) in 0.0650 seconds
```

- **describe_namespace:** muestra la información del namespace.

```
describe_namespace 'master'
```

```
hbase(main):005:0> describe_namespace 'master'
DESCRIPTION
{NAME => 'master', Propiedad1 => 'Valor1'}
1 row(s) in 0.0170 seconds
```

- **list_namespace:** permite mostrar la información de los namespaces existentes.

```
list_namespace
```

```
hbase(main):006:0> list_namespace
NAMESPACE
default
hbase
master
3 row(s) in 0.0510 seconds
```

create

Crea una nueva tabla en HBase.

- Ejemplo 1: namespace = master

```
create 'master:tabla_ejemplo', {NAME => 'column_family_1', VERSIONS => 1, TTL => 2592000, BLOCKCACHE => true}
```

```
hbase(main):001:0> create 'master:tabla_ejemplo', {NAME => 'column_family_1', VERSIONS => 1, TTL => 2592000, BLOCKCACHE => true}
0 row(s) in 1.5710 seconds
=> Hbase::Table - master:tabla_ejemplo
hbase(main):002:0>
```

- Ejemplo 2: namespace = default

```
create 'tabla_ejemplo_en_default_namespace', {NAME => 'column_family_1'}, {NAME => 'column_family_2'}, {NAME => 'column_family_3'}
```

```
hbase(main):002:0> create 'tabla_ejemplo_en_default_namespace', {NAME => 'column_family_1'}, {NAME => 'column_family_2'}, {NAME => 'column_family_3'}
0 row(s) in 1.2450 seconds
=> Hbase::Table - tabla_ejemplo_en_default_namespace
hbase(main):003:0> ■
```

- create 'tabla1', 'column_family_1', 'column_family_2', 'column_family_3'

```
hbase(main):003:0> create 'tabla1', 'column_family_1', 'column_family_2', 'column_family_3'
0 row(s) in 1.2450 seconds
=> Hbase::Table - tabla1
hbase(main):004:0>
```

alter

Permite modificar la información de una tabla así como atributos como MAX_FILESIZEx, MEMSTORE_FLUSHSIZE, READONLY, y DEFERRED_LOG_FLUSH.

- Ejemplo 1: cambiar o añadir la columna Family Column_family_1 en la tabla tabla1 manteniendo un máximo de cinco versiones de celdas.
alter 'tabla1', NAME => 'column_family_1', VERSIONS => 5

```
hbase(main):004:0> alter 'tabla1', NAME => 'column_family_1', VERSIONS => 5
Updating all regions with the new schema...
1/1 regions updated.
```

- Ejemplo 2: cambiar el tamaño máximo de la Column Family a 128 MB.
alter 'tabla1', METHOD => 'table_att', MAX_FILESIZEx => '134217728'

```
hbase(main):005:0> alter 'tabla1', METHOD => 'table_att', MAX_FILESIZEx => '134217728'
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 1.9910 seconds
```

- Ejemplo 3: cambiar varios parámetros de una vez.
alter 'tabla1', {NAME => 'column_family_1'}, {NAME => 'column_family_2', METHOD => 'delete'}

```
hbase(main):006:0> alter 'tabla1', {NAME => 'column_family_1'}, {NAME => 'column_family_2', METHOD => 'delete'}
Updating all regions with the new schema...
1/1 regions updated.
Done.
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 3.9680 seconds
```

delete

Permite borrar elementos de una tabla.

- Ejemplo 1: borrar la Column Family 1 de la tabla 1.

```
alter 'tabla1', NAME => 'column_family_1', METHOD => 'delete'
```

```
hbase(main):007:0> alter 'tabla1', NAME => 'column_family_1', METHOD => 'delete'
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 1.9310 seconds
```

describe

Muestra la descripción de la tabla.

Ejemplo: *describe 'tabla1'*

```
hbase(main):008:0> describe 'tabla1'
Table tabla1 is ENABLED

tabla1, {TABLE_ATTRIBUTES => {MAX_FILESIZE => '134217728'}

COLUMN FAMILIES DESCRIPTION

{NAME => 'column_family_3', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSION => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
1 row(s) in 0.0970 seconds
```

disable

Deshabilita una tabla. Por ejemplo, es necesario deshabilitarla para poder borrarla.

Ejemplo: *disable 'tabla1'*

```
hbase(main):028:0> disable 'tabla1'
0 row(s) in 1.3480 seconds
```

drop

Elimina una tabla. Antes de eliminarla hay que deshabilitarla. Si la tabla tiene más de una región, hay que ejecutar antes el comando de compactación en .META.

Ejemplo: *drop 'tabla1'*

```
hbase(main):029:0> drop 'tabla1'
0 row(s) in 0.2560 seconds
```

enable

Habilita una tabla.

Ejemplo:

```
create 't1', 'col1', 'col2', 'col3'
```

```
disable 't1'
```

```
enable 't1'
```

```
hbase(main):039:0> create 't1', 'col1', 'col2', 'col3'  
0 row(s) in 0.2240 seconds  
  
=> Hbase::Table - t1  
hbase(main):040:0> disable 't1'  
0 row(s) in 1.4420 seconds  
  
hbase(main):041:0> enable 't1'  
0 row(s) in 0.2870 seconds
```

exists

Comprueba si una tabla existe.

Ejemplo: exists 't1'

```
hbase(main):041:0> exists 't1'  
Table t1 does exist  
0 row(s) in 0.0840 seconds
```

is_disabled

Comprueba si una tabla está deshabilitada.

Ejemplo: is_disabled 't1'

```
hbase(main):042:0> is_disabled 't1'  
false  
0 row(s) in 0.0610 seconds
```

is_enabled

Comprueba si una tabla está habilitada.

Ejemplo: is_enabled 't1'

```
hbase(main):043:0> is_enabled 't1'  
true  
0 row(s) in 0.0530 seconds
```

list

Lista todas las tablas de una base de datos.

Ejemplo: list

```
hbase(main):013:0> list  
TABLE  
master:tabla_ejemplo  
t1  
tabla_ejemplo_en_default_namespace  
3 row(s) in 0.0030 seconds  
=> ["master:tabla_ejemplo", "t1", "tabla_ejemplo_en_default_namespace"]
```

truncate

Deshabilita, elimina o recrea una tabla.

Ejemplo: truncate 't1'

```
hbase(main):045:0> truncate 't1'  
Truncating 't1' table (it may take a while):  
- Disabling table...  
- Truncating table...  
0 row(s) in 1.5320 seconds
```

6.10.3. Comandos de manipulación de datos CRUD

A continuación, se repasan los comandos principales para llevar a cabo tareas de creación, lectura, actualización y eliminación de información en HBase.

Comandos de manipulación de datos sobre filas y columnas

put

Inserta el valor de una celda en la tabla/fila/columna especificada y opcionalmente con las coordenadas del timestamp.

- Ejemplo: poner una celda con valor 1 en la tabla 1 en la fila 1 en la columna 1.

```
put 't1', 'r1', 'col1', '1'
```

```
hbase(main):045:0> put 't1', 'r1', 'col1', '1'  
0 row(s) in 0.0290 seconds
```

```
put 't1', 'r2','col1',1
```

get

Recupera los contenidos de una celda o fila.

- Ejemplo:

```
get 't1', 'r1'
```

```
hbase(main):046:0> get 't1', 'r1'  
COLUMN           CELL  
 col1:           timestamp=1421773810867, value=1  
1 row(s) in 0.0330 seconds
```

- Si se dispone de un conjunto de datos más extenso, se pueden emplear las siguientes consultas:

```
get 't1', 'r1', {TIMERANGE => [ts1, ts2]}
```

```
get 't1', 'r1', {COLUMN => 'col1'}
```

```
get 't1', 'r1', {COLUMN => ['col1', 'col2', 'col3']}
```

```
get 't1', 'r1', {COLUMN => 'col1', TIMESTAMP => ts1}
```

```
get 't1', 'r1', {COLUMN => 'col1', TIMERANGE => [ts1, ts2], VERSIONS => 4}
```

```
get 't1', 'r1', {COLUMN => 'col1', TIMESTAMP => ts1, VERSIONS => 4}
```

```
get 't1', 'r1', 'col1'
```

```
get 't1', 'r1', 'col1', 'col2'
```

```
get 't1', 'r1', ['col1', 'col2']
```

incr

Aumenta el valor de una celda, indicando sus coordenadas (tabla/fila/columna).

- Ejemplos:

incr 't1', 'r1', 'col1' → *incrementar en 1*

incr 't1', 'r1', 'col1', 1 → *incrementar en 1*

incr 't1', 'r1', 'col1', 10 → *incrementar en 10*

count

Devuelve el número de filas de una tabla. Esta operación tardará mucho. Por defecto, count se muestra cada mil filas por defecto, pero el intervalo puede ser indicado excepcionalmente, del mismo modo que los escaneos de caché se pueden indicar.

- Ejemplos:

count 't1'

```
hbase(main):057:0> count 't1'  
2 row(s) in 0.1160 seconds  
=> 2
```

count 't1', INTERVAL => 100000

count 't1', CACHE => 1000

count 't1', INTERVAL => 10, CACHE => 1000

getcounter

Devuelve el contador de una celda especificando las coordenadas de la tabla/fila/columna.

- Ejemplo:

get_counter 't1', 'r1', 'col1'

delete

Elimina una celda de una tabla/fila/columna.

- Ejemplo:

```
delete 't1', 'r1', 'col1'
```

```
hbase(main):061:0> delete 't1', 'r1', 'col1'  
0 row(s) in 0.0780 seconds
```

delete_all

Elimina toda una fila. Opcionalmente se puede indicar la columna y el timestamp

- Ejemplos:

```
deleteall 't1', 'r1'
```

```
hbase(main):062:0> deleteall 't1', 'r1'  
0 row(s) in 0.0350 seconds
```

```
deleteall 't1', 'r1', 'c1'
```

```
deleteall 't1', 'r1', 'c1', ts1
```

6.10.4. Comandos avanzados

Adicionalmente, podéis ver otros comandos avanzados de HBase. Algunos de ellos se mostrarán en los siguientes ejemplos.

Herramientas de HBase

assign

Asigna una región. Usar con cuidado y solo por expertos. Si la región existe, este comando hará una reasignación.

- Ejemplo: *hbase> assign 'REGION_NAME'*

balancer

Ejecuta el balanceador del clúster. Devuelve *true* si el balanceador se ejecuta y fue capaz de indicar a las regiones del servidor que desasignaran las regiones para balancear. En caso contrario devuelve *false*.

- Ejemplo: *hbase> balancer*

balance_switch

Activa/inactiva el balanceador.

- Ejemplos:

```
hbase>balance_switch true
hbase>balance_switch false
```

close_region

Cierra una región. Solicita al servidor maestro el cierre de una región 'REGIONNAME' en el clúster o servidor indicado, mediante 'SERVER_NAME'. Al cerrar una región se espera que la región sea un nombre de región correcto del estilo

TestTable,0094429456,1289497600452.527db22f95c8a9e0116f0cc13c680396. El nombre de la región

está codificado en *hash* al final del nombre de la región, es decir, 527db22f95c8a9e0116f0cc13c680396.

El nombre del servidor se indica con el host, su puerto y su código de inicio. Por ejemplo:

host187.example.com,60020,1289493121758. Si se cierra una región indicando el servidor, el nodo maestro desconocerá su cierre. Una vez cerrada una región, permanecerá cerrada y habrá que emplear el comando assign para volver a activarla.

- Ejemplos:

```
hbase>close_region 'REGIONNAME'
hbase> close_region 'REGIONNAME', 'SERVER_NAME'
```

compact

Compacta todas las regiones de la tabla indicada o de la región indicada. También puede compactar una única Column Family en una región.

- Ejemplos:

- Compactar todas las regiones de una tabla:`hbase> compact 't1'`
- Compactar toda una región:`hbase> compact 'r1'`
- Compactar una Column Family dentro de una región:`hbase> compact 'r1', 'c1'`
- Compactar una Column Family dentro de una tabla:`hbase> compact 't1', 'c1'`

flush

Liberá la información de las regiones de la tabla o de la región indicada.

- Ejemplos:

```
hbase> flush 'TABLENAME'
hbase> flush 'REGIONNAME'
```

hlog_roll

Reinicia la escritura del log en un nuevo fichero. Se debe indicar el nombre del servidor de regiones como parámetro.

- Ejemplos

```
hbase> hlog_roll host187.example.com,60020,1289493121758
```

major_compact

Ejecuta una compactación completa de la tabla indicada o de la región. Para compactar una única Column Family dentro de una región se debe especificar el nombre de la región seguido del Column Family.

- Ejemplos:

- *Compactar todas las regiones de una tabla: hbase> major_compact 't1'*
- *Compactar una Column Family dentro de una región: hbase> major_compact 'r1', 'c1'*
- *Compactar una Column Family dentro de una tabla: hbase> major_compact 't1', 'c1'*

move

Permite mover regiones. Opcionalmente se puede indicar el objetivo al que se quiere mover. En caso contrario, se moverá aleatoriamente. Se debe indicar el nombre de la región codificado para evitar solapamientos.

- Ejemplos:

```
hbase> move 'ENCODED_REGIONNAME'  
hbase> move 'ENCODED_REGIONNAME', 'SERVER_NAME'
```

scan

Realiza un escaneo de toda la tabla. Opcionalmente se puede indicar el diccionario con los parámetros que se deben escanear, como por ejemplo: LIMIT, STARTROW, STOPROW, TIMESTAMP, COLUMNS. Si no se especifican las columnas, se escaneará toda la tabla.

- Ejemplos:

```
hbase> scan '.META'  
hbase> scan '.META.', {COLUMNS => 'info:regioninfo'}  
hbase> scan 't1', {COLUMNS => ['c1','c2'], LIMIT => 10, \STARTROW => 'xyz'}
```

split

Permite partir la tabla o la región indicada en una única región. Con el segundo parámetro se puede indicar la clave por la que se partirá la región.

- Ejemplos:

```
hbase> split 'TABLENAME'  
hbase> split 'REGIONNAME'
```

unassign

Desasigna una región, cerrándola de su ubicación actual y reabriéndola después. Se puede indicar mediante el comando *true* la reasignación forzada, eliminando toda la información en el nodo maestro.

- Ejemplos:

```
hbase> unassign 'REGIONNAME'  
hbase> unassign 'REGIONNAME', true
```

zk_dump

Vacia el estado del clúster HBase empleando un comando similar al Zookeeper.

- Ejemplos:
hbase> zk_dump

Herramientas de replicación del clúster

- **add_peer**: añade una copia del clúster para replicarlo. Se debe indicar el ID y la clave del clúster para componerlo.
 - Ejemplos:
hbase> add_peer '1', "server1.cie.com:2181:/hbase"
hbase> add_peer '2', "zk1,zk2,zk3:2182:/hbase-prod"
- **remove_peer**: detiene la replicación y elimina la metainformación asociada
 - Ejemplo:
hbase> remove_peer '1'
- **list_peers**: muestra todas las replicaciones de clústers.
 - Ejemplo:
hbase> list_peers
- **enable_peer**: reinicia la replicación de un clúster que había sido deshabilitada.
 - Ejemplo:
hbase> enable_peer '1'
- **disable_peer**: para la replicación de un clúster, pero lo mantiene para terminar su replicación.
 - Ejemplo:
hbase> disable_peer '1'
- **start_replication**: reinicia todas las replicaciones. Solo debe usarse en situaciones de carga críticas.
 - Ejemplo:
hbase> start_replication
- **stop_replication**: detiene todas las replicaciones. Solo debe usarse en situaciones críticas.
 - Ejemplo:
hbase> stop_replication

Herramientas de seguridad

- **grant**: permite dar permisos específicos a usuarios (RWXCA, es decir, read, write, exec, create, admin).
 - Ejemplos:
hbase> grant 'johnsmith', 'RWXCA' hbase
hbase> grant 'bobsmith', 'RW', 't1', 'f1', 'col1'
- **revoke**: elimina los permisos de acceso de un usuario.
 - Ejemplo:
hbase> revoke 'bobsmith', 't1', 'f1', 'col1'
- **user_permission**: muestra todos los permisos de un usuario.
 - Ejemplos:
hbase> user_permission
hbase> user_permission 'table1'

6.11. Ejemplo

A continuación, se describe un ejemplo de uso de HBase. Los objetivos del ejemplo son los siguientes:

1. Arrancar HBase y acceder a la consola de HBase.

```
cd /home/bigdata/hbase
bin/start-hbase.sh
jps
bin/hbase shell
```

```
bigdata@bigdata:~/hbase$ hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017
hbase(main):001:0>
```

2. Comprobar el estado de HBase.

```
status
hbase(main):001:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 5.0000 average load
hbase(main):002:0>
```

3. Crear un namespace llamado ejercicios.

```
create_namespace 'ejercicios'
```

```
hbase(main):002:0> create_namespace 'ejercicios'  
0 row(s) in 0.0470 seconds
```

4. Crear una tabla llamada predicciones con las siguientes columnas: localidad, temperatura y cuota de nieve.

```
create 'ejercicios:prediccion', 'localidad', 'temperatura', 'cuota_nieve'
```

```
hbase(main):003:0> create 'ejercicios:prediccion', 'localidad', 'temperatura', '  
cuota_nieve'  
0 row(s) in 0.2800 seconds  
  
=> Hbase::Table - ejercicios:prediccion  
Hbase::Table{  
  name: ejercicios:prediccion  
  columns: [localidad, temperatura, cuota_nieve]  
}
```

5. Hacer un describe de la tabla.

```
describe 'ejercicios:prediccion'
```

```
hbase(main):004:0> describe 'ejercicios:prediccion'  
Table ejercicios:prediccion is ENABLED  
COLUMN FAMILIES DESCRIPTION  
{NAME => 'cuota_nieve' DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATIO  
N_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN VERSIONS => '0', TTL =>  
'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}  
{NAME => 'localidad' DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATIO  
N_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN VERSIONS => '0', TTL =>  
'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}  
{NAME => 'temperatura' DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATIO  
N_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN VERSIONS => '0', TTL =>  
'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}  
3 row(s) in 0.0900 seconds
```

6. Acceder a la página de la AEMET para obtener las predicciones de Madrid.

<http://www.aemet.es/es/eltiempo/prediccion/municipios/madrid-id28079>

Predicción por Municipios. Madrid (Madrid)

Predicción 7 días		Predicción por horas			El tiempo en su web												
		Tabla		Gráfica													
Capital: Madrid (altitud: 657 m)																	
Latitud: 40° 24' 30" N - Longitud: 3° 41' 15" O - Posición: Ver localización ►																	
Zona de avisos: Metropolitana y Henares																	
Ver tabla detallada ▾		Descargar XML de la predicción detallada de Madrid															
Fecha	mié 21		jue 22			vie 23		sáb 24		dom 25	lun 26	mar 27					
	12-18	18-24	0-6	6-12	12-18	18-24	0-12	12-24	0-12	12-24							
Estado del cielo																	
Prob. precip.	100%	0%	0%	5%	0%	0%	0%	0%	0%	5%	0%	5%					
Cota nieve prov.(m)	800																
Temp. mín./máx. (°C)	-2 / 6		-2 / 9			-3 / 7		-2 / 10		-3 / 8	-2 / 8	-2 / 10					
Viento (km/h)								-	-	-	-	-					
Indice UV máximo																	
Avisos Metropolitana y Henares																	
	Sin Riesgo ►		Sin Riesgo ►			Sin Riesgo ►											

7. Cargar los datos de las predicciones de la AEMET de Madrid

Empleando como clave la fecha del día y el ID de la estación de la AEMET (28079). Se debe tener en cuenta que habrá que insertar varios valores por cada registro y que en la columna de temperatura deberemos almacenar por separado la temperatura máxima y mínima.

```
put 'ejercicios:prediccion', '20150121#28079', 'temperatura:max', 6
```

```
put 'ejercicios:prediccion', '20150121#28079', 'temperatura:min', -2
```

```
put 'ejercicios:prediccion', '20150121#28079', 'cuota_nieve',800
```

`put 'ejercicios:prediccion', '20150122#28079', 'temperatura:max', 9`

```
put 'ejercicios:prediccion', '20150122#28079', 'temperatura:min', -2
```

`put 'ejercicios:prediccion', '20150123#28079', 'temperatura:max', 7`

```
put 'ejercicios:prediccion', '20150123#28079', 'temperatura:min', -3
```

put 'ejercicios:prediccion', '20150124#28079', 'temperatura:max', 10

`put 'ejercicios:prediccion', '20150124#28079', 'temperatura:min', -2`

```

put 'ejercicios:prediccion', '20150125#28079', 'temperatura:max', 8
put 'ejercicios:prediccion', '20150125#28079', 'temperatura:min', -2
put 'ejercicios:prediccion', '20150126#28079', 'temperatura:max', 10
put 'ejercicios:prediccion', '20150126#28079', 'temperatura:min', -2

hbase(main):022:0>
hbase(main):023:0* put 'ejercicios:prediccion', '20150125#28079', 'temperatura:max', 8
0 row(s) in 0.0040 seconds

hbase(main):024:0>
hbase(main):025:0* put 'ejercicios:prediccion', '20150125#28079', 'temperatura:min', -2
0 row(s) in 0.0070 seconds

hbase(main):026:0>
hbase(main):027:0* put 'ejercicios:prediccion', '20150126#28079', 'temperatura:max', 10
0 row(s) in 0.0080 seconds

hbase(main):028:0>
hbase(main):029:0* put 'ejercicios:prediccion', '20150126#28079', 'temperatura:min', -2
0 row(s) in 0.0160 seconds

hbase(main):030:0>
hbase(main):031:0* 

```

8. Escanear la información de la tabla.

```
scan 'ejercicios:prediccion'
```

```

hbase(main):033:0> scan 'ejercicios:prediccion'
ROW                                COLUMN+CELL
 20150121#28079      column=cuota_nieve:, timestamp=1421842504491, value=800
 20150121#28079      column=temperatura:max, timestamp=1421842504265, value=6
 20150121#28079      column=temperatura:min, timestamp=1421842504405, value=-2
 20150122#28079      column=temperatura:max, timestamp=1421842504571, value=9
 20150122#28079      column=temperatura:min, timestamp=1421842504706, value=-2
 20150123#28079      column=temperatura:max, timestamp=1421842504790, value=7
 20150123#28079      column=temperatura:min, timestamp=1421842504854, value=-3
 20150124#28079      column=temperatura:max, timestamp=1421842504927, value=10
 20150124#28079      column=temperatura:min, timestamp=1421842504990, value=-2
 20150125#28079      column=temperatura:max, timestamp=1421842505119, value=8
 20150125#28079      column=temperatura:min, timestamp=1421842505193, value=-2
 20150126#28079      column=temperatura:max, timestamp=1421842505252, value=10
 20150126#28079      column=temperatura:min, timestamp=1421842505315, value=-2
6 row(s) in 0.1640 seconds

```

9. Consultar la información del día 21/01/2015 del municipio 28079.

```
get 'ejercicios:prediccion','20150121#28079'
```

```
hbase(main):034:0> get 'ejercicios:prediccion','20150121#28079'  
COLUMN CELL  
cuota_nieve: timestamp=1421842504491, value=800  
temperatura:max timestamp=1421842504265, value=6  
temperatura:min timestamp=1421842504405, value=-2  
3 row(s) in 0.0630 seconds
```

10. Consultar la información de la temperatura máxima del día 21/01/2015 del municipio 28079.

```
get 'ejercicios:prediccion','20150121#28079', 'temperatura:max'
```

```
hbase(main):035:0> get 'ejercicios:prediccion','20150121#28079', 'temperatura:max'  
COLUMN CELL  
temperatura:max timestamp=1421842504265, value=6  
1 row(s) in 0.0400 seconds
```

11. Consultar la información de la cuota de nieve del día 21/01/2015 del municipio 28079.

```
get 'ejercicios:prediccion','20150121#28079', 'cuota_nieve'
```

```
hbase(main):036:0> get 'ejercicios:prediccion','20150121#28079', 'cuota_nieve'  
COLUMN CELL  
cuota_nieve: timestamp=1421842504491, value=800  
1 row(s) in 0.0380 seconds
```

12. Eliminar la temperatura máxima del registro del día 23/01/2015 del municipio 28079.

```
delete 'ejercicios:prediccion','20150123#28079', 'temperatura:max'
```

```
hbase(main):037:0> delete 'ejercicios:prediccion','20150123#28079', 'temperatura:max'  
0 row(s) in 0.0320 seconds
```

13. Consultar el registro del día 23/01/2015 del municipio 28079.

```
get 'ejercicios:prediccion','20150123#28079'
```

```
hbase(main):038:0> get 'ejercicios:prediccion','20150123#28079'  
COLUMN CELL  
temperatura:min timestamp=1421842504854, value=-3  
1 row(s) in 0.0170 seconds
```

14. Emplear los comandos de ayuda.

`help 'scan'`

```
hbase(main):034:0> help 'scan'
Scan a table; pass table name and optionally a dictionary of scanner
specifications. Scanner specifications may include one or more of:
TIMERANGE, FILTER, LIMIT, STARTROW, STOPROW, ROWPREFIXFILTER, TIMESTAMP,
MAXLENGTH or COLUMNS, CACHE or RAW, VERSIONS
```

If no columns are specified, all columns will be scanned.
To scan all members of a column family, leave the qualifier empty as in
'col_family:'.

The filter can be specified in two ways:

1. Using a filterString - more information on this is available in the Filter Language document attached to the HBASE-4176 JIRA
2. Using the entire package name of the filter.

Some examples:

```
hbase> scan 'hbase:meta'
hbase> scan 'hbase:meta', {COLUMNS => 'info:regioninfo'}
hbase> scan 'ns1:t1', {COLUMNS => ['c1', 'c2'], LIMIT => 10, STARTROW => 'xyz'}
hbase> scan 't1', {COLUMNS => ['c1', 'c2'], LIMIT => 10, STARTROW => 'xyz'}
hbase> scan 't1', {COLUMNS => 'c1', TIMERANGE => [1303668804, 1303668904]}
hbase> scan 't1', {REVERSED => true}
hbase> scan 't1', {ROWPREFIXFILTER => 'row2', FILTER =>
  (QualifierFilter (>=, 'binary:xyz')) AND (TimestampsFilter ( 123, 456))"}
hbase> scan 't1', {FILTER =>
  org.apache.hadoop.hbase.filter.ColumnPaginationFilter.new(1, 0)}
hbase> scan 't1', {CONSISTENCY => 'TIMELINE'}
```

15. Emplear **limit** para obtener algunos valores.

`scan 'ejercicios:prediccion', { COLUMNS => 'temperatura', LIMIT=> 2 }`

```
hbase(main):035:0> scan 'ejercicios:prediccion', { COLUMNS => 'temperatura', LIMIT=> 2 }
ROW                                     COLUMN+CELL
20151213#28079                         column=temperatura:max, timestamp=1454244216021, value=13
20151213#28079                         column=temperatura:min, timestamp=1454244216146, value=5
20151214#28079                         column=temperatura:max, timestamp=1454244216249, value=14
20151214#28079                         column=temperatura:min, timestamp=1454244216313, value=7
2 row(s) in 0.0600 seconds
```

16. Emplear rangos de fechas.

```
scan 'ejercicios:prediccion', { COLUMNS => 'temperatura', TIMERANGE => [1504459258192, 1504459258544] }
```

```
hbase(main):038:0> scan 'ejercicios:prediccion', { COLUMNS => 'temperatura', TIMERANGE => [1504459258192, 1504459258544] }
ROW                                COLUMN+CELL
20150122#28079                      column=temperatura:max, timestamp=1504459258192, value=9
20150122#28079                      column=temperatura:min, timestamp=1504459258299, value=-2
20150123#28079                      column=temperatura:min, timestamp=1504459258442, value=-3
2 row(s) in 0.0700 seconds
```

17. Filtrado por columnas para obtener valores de temperatura mayores a 12.

```
scan 'ejercicios:prediccion', { COLUMNS => 'temperatura:max', FILTER => "ValueFilter( >, 'binary:12' )"} 
```

```
hbase(main):039:0> scan 'ejercicios:prediccion', { COLUMNS => 'temperatura:max', FILTER => "ValueFilter( >, 'binary:12' )"}
ROW                                COLUMN+CELL
20150121#28079                      column=temperatura:max, timestamp=1504459257742, value=6
20150122#28079                      column=temperatura:max, timestamp=1504459258192, value=9
20150125#28079                      column=temperatura:max, timestamp=1504459258704, value=8
3 row(s) in 0.0670 seconds
```

18. Eliminar un registro completo del día 26.

```
deleteall 'ejercicios:prediccion','20150126#28079'
```

```
hbase(main):039:0> deleteall 'ejercicios:prediccion','20150126#28079'
0 row(s) in 0.0110 seconds
```

19. Deshabilitar tabla de predicción.

```
disable 'ejercicios:prediccion'
```

```
hbase(main):040:0> disable 'ejercicios:prediccion'
0 row(s) in 1.3660 seconds
```

20. Borrar tabla de predicciones.

```
drop 'ejercicios:prediccion'
```

```
hbase(main):041:0> drop 'ejercicios:prediccion'
0 row(s) in 0.2500 seconds
```

21. Listar tablas.*list*

```
hbase(main):043:0> list
TABLE
master:tabla_ejemplo
t1
tabla_ejemplo_en_default_namespace
3 row(s) in 0.0360 seconds

=> ["master:tabla_ejemplo", "t1", "tabla_ejemplo_en_default_namespace"]
```

6.12. Ejemplo shell

En este ejemplo, veremos cómo podemos crear distintos *scripts* para crear y cargar información en HBase, así como realizar determinadas consultas.

1. Crear una carpeta en /home/bigdata llamada ejemplos.

```
mkdir /home/bigdata/ejemplosHBase
```

```
bigdata@bigdata:~/hbase$ mkdir /home/bigdata/ejemplosHBase
```

2. Creamos un fichero con la información.

```
sudo nano /home/bigdata/ejemplosHBase/gestionar_alumnos.sh
```

```
bigdata@bigdata:~/hbase$ sudo nano /home/bigdata/ejemplosHBase/gestionar_alumnos.sh
```

```
#!/bin/sh
```

```
# Creamos la tabla
```

```
create 'alumnos', 'datos_personales'
```

```
# Consultamos la información de la tabla
```

```
describe 'alumnos'
```

```
# Añadimos un registro llamado alumno con distinta información en una única Column Family con varios Column Qualifiers
```

```
put 'alumnos', 'alumno1', 'datos_personales:nombre', 'Juan'
```

```
put 'alumnos', 'alumno1', 'datos_personales:apellido1', 'Pérez'
```

```

put 'alumnos', 'alumno1', 'datos_personales:apellido2', 'Garcia'

put 'alumnos', 'alumno1', 'datos_personales:email', 'jperez@masterbigdata.es'

# Comprobamos que la tabla tiene datos

scan 'alumnos'

# Recuperamos un elemento

get 'alumnos', 'alumno1'

# Desabilitamos la tabla

disable 'alumnos'

# Borramos la tabla.

drop 'alumnos'

# Salimos de la shell

exit

```

GNU nano 2.2.6 Archivo: /home/bigdata/ejemplosHBase/gestionar_alumnos.sh

```

#!/bin/sh
# Creamos la tabla
create 'alumnos', 'datos_personales'
# Consultamos la información de la tabla
describe 'alumnos'
# ^~demos un registro llamado alumno con distinta información en una única column family con va
Archivos, 'alumno1', 'datos_personales:nombre', 'Juan'
put 'alumnos', 'alumno1', 'datos_personales:apellido1', 'Pérez'
put 'alumnos', 'alumno1', 'datos_personales:apellido2', 'Garcia'
put 'alumnos', 'alumno1', 'datos_personales:email', 'jperez@masterbigdata.es'
# Comprobamos que la tabla tiene datos
scan 'alumnos'
# Recuperamos un elemento
get 'alumnos', 'alumno1'
# Desabilitamos la tabla
disable 'alumnos'
# Borramos la tabla.
drop 'alumnos'
# Salimos de la shell
exit

```

3. Comprobamos si está arrancado HBase y, si no, lo arrancamos.

```
jps
```

```
bigdata@bigdata:~/hbase$ jps
31200 NodeManager
30657 DataNode
3025 Jps
31058 ResourceManager
30904 SecondaryNameNode
32396 HMaster
30511 NameNode
```

4. Ejecutamos el script.

```
./bin/hbase shell /home/bigdata/ejemplosHBase/gestionar_alumnos.sh
```

```
#!/bin/sh
# Creamos la tabla
create 'alumnos', 'datos_personales'
# Consultamos la información de la tabla
describe 'alumnos'
```

```
bigdata@bigdata:~/hbase$ ./bin/hbase shell /home/bigdata/ejemplosHBase/gestionar_alumnos.sh
0 row(s) in 1.4920 seconds

Table alumnos is ENABLED
alumnos
COLUMN FAMILIES DESCRIPTION
{NAME => 'datos_personales', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_C
ELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS =>
'0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
1 row(s) in 0.2160 seconds
```

```
# Añadimos un registro llamado alumno con distinta información en una única column family con varios
put 'alumnos', 'alumno1', 'datos_personales:nombre', 'Juan'
put 'alumnos', 'alumno1', 'datos_personales:apellido1', 'Pérez'
put 'alumnos', 'alumno1', 'datos_personales:apellido2', 'García'
put 'alumnos', 'alumno1', 'datos_personales:email', 'jperez@masterbigdata.es'
```

```
0 row(s) in 0.2530 seconds
0 row(s) in 0.0070 seconds
0 row(s) in 0.0140 seconds
0 row(s) in 0.0280 seconds
```

```
# Comprobamos que la tabla tiene datos.
scan 'alumnos'
```

```
ROW                               COLUMN+CELL
alumno1                           column=datos_personales:apellido1, timestamp=1471255656523, value=P\xC3\xA9rez
alumno1                           column=datos_personales:apellido2, timestamp=1471255656539, value=Garcia
alumno1                           column=datos_personales:email, timestamp=1471255656571, value=jperez@masterbigdata.es
alumno1                           column=datos_personales:nombre, timestamp=1471255656502, value=Juan
1 row(s) in 0.1290 seconds
```

```
# Recuperamos un elemento
get 'alumnos', 'alumno1'
```

COLUMN	CELL
datos_personales:apellido1	timestamp=1471255656523, value=P\xC3\xA9rez
datos_personales:apellido2	timestamp=1471255656539, value=Garcia
datos_personales:email	timestamp=1471255656571, value=jperez@masterbigdata.es
datos_personales:nombre	timestamp=1471255656502, value=Juan

```
4 row(s) in 0.1200 seconds
```

```
# Deshabilitamos la tabla
disable 'alumnos'
# Borramos la tabla.
drop 'alumnos'
# Salimos de la shell
exit
```

```
0 row(s) in 2.3320 seconds
0 row(s) in 1.2850 seconds
bigdata@bigdata:~/hbase$
```

5. Volvemos a crear la tabla.

```
echo "create 'alumnos', 'info'" | ./bin/hbase shell
```

```
bigdata@bigdata:~/hbase$ echo "create 'alumnos', 'info'" | ./bin/hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

create 'alumnos', 'info'
0 row(s) in 1.5700 seconds

Hbase::Table - alumnos
```

6. Accedemos a la shell para llenar de datos la tabla.

```
hbase shell
```

```
bigdata@bigdata:~/hbase$ hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

hbase(main):001:0> █
```

7. Empleamos JRuby para hacer una carga de datos.

```
hbase(main):001:0> for i in '1'..'20' do |
```

```
    hbase(main):002:1* put "alumnos", "alumno_#{i}", "info:email", "alumno_#{i}@masterbigdata.com" |
```

```
hbase(main):003:1* end
```

```
hbase(main):001:0> for i in '1'..'20' do \
hbase(main):002:1* put "alumnos", "alumno_#{i}", "info:email", "alumno_#{i}@masterbigdata.com" \
hbase(main):003:1* end
0 row(s) in 0.7130 seconds

0 row(s) in 0.0080 seconds
0 row(s) in 0.0160 seconds
0 row(s) in 0.0280 seconds
```

```
0 row(s) in 0.0130 seconds
```

```
=> "1".."20"
```

```
hbase(main):004:0> █
```

8. Comprobamos que los datos se han insertado correctamente.

```
hbase(main):004:0> scan 'alumnos'
```

```
hbase(main):004:0> scan 'alumnos'
ROW                                         COLUMN+CELL
alumno_1                                     column=info:email, timestamp=1471256222105, value=alumno_1@masterbigdata.com
alumno_10                                    column=info:email, timestamp=1471256222278, value=alumno_10@masterbigdata.com
alumno_11                                    column=info:email, timestamp=1471256222294, value=alumno_11@masterbigdata.com
alumno_12                                    column=info:email, timestamp=1471256222310, value=alumno_12@masterbigdata.com
alumno_13                                    column=info:email, timestamp=1471256222326, value=alumno_13@masterbigdata.com
alumno_14                                    column=info:email, timestamp=1471256222350, value=alumno_14@masterbigdata.com
alumno_15                                    column=info:email, timestamp=1471256222360, value=alumno_15@masterbigdata.com
alumno_16                                    column=info:email, timestamp=1471256222407, value=alumno_16@masterbigdata.com
alumno_17                                    column=info:email, timestamp=1471256222431, value=alumno_17@masterbigdata.com
alumno_18                                    column=info:email, timestamp=1471256222452, value=alumno_18@masterbigdata.com
alumno_19                                    column=info:email, timestamp=1471256222492, value=alumno_19@masterbigdata.com
alumno_2                                     column=info:email, timestamp=1471256222132, value=alumno_2@masterbigdata.com
alumno_20                                    column=info:email, timestamp=1471256222507, value=alumno_20@masterbigdata.com
alumno_3                                     column=info:email, timestamp=1471256222143, value=alumno_3@masterbigdata.com
alumno_4                                     column=info:email, timestamp=1471256222178, value=alumno_4@masterbigdata.com
alumno_5                                     column=info:email, timestamp=1471256222190, value=alumno_5@masterbigdata.com
alumno_6                                     column=info:email, timestamp=1471256222221, value=alumno_6@masterbigdata.com
alumno_7                                     column=info:email, timestamp=1471256222238, value=alumno_7@masterbigdata.com
alumno_8                                     column=info:email, timestamp=1471256222247, value=alumno_8@masterbigdata.com
alumno_9                                     column=info:email, timestamp=1471256222260, value=alumno_9@masterbigdata.com
20 row(s) in 0.1610 seconds
```

9. Obtenemos un alumno.

```
hbase(main):005:0> get 'alumnos', 'alumno_1'
```

```
hbase(main):005:0> get 'alumnos', 'alumno_1'
COLUMN           CELL
info:email      timestamp=1471256222105, value=alumno_1@masterbigdata.com
1 row(s) in 0.1290 seconds
```

10. Obtenemos la fecha de inserción de su dato.

```
hbase(main):006:0> Time.at(1471256222105/1000)
```

```
hbase(main):006:0> Time.at(1471256222105/1000)
=> Mon Aug 15 12:17:02 +0200 2016
```

11. Abandonamos la shell.

```
hbase(main):007:0> exit
```

```
hbase(main):007:0> exit
bigdata@bigdata:~/hbase$
```

12. Creamos un fichero con la información.

```
sudo nano /home/bigdata/ejemplosHBase/scan_alumnos.sh
```

```
bigdata@bigdata:~/hbase$ sudo nano /home/bigdata/ejemplosHBase/scan_alumnos.sh
```

```
scan 'alumnos', {STARTROW => 'alumno_3', STOPROW => 'alumno_8'}
```

```
exit
```

```
GNU nano 2.2.6                               Archivo: /home/bigdata/ejemplosHBase/scan_alumnos.sh
scan 'alumnos', [STARTROW => 'alumno_3', STOPROW => 'alumno_8']
exit
```

13. Ejecutamos el script.

```
hbase shell /home/bigdata/ejemplosHBase/scan_alumnos.sh
```

```
bigdata@bigdata:~/hbase$ hbase shell /home/bigdata/ejemplosHBase/scan_alumnos.sh
ROW                                COLUMN+CELL
alumno_3                            column=info:email, timestamp=1504460323334, value=alumno_3@masterbigdata.com
alumno_4                            column=info:email, timestamp=1504460323361, value=alumno_4@masterbigdata.com
alumno_5                            column=info:email, timestamp=1504460323374, value=alumno_5@masterbigdata.com
alumno_6                            column=info:email, timestamp=1504460323385, value=alumno_6@masterbigdata.com
alumno_7                            column=info:email, timestamp=1504460323410, value=alumno_7@masterbigdata.com
5 row(s) in 0.5440 seconds
```

14. Mejoramos el fichero de consulta.

```
sudo nano /home/bigdata/ejemplosHBase/scan_alumnos2.sh

bigdata@bigdata:~/hbase$ sudo nano /home/bigdata/ejemplosHBase/scan_alumnos2.sh

#!/bin/bash

TABLE=$1

STARTROW=$2

STOPROW=$3

exec hbase shell <<EOF

scan "${TABLE}", {STARTROW => "${STARTROW}", STOPROW => "${STOPROW}"}

EOF
```

```
GNU nano 2.2.6                               Archivo: /home/bigdata/ejemplosHBase/scan_alumnos2.sh

#!/bin/bash
TABLE=$1
STARTROW=$2
STOPROW=$3
exec hbase shell <<EOF
    scan "${TABLE}", {STARTROW => "${STARTROW}", STOPROW => "${STOPROW}"}
EOF
```

15. Ejecutamos el script.

```
cd /home/bigdata/ejemplosHBase

sudo chmod 755 scan_alumnos2.sh

ls -lrt scan_alumnos2.sh

bigdata@bigdata:~/hbase$ cd /home/bigdata/ejemplosHBase
bigdata@bigdata:~/ejemplosHBase$ sudo chmod 755 scan_alumnos2.sh
bigdata@bigdata:~/ejemplosHBase$ ls -lrt scan_alumnos2.sh
-rwxr-xr-x 1 root root 147 sep 3 19:41 scan_alumnos2.sh
```

16. Ejecutamos el script.

```
./scan_alumnos2.sh alumnos alumno_3 alumno_8
```

```
bigdata@bigdata:~/ejemplosHBase$ ./scan_alumnos2.sh alumnos alumno_3 alumno_8
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

      scan "alumnos", {STARTROW => "alumno_3", STOPROW => "alumno_8"}
ROW                                     COLUMN+CELL
alumno_3                               column=info:email, timestamp=1504460323334, value=alumno_3@masterbigdata.com
alumno_4                               column=info:email, timestamp=1504460323361, value=alumno_4@masterbigdata.com
alumno_5                               column=info:email, timestamp=1504460323374, value=alumno_5@masterbigdata.com
alumno_6                               column=info:email, timestamp=1504460323385, value=alumno_6@masterbigdata.com
alumno_7                               column=info:email, timestamp=1504460323410, value=alumno_7@masterbigdata.com
5 row(s) in 0.6120 seconds
```

17. Repetimos sin indicar fecha final.

```
./scan_alumnos2.sh alumnos alumno_3
```

```
bigdata@bigdata:~/ejemplosHBase$ ./scan_alumnos2.sh alumnos alumno_3
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

      scan "alumnos", {STARTROW => "alumno_3", STOPROW => ""}
ROW                                     COLUMN+CELL
alumno_3                               column=info:email, timestamp=1504460323334, value=alumno_3@masterbigdata.com
alumno_4                               column=info:email, timestamp=1504460323361, value=alumno_4@masterbigdata.com
alumno_5                               column=info:email, timestamp=1504460323374, value=alumno_5@masterbigdata.com
alumno_6                               column=info:email, timestamp=1504460323385, value=alumno_6@masterbigdata.com
alumno_7                               column=info:email, timestamp=1504460323410, value=alumno_7@masterbigdata.com
alumno_8                               column=info:email, timestamp=1504460323426, value=alumno_8@masterbigdata.com
alumno_9                               column=info:email, timestamp=1504460323443, value=alumno_9@masterbigdata.com
7 row(s) in 0.5520 seconds
```

6.13. Integración con PIG

En este ejemplo veremos cómo podemos integrar dos de las herramientas que hemos visto en esta unidad: PIG y Hbase.

1

1. Arrancar HBase y acceder a la consola de HBase.

```
cd /home/bigdata/hbase
```

```
bin/start-hbase.sh
```

```
jps
```

```
bin/hbase shell
```

```
bigdata@bigdata:~/hbase$ hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017
hbase(main):001:0>
```

2

2. Crear una tabla llamada personas en el namespace ejemplos con una columna llamada info.

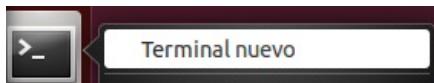
```
create 'ejercicios:personas', 'info'
```

```
hbase(main):001:0> create 'ejercicios:personas', 'info'
0 row(s) in 1.6080 seconds

=> Hbase::Table - ejercicios:personas
hbase(main):002:0>
```

3

3. Abrir otra consola.



4

4. Crear una carpeta en /home/bigdata llamada ejemplos.

```
mkdir /home/bigdata/ejemplos
```

```
bigdata@bigdata:~$ mkdir /home/bigdata/ejemplos
```

5

5. Crear un fichero llamado personas.csv que contendrá los nombres y apellidos de un grupo de personas.

```
sudo nano /home/bigdata/ejemplos/personas.csv
```

```
bigdata@bigdata:~$ sudo nano /home/bigdata/ejemplos/personas.csv
```

1, Juan, Perez

2, Maria, Sanchez

3, Pedro, Rodriguez

4, Marta, Gomez

```
bigdata@bigdata: ~
GNU nano 2.2.6      Archivo: /home/bigdata/ejemplos/personas.csv

1, Juan, Pérez
2, María, Sánchez
3, Pedro, Rodríguez
4, Marta, Gómez
```

6

6. Crear fichero PIG de carga de datos.

```
sudo nano /home/bigdata/ejemplos/carga_personas.pig
```

```
bigdata@bigdata:~$ sudo nano /home/bigdata/ejemplos/carga_personas.pig
```

```
register '/home/bigdata/hbase/lib/hbase-common-1.2.6.jar';
```

```
register '/home/bigdata/hbase/lib/hbase-client-1.2.6.jar';
```

```
register '/home/bigdata/hbase/lib/protobuf-java-2.5.0.jar';
```

-- Cargamos el fichero csv empleando el separador , en los campos id, nombre y apellidos

```
datos = LOAD 'personas.csv' USING PigStorage(',') AS (
```

```
id: chararray,
```

```
nombre: chararray,
```

```
apellidos: chararray );
```

-- Una vez cargados los datos en PIG, realizamos la carga en el namespace ejercicios dentro de la tabla personas, en la columna info.

```
STORE datos INTO 'hbase://ejercicios:personas' USING
```

```
org.apache.pig.backend.hadoop.hbase.HBaseStorage ('info:nombre info:apellidos');
```

```
bigdata@bigdata: ~/ejemplos
GNU nano 2.2.6 Archivo: ...me/bigdata/ejemplos/carga_personas.pig Modificado

-- Cargamos el fichero csv empleando el separador , en los campos id, nombre y $

datos = LOAD 'personas.csv' USING PigStorage(',') AS (
id: chararray,
nombre: chararray,
apellidos: chararray );

-- Una vez cargados los datos en PIG, realizamos la carga en el namespace ejercicios

STORE datos INTO 'hbase://ejercicios:personas' USING
org.apache.pig.backend.hadoop.hbase.HBaseStorage (
'info:nombre info:apellidos');
```

7

7. Ejecutar el script.

```
cd /home/bigdata/ejemplos
```

```
pig -x local carga_personas.pig
```

```
bigdata@bigdata:~/ejemplos$ pig -x local carga_personas.pig
17/09/03 20:05:32 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
17/09/03 20:05:32 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2017-09-03 20:05:32.551 [main] INFO org.apache.pig.Main - Apache Pig Version 0.16.0 (r1746538) compiled Jun 01 2016, 23:10:49
2017-09-03 20:05:32.551 [main] INFO org.apache.pig.Main - Logging error messages to: /home/bigdata/ejemplos/pig_1504461932549.log
2017-09-03 20:05:32.600 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - user.name is deprecated. Instead, use mapreduce.job.user.name
2017-09-03 20:05:32.605 [main] INFO org.apache.hadoop.mapred.JobTracker - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-09-03 20:05:33.205 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-09-03 20:05:33.206 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFs
2017-09-03 20:05:33.207 [main] INFO org.apache.hadoop.mapred.JobTracker - Connecting to hadoop@bigdata:54310. Connecting to hadoop@bigdata:54310. System at: file:///tmp/pig_r1504461932549_0001
2017-09-03 20:05:33.255 [main] INFO org.apache.pig.PigServer - Pig 1.1.0 is running. Please report bugs to pig.apache.org. Pig ID is 10. The session: Pig-carga_personas.pig-6523c72-d41e-449-80df-30f189f57c44
2017-09-03 20:05:33.255 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
2017-09-03 20:05:33.384 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFs
2017-09-03 20:05:33.401 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFs
2017-09-03 20:05:33.466 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFs
2017-09-03 20:05:33.636 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFs
2017-09-03 20:05:34.154 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFs
2017-09-03 20:05:34.155 [main] INFO org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (Tenured Gen) of size 0x99072512 to monitor. collectionUsageThreshold = 4895072
2017-09-03 20:05:34.330 [main] ERROR org.apache.pig.PigServer - exception during parsing: Error during parsing. Pig script failed to parse:
<file carga_personas.pig, line 14, column 0> pig script failed to validate: java.lang.RuntimeException: could not instantiate 'org.apache.pig.backend.hadoop.hbase.HBaseStorage' with arguments '[info:nomb
e info:apellidos'
Failed to parse: Pig script failed to parse:
<file carga_personas.pig, line 14, column 0> pig script failed to validate: java.lang.RuntimeException: could not instantiate 'org.apache.pig.backend.hadoop.hbase.HBaseStorage' with arguments '[info:nomb
e info:apellidos'
        at org.apache.pig.parser.QueryParserDriver.parse(QueryParserDriver.java:199)
        at org.apache.pig.PigServerGraph.parseQuery(PigServer.java:1819)
        at org.apache.pig.PigServerGraph.parseAndBuild(PigServer.java:527)
        at org.apache.pig.PigServer.parseAndBuild(PigServer.java:400)
        at org.apache.pig.PigServer.executeBatch(PigServer.java:473)
        at org.apache.pig.PigServer.executeBatch(PigServer.java:473)
```

Para que el ejemplo funcione, hay que compilar PIG con compatibilidad con HBase 1.X:<https://issues.apache.org/jira/browse/PIG-4728>

Instalamos Ant:

```
sudo apt-get install ant
```

Una vez instalado, ejecutaremos los comandos:

```
cd /home/bigdata/pig
```

```
sudo ant jar -Dhadoopversion=23 -Dhbase95.version=1.2.6
```

Este paso puede tardar varios minutos.

```
sudo rm pig-0.16.0-core-h2.jar
```

```
sudo mv pig-0.16.0-SNAPSHOT-core-h2.jar pig-0.16.0-core-h2.jar
```

Ahora si...

```
cd /home/bigdata/ejemplos
```

```
pig -x local carga_personas.pig
```

```
bigdata@bigdata:/ejemplos$ pig -x local carga_personas.pig
17/09/03 20:14:59 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
17/09/03 20:14:59 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2017-09-03 20:14:59,268 [main] INFO org.apache.pig.Main - Apache Pig version 0.16.0-SNAPSHOT (r: unknown) compiled sep 03 2017, 20:12:59
2017-09-03 20:14:59,220 [main] INFO org.apache.pig.Main - Logging error messages to: /home/bigdata/ejemplos/pig_1504462499207.log
2017-09-03 20:14:59,272 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - user.name is deprecated. Instead, use mapreduce.job.user.
2017-09-03 20:15:00,265 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapped.job.tracker is deprecated. Instead, use mapreduce.
2017-09-03 20:15:00,266 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-09-03 20:15:00,267 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: file:/.
2017-09-03 20:15:00,397 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-carga_personas.pig:a04606cf-fe2d-4518-98a6-c4d
2017-09-03 20:15:00,483 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
2017-09-03 20:15:00,609 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-09-03 20:15:00,782 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-09-03 20:15:00,947 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-09-03 20:15:01,073 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-09-03 20:15:01,746 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-09-03 20:15:02,023 [main] INFO org.apache.pig.impl.spillableMemoryManager - Selected heap (Tenured Gen) of size 699072512 to monitor.
50752
2017-09-03 20:15:02,218 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-09-03 20:15:02,668 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-09-03 20:15:02,736 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: UNKNOWN
2017-09-03 20:15:02,862 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-09-03 20:15:03,049 [main] INFO org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrun
imizer, LoadTypecastInserter, MergeFilter, MergeForEach, PartitionFilterOptimizer, PredicatePushdownOptimizer, PushDownForEachFlatten, PushUpFil
ter, PushUpForJoin, ReorderForJoin, ReorderForUpdate, ReorderForUpdateWithJoin, ReorderForUpdateWithScan, ReorderForUpdateWithScanWithJoin, Reo
rderForUpdateWithScanWithUpdate, ReorderForUpdateWithScanWithUpdateWithJoin, ReorderForUpdateWithScanWithUpdateWithScan, ReorderForUpdateWit
hScan, ReorderForUpdateWithScanWithUpdate, ReorderForUpdateWithScanWithUpdateWithJoin, ReorderForUpdateWithScanWithUpdateWithScan, ReorderForU
pForUpdateWithScanWithUpdateWithScan, ReorderForUpdateWithScanWithUpdateWithScanWithJoin, ReorderForUpdateWithScanWithUpdateWithScanWithUpda
t
2017-09-03 20:15:03,541 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100
2017-09-03 20:15:03,585 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-09-03 20:15:03,670 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimi
2017-09-03 20:15:03,680 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimi
2017-09-03 20:15:03,828 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-09-03 20:15:03,907 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - session.id is deprecated. Instead, use dfs.metrics.sessionId
2017-09-03 20:15:03,917 [main] INFO org.apache.hadoop.metrics.jvm.JvmMetrics - Initializing JVM Metrics with processName=JobTracker, sessionId=
2017-09-03 20:15:04,027 [main] INFO org.apache.pig.tools.pigstats.mapreduce.MRScriptState - Pig script settings are added to the job
```

```
HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.8.0 0.16.0-SNAPSHOT bigdata 2017-09-03 20:15:03 2017-09-03 20:15:08 UNKNOWN
Success!
Job Stats (time in seconds):
JobId Maps Reduces MaxMapTime MinMapTime AvgMapTime MedianMapTime MaxReduceTime MinReduceTime AvgReduceTime MedianReducetime Alias Feature Outputs
job_local1864560052_0001 1 0 n/a n/a n/a 0 0 0 0 datos MAP ONLY hbase://ejercicios:personas,
Input(s):
Successfully read 4 records from: "file:///home/bigdata/ejemplos/personas.csv"
Output(s):
Successfully stored 4 records in: "hbase://ejercicios:personas"
Counters:
Total records written : 4
Total bytes written : 144
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total Records proactively spilled: 0
Job DAG:
job_local1864560052_0001

2017-09-03 20:15:08,961 [main] INFO org.apache.hadoop.metrics.jvm.JvmMetrics - Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already initialized
2017-09-03 20:15:08,968 [main] INFO org.apache.hadoop.metrics.jvm.JvmMetrics - Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already initialized
2017-09-03 20:15:08,980 [main] INFO org.apache.hadoop.metrics.jvm.JvmMetrics - Cannot initialize JVM Metrics with processName=Jobtracker, sessionId= - already initialized
2017-09-03 20:15:09,016 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapreduce.MapReduceLauncher - Success!
2017-09-03 20:15:09,087 [main] INFO org.apache.pig.Main - Pig script completed in 10 seconds and 783 milliseconds (10783 ms)
bigdata@bigdata:-/ejemplos$
```

8

8. Consultamos los datos en HBase.

```
scan 'ejercicios:personas'
```

```
hbase(main):001:0> scan 'ejercicios:personas'
ROW                                     COLUMN+CELL
1                                         column=info:apellidos, timestamp=1504462507818, value= Perez
1                                         column=info:nombre, timestamp=1504462507818, value= Juan
2                                         column=info:apellidos, timestamp=1504462507844, value= Sanchez
2                                         column=info:nombre, timestamp=1504462507844, value= Maria
3                                         column=info:apellidos, timestamp=1504462507844, value= Rodr\xC3\xADguez
3                                         column=info:nombre, timestamp=1504462507844, value= Pedro
4                                         column=info:apellidos, timestamp=1504462507851, value= Gomez
4                                         column=info:nombre, timestamp=1504462507851, value= Marta
4 row(s) in 0.5840 seconds
hbase(main):002:0>
```

6.14. Lectura recomendada



Se recomienda leer la documentación original de la Fundación Apache para profundizar en los contenidos: [Apache HBase™ Reference Guide¹⁷](#).



Apache HBase™ Reference Guide

Apache HBase Team

Version 2.0.0-SNAPSHOT

¹⁷También se puede descargar en formato pdf en el siguiente enlace:
http://hbase.apache.org/apache_hbase_reference_guide.pdf

VII. Y esto no termina aquí...

Apache Hadoop ha sido un punto de inflexión en cuanto a sistemas de procesamiento paralelo se refiere, pero el camino no termina aquí. Desde su publicación, han ido saliendo *frameworks* que intentan mejorar o aportar nuevas funcionalidades a Hadoop como es el caso de Spark, Flink, etc.

Para seguir avanzando en el aprendizaje, es conveniente adentrarse también en estas tecnologías.

Engine comparison				
	 Hadoop MapReduce	 TEZ	 Spark	 Flink
API	MapReduce on k/v pairs	k/v pair Readers/Writers	Transformations on k/v pair collections	Iterative transformations on collections
Paradigm	MapReduce	DAG	RDD	Cyclic dataflows
Optimization	none	none	Optimization of SQL queries	Optimization in all APIs
Execution	Batch sorting	Batch sorting and partitioning	Batch with memory pinning	Stream with out-of-core algorithms

Figura 3. Comparación de motores. Fuente: Fundación Apache.

VIII. Resumen



En esta unidad hemos visto algunas de las herramientas que complementan a los elementos principales en el ecosistema Hadoop, en concreto:

- PIG, un lenguaje que nos permite practicar análisis de datos sobre información que resida en el clúster.
- HIVE que, como *datawarehouse* sobre Hadoop, nos permite la construcción de grandes tablas a partir de información de Hadoop para consultar a través de HQL, un lenguaje de consultas similar al SQL del mundo relacional.
- SQOOP, para la importación de datos desde el mundo relacional a HDFS y HBase, una base de datos NoSQL que opera sobre HDFS y por lo tanto se beneficia de todas sus características (escalabilidad, tolerancia a fallos, redundancia, etc.).

IX. Anexos

Se recomienda la lectura de los anexos 1 y 2, que se pueden encontrar en la sección de Recursos de esta unidad:

- **ANEXO 1. ARQUITECTURA**
- **ANEXO 2. CONFIGURACIÓN DE UN CLÚSTER EN HADOOP**

Ejercicios

Ejercicio 1: PIG



Objetivo: poner en práctica los conocimientos adquiridos sobre la herramienta para resolver el ejercicio.

Utilizando PIG y partiendo de la discografía de Pink Floyd (año, nombre del disco, ranking EE. UU., ranking Reino Unido):



Anotación: Discografía de Pink Floyd

- 1967, The Piper at the Gates of Dawn, 131,6
- 1968, A Saucerful of Secrets, 999,9
- 1969, Music from the Film More, 153,9
- 1969, Ummagumma, 74,5
- 1970, Atom Heart Mother, 55,1
- 1972, Obscured by Clouds, 46,6
- 1973, The Dark Side of the Moon, 1,1
- 1975, Wish you Were Here, 1,1
- 1977, Animals, 3,2
- 1979, The Wall, 1,3
- 1983, The Final Cut, 6,1
- 1987, A Momentary Lapse of Reason, 3,3
- 1994, The Division Bell, 1,1
- 2014, The Endless River, 3, 1

Se pide indicar los comandos empleados para resolver las siguientes preguntas:

1. Crear un fichero llamado discos.txt.
2. Arrancar HDFS, Yarn y el job history.
3. Subir el fichero a HDFS dentro de la carpeta/ejerciciosPig/discos.txt.
4. Ejecutar la instrucción ls sobre Hadoop para indicar el tamaño del fichero.
5. Arrancar PIG en modo distribuido (si se desea eliminar trazas de log) y ejecutar el siguiente comando: cat hdfs://localhost:9000/ejerciciosPig/discos.txt para confirmar que los primeros puntos han funcionado correctamente y el fichero está subido a HDFS.
6. Cargar el fichero de HDFS en una variable llamada discos.
7. Calcular los discos que no estuvieron ni en el top 10 de EE. UU. ni en el top 10 de Reino Unido (indicar también el resultado).

8. Obtener la máxima y mínima posición que ocuparon los discos de Pink Floyd en EE. UU. y en Reino Unido (indicar también el resultado) empleando los comandos de Latin Pig.
9. Explica con tus propias palabras lo que se pretende obtener con los siguientes comandos e indica el resultado obtenido:
a = foreach discos generate anio;
b = distinct a;
dump b;
10. (OPCIONAL) Realiza el ejercicio 8 empleando las funciones de piggybank (se recomienda consultar la siguiente página web: <https://pig.apache.org/docs/r0.15.0/api/org/apache/pig/EvalFunc.html>).

Solución

1

1. Crear un fichero llamado discos.txt:

Se crea el fichero discos.txt en una carpeta nueva ejerciciosPig:

```
mkdir ejerciciosPig
```

```
bigdata@bigdata:~/pig$ mkdir ejerciciosPig
bigdata@bigdata:~/pig$ ls
bin          docs      legacy      NOTICE.txt
build.xml    ejemplos  lib         pig-0.15.0-core-h1.j
CHANGES.txt  [ejerciciosPig] lib-src    pig-0.15.0-core-h2.j
conf         ivy       license   pig_1457552356619.lo
contrib      ivy.xml   LICENSE.txt  pig_1457553266020.lo
```

```
sudo nano ejerciciosPig/discos.txt
```

GNU nano 2.2.6	Archivo: discos.txt
1967, The Piper at the Gates of Dawn,131,6 1968, A Saucerful of Secrets,999,9 1969, Music from the Film More,153,9 1969, Ummagumma,74,5 1970, Atom Heart Mother,55,1 1972, Obscured by Clouds, 46,6 1973, The Dark Side of the Moon, 1,1 1975, Wish you Were Here, 1,1 1977, Animals, 3,2 1979, The Wall, 1,3 1983, The Final Cut, 6,1 1987, A Momentary Lapse of Reason,3,3 1994, The Division Bell, 1,1 2014, The Endless River, 3, 1	

```
bigdata@bigdata:~/pig/ejerciciosPig$ sudo nano discos.txt
bigdata@bigdata:~/pig/ejerciciosPig$ cat discos.txt
1967, The Piper at the Gates of Dawn,131,6
1968, A Saucerful of Secrets,999,9
1969, Music from the Film More,153,9
1969, Ummagumma,74,5
1970, Atom Heart Mother,55,1
1972, Obscured by Clouds, 46,6
1973, The Dark Side of the Moon, 1,1
1975, Wish you Were Here, 1,1
1977, Animals, 3,2
1979, The Wall, 1,3
1983, The Final Cut, 6,1
1987, A Momentary Lapse of Reason,3,3
1994, The Division Bell, 1,1
2014, The Endless River, 3, 1
```

2

2. Arrancar HDFS, Yarn y el job history:

Comprobar que están ya arrancados los servicios de HDFS, Yarn y el job history:

```
bigdata@bigdata:~/pig$ jps
2510 NameNode
5584 JobHistoryServer
3017 ResourceManager
2447 Jps
2796 SecondaryNameNode
2632 DataNode
3142 NodeManager
915 HMaster
bigdata@bigdata:~/pig$
```

3

3. Subir el fichero a HDFS dentro de la carpeta /ejerciciosPig/discos.txt:

Subir el fichero a HDFS:

```
hdfs dfs -put ejerciciosPig/discos.txt hdfs://localhost:9000/ejerciciosPig
```

```
bigdata@bigdata:~/pig$ hdfs dfs -put ejerciciosPig/discos.txt hdfs://localhost:9000/ejerciciosPig
16/03/13 11:47:17 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
bigdata@bigdata:~/pig$
```

4

4. Ejecutar la instrucción ls sobre Hadoop para indicar el tamaño del fichero:

```
hdfs dfs -ls -R /ejerciciosPig
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -ls -R /ejerciciosPig
16/03/13 12:52:26 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
bigdata@bigdata:~/hadoop$ -rw-r--r-- 1 bigdata supergroup 424 2016-03-13 12:50 /ejerciciosPig/discos.txt
```

El tamaño del fichero es **424**.

5. Arrancar PIG en modo distribuido (si se desea eliminar trazas de log) y ejecutar el siguiente comando: cat hdfs://localhost:9000/ejerciciosPig/discos.txt para confirmar que los primeros puntos han funcionado correctamente y el fichero está subido a HDFS:

Acceder a PIG en modo distribuido:

```
pig -4 conf/nolog.conf -x mapreduce
```

Comprobar que el fichero está correctamente subido a HDFS:

```
cat hdfs://localhost:9000/ejerciciosPig/discos.txt
```

```
bigdata@bigdata:~/pig$ pig -4 conf/nolog.conf -x mapreduce
16/03/13 12:53:52 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
16/03/13 12:53:52 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
16/03/13 12:53:52 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
16/03/13 12:53:52 INFO pig.Main: Loaded log4j properties from file: conf/nolog.conf
grunt> REGISTER '/home/bigdata/mapreduce/wc.jar';
grunt> DEFINE MY_UDF com.myudfs.UPPER();
grunt> SET default_parallel 10;
grunt> cat hdfs://localhost:9000/ejerciciosPig/discos.txt
1967, The Piper at the Gates of Dawn,131,6
1968, A Saucerful of Secrets,999,9
1969, Music from the Film More,153,9
1969, Ummagumma,74,5
1970, Atom Heart Mother,55,1
1972, Obscured by Clouds, 46,6
1973, The Dark Side of the Moon, 1,1
1975, Wish you Were Here, 1,1
1977, Animals, 3,2
1979, The Wall, 1,3
1983, The Final Cut, 6,1
1987, A Momentary Lapse of Reason,3,3
1994, The Division Bell, 1,1
2014, The Endless River, 3, 1
grunt>
```

6. Cargar el fichero de HDFS en una variable llamada discos:

Cargar el fichero en la variable discos:

```
discos = load 'ejerciciosPig/discos.txt' using PigStorage(',') as (anio:int, disco:chararray, eeuu:int, uk:int);
```

```
grunt> discos = load 'ejerciciosPig/discos.txt' using PigStorage(',') as (anito:int, disco:chararray, eeuu:int, uk:int);
2016-03-13 18:03:06,764 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFs
2016-03-13 18:03:06,810 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt>
```

Comprobar que se ha cargado bien:

describe discos

illustrate discos (mostramos un ejemplo de la carga con illustrate)

```
grunt> discos = load 'ejerciciosPig/dtscos.txt' using PigStorage(',') as (anio:int, disco:chararray, eeuu:int, uk:int);
2016-03-13 18:03:06,764 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFs
2016-03-13 18:03:06,810 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytesPerChecksum
grunt> describe discos
discos: {anio: int,disco: chararray,eeuu: int,uk: int}
grunt> illustrate discos
```

discos	anio:int	disco:chararray	eeuu:int	uk:int
	1979	The Wall	1	3

dump discos

```
(1967, The Piper at the Gates of Dawn,131,6)
(1968, A Saucerful of Secrets,999,9)
(1969, Music from the Film More,153,9)
(1969, Ummagumma,74,5)
(1970, Atom Heart Mother,55,1)
(1972, Obscured by Clouds,46,6)
(1973, The Dark Side of the Moon,1,1)
(1975, Wish you Were Here,1,1)
(1977, Animals,3,2)
(1979, The Wall,1,3)
(1983, The Final Cut,6,1)
(1987, A Momentary Lapse of Reason,3,3)
(1994, The Division Bell,1,1)
(2014, The Endless River,3,1)
grunt> █
```

7

7. Calcular los discos que no estuvieron ni en el top 10 de EE. UU. ni en el top 10 de Reino Unido (indicar también el resultado):

Para calcular los discos que no han estado en el top 10 ni de EE. UU. ni de Reino Unido, hay que filtrar por ambos:

```
discos_notop = filter discos by eeuu > 10 and uk > 10;
```

```
dump discos_notop;
```

```
grunt> discos_notop = filter discos by eeuu > 10 and uk > 10;
grunt> dump discos_notop;
grunt> ■
```

No hay ningún disco que no obtuviera ni en el top 10 de EE. UU. ni en el top 10 de Reino Unido.

8

8. Obtener la máxima y mínima posición que ocuparon los discos de Pink Floyd en EE. UU. y en Reino Unido (indicar también el resultado) empleando los comandos de Latin Pig:

```
rank_eedu = order discos by eedu desc;
```

```
rank_uk = order discos by uk desc;
```

```
min_eedu = limit rank_eedu 1;
```

```
min_uk = limit rank_uk 1;
```

```
rank2_eedu = order discos by eedu;
```

```
rank2_uk = order discos by uk;
```

```
max_eedu = limit rank2_eedu 1;
```

```
max_uk = limit rank2_uk 1;
```

```
grunt> rank_eedu = order discos by eedu desc;
grunt> rank_uk = order discos by uk desc;
```

```
grunt> min_eedu = limit rank_eedu 1;
grunt> min_uk = limit rank_uk 1;
grunt> rank2_eedu = order discos by eedu;
grunt> rank2_uk = order discos by uk;
```

```
grunt> max_eedu = limit rank2_eedu 1;
grunt> max_uk = limit rank2_uk 1;
```

La máxima y mínima posición que ocuparon los discos de Pink Floyd en EE. UU.:

Máxima: 1

Mínima: 999

```
dump max_eeuu;
```

```
dump min_eeuu;
```

```
grunt> dump max_eeuu;
(1994, The Division Bell,1,1)
grunt> dump min_eeuu;
(1968, A Saucerful of Secrets,999,9)
```

La máxima y mínima posición que ocuparon los discos de Pink Floyd en Reino Unido:

Máxima: 1

Mínima: 999

```
dump max_uk;
```

```
dump min_uk;
```

```
grunt> dump max_uk;
(1983, The Final Cut,6,1)
grunt> dump min_uk;
(1968, A Saucerful of Secrets,999,9)
```

9

9. Explica con tus propias palabras lo que se pretende obtener con los siguientes comandos e indica el resultado obtenido:

a = foreach discos generate anio;

b = distinct a;

dump b;

Con **foreach** obtenemos los datos de la columna anio de la variable discos, es decir, los años en los que se ha publicado un disco.

Con **distinct** eliminamos duplicados de la variable que contiene los años.

Con **dump** mostramos el resultado obtenido.

```
grunt> a = foreach discos generate anio;
grunt> dump a;
(1967)
(1968)
(1969)
(1969)
(1970)
(1972)
(1973)
(1975)
(1977)
(1979)
(1983)
(1987)
(1994)
(2014)
grunt> b = distinct a;
grunt> dump b;
(1973)
(1983)
(1967)
(1977)
(1987)
(1969)
(1979)
(1970)
(1994)
(2014)
(1968)
(1975)
(1972)
grunt>
```

10

10 . Realiza el ejercicio 8 empleando las funciones de piggybank (opcional)
<https://pig.apache.org/docs/r0.15.0/api/org/apache/pig/EvalFunc.html>

Registro la librería de piggybank:

```
grunt> REGISTER /home/bigdata/pig/lib/piggybank.jar
```

```
a = foreach discos generate eeuu;
```

```
dump a;
```

```
max_euuu = foreach a GENERATE org.apache.pig.piggybank.evaluation.math.MAX (eeuu);
```

```
min_euuu = foreach a GENERATE org.apache.pig.piggybank.evaluation.math.MIN (eeuu);
```

```
b = foreach discos generate uk;
```

```
dump b;
```

```
max_uk = foreach a GENERATE org.apache.pig.piggybank.evaluation.math.MAX (uk);
```

```
min_uk = foreach a GENERATE org.apache.pig.piggybank.evaluation.math.MIN (uk);
```

Ejercicio 2: HIVE



Objetivo: poner en práctica los conocimientos adquiridos sobre la herramienta para resolver el ejercicio.

Utilizando HIVE y partiendo de la discografía de Pink Floyd (año, nombre del disco, ranking EE. UU., ranking Reino Unido):

s.l.



Anotación: discografía de Pink Floyd

- 1967, The Piper at the Gates of Dawn, 131,6
- 1968, A Saucerful of Secrets, 999,9
- 1969, Music from the Film More, 153,9
- 1969, Ummagumma, 74,5
- 1970, Atom Heart Mother, 55,1
- 1972, Obscured by Clouds, 46,6
- 1973, The Dark Side of the Moon, 1,1
- 1975, Wish you Were Here, 1,1
- 1977, Animals, 3,2
- 1979, The Wall, 1,3
- 1983, The Final Cut, 6,1
- 1987, A Momentary Lapse of Reason, 3,3
- 1994, The Division Bell, 1,1
- 2014, The Endless River, 3, 1

Se pide indicar los comandos empleados para resolver las siguientes preguntas:

1. Crear un fichero de texto con la información anterior (**importante**: al crear el fichero, se debe tener cuidado con los caracteres al final de línea).
2. Crear una tabla en HIVE que permita almacenar los datos anteriores indicando que el formato de separación es como el siguiente, de tipo tabulación (create table(.....) row format delimited fields terminated by ';' stored as textfile;)
3. Cargar el fichero de texto.
4. Acceder a HIVE y ejecutar una consulta sencilla (select *) para verificar que hay datos y se han cargado correctamente. En caso contrario, volver a cargar los datos.
5. Calcular los discos que estuvieron a la vez entre los cinco primeros puestos en EE. UU. y Reino Unido.
6. (OPCIONAL) Obtener la máxima y mínima posición que ocuparon los discos de Pink Floyd en EE. UU. y en Reino Unido (por ejemplo, empleando los comandos order y limit en dos sentencias).

Solución

Partiendo de la discografía de Pink Floyd (año, nombre del disco, ranking EE. UU., ranking Reino Unido) indicar los comandos empleados para resolver las siguientes preguntas:

1

1. Crear un fichero de texto con la información anterior (IMPORTANTE: al crear el fichero tener cuidado con los caracteres al final de línea):

Crear el fichero de texto con la información de la discografía:

sudo nano discografia

```
bigdata@bigdata:~/ejemplosHive$ sudo nano discografia
[sudo] password for bigdata:
Lo sentimos, vuelva a intentarlo.
[sudo] password for bigdata:
```

GNU nano 2.2.6	Archivo: discografia
1967,The Piper at the Gates of Dawn,131,6 1968,A Saucerful of Secrets,999,9 1969,Music from the Film More,153,9 1969,Ummagumma,74,5 1970,Atom Heart Mother,55,1 1972,Obscured by Clouds,46,6 1973,The Dark Side of the Moon,1,1 1975,Wish you Were Here,1,1 1977,Animals,3,2 1979,The Wall,1,3 1983,The Final Cut,6,1 1987,A Momentary Lapse of Reason,3,3 1994,The Division Bell,1,1 2014,The Endless River,3,1	

2

2. Crear una tabla en HIVE que permita almacenar los datos anteriores indicando que el formato de separación es como el siguiente, de tipo tabulación (create table (.....) row format delimited fields terminated by '' stored as textfile;):

Arrancar HIVE:

hive

Crear una tabla para almacenar la discografía con los cuatro campos necesarios (año, disco, ranking EE. UU. y ranking Reino Unido):

create table discografia (

anio INT,

disco STRING,

eeuu INT,

uk INT)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

STORED AS TEXTFILE;

```
hive> create table discografia (
    >     anio INT,
    >     disco STRING,
    >     eeuu INT,
    >     uk INT)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LINES TERMINATED BY '\n'
    > STORED AS TEXTFILE;
OK
Time taken: 1.215 seconds
```

Comprobar que se ha creado:

show tables;

```
hive> show tables;
OK
Comments:
discografia
docs
empleados1
empleados2
empleados3
empleados4
test
word_counts
Time taken: 0.204 seconds, Fetched: 9 row(s)
```

3

3. Cargar el fichero de texto:

Cargar el fichero discografía en la tabla discografía:

```
LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/discografia' INTO TABLE
discografia;
```

```
hive> LOAD DATA LOCAL INPATH '/home/bigdata/ejemplosHive/discografia' INTO TABLE discografia;
Loading data to table default.discografia
Table default.discografia stats: [numFiles=1, numRows=0, totalSize=401, rawDataSize=0]
OK
Time taken: 2.35 seconds
```

4

4. Acceder a HIVE y ejecutar una consulta sencilla (select *) para verificar que hay datos y se han cargado correctamente. En caso contrario, volver a cargar los datos:

Consultar los datos de la tabla con un select *:

```
select * from discografia;
```

```
hive> select * from discografia;
OK
1967    The Piper at the Gates of Dawn  131      6
1968    A Saucerful of Secrets  999      9
1969    Music from the Film More   153      9
1969    Ummagumma     74      5
1970    Atom Heart Mother    55      1
1972    Obscured by Clouds    46      6
1973    The Dark Side of the Moon  1      1
1975    Wish you Were Here    1      1
1977    Animals 3            2
1979    The Wall             1      3
1983    The Final Cut       6      1
1987    A Momentary Lapse of Reason  3      3
1994    The Division Bell    1      1
2014    The Endless River   3      1
Time taken: 0.631 seconds, Fetched: 14 row(s)
hive> ■
```

5

5. Calcular los discos que estuvieron a la vez entre los cinco primeros puestos en EE. UU. y Reino Unido:

```
select * from discografia where eeuu <= 5 and uk <= 5;
```

```
hive> select * from discografia where eeuu <= 5 and uk <= 5;
OK
1973    The Dark Side of the Moon      1      1
1975    Wish you Were Here     1      1
1977    Animals 3            2
1979    The Wall             1      3
1987    A Momentary Lapse of Reason  3      3
1994    The Division Bell    1      1
2014    The Endless River   3      1
Time taken: 0.626 seconds, Fetched: 7 row(s)
hive> ■
```

6

6. (OPCIONAL) Obtener la máxima y mínima posición que ocuparon los discos de Pink Floyd en EE. UU. y en Reino Unido (por ejemplo, empleando los comandos order y limit en dos sentencias):

Usar los operadores max y min respectivamente para obtener lo solicitado:

```
select max(eeuu) as MaxEEUU, max(uk) as MaxUK from discografia;
```

```
hive> select * from discografia where eeuu <= 5 and uk <= 5;
OK
1973    The Dark Side of the Moon      1      1
1975    Wish you Were Here      1      1
1977    Animals 3      2
1979    The Wall      1      3
1987    A Momentary Lapse of Reason      3      3
1994    The Division Bell      1      1
2014    The Endless River      3      1
Time taken: 0.626 seconds, Fetched: 7 row(s)
```

Obtenemos **999** para EE. UU. y **9** para Reino Unido.

```
select min(eeuu) as MinEEUU, min(uk) as MinUK from discografia;
```

```
hive> select min(eeuu) as MinEEUU, min(uk) as MinUK from discografia;
Query ID = bigdata_20160313141931_dc730a7d-d7f0-41d7-b375-30ffcf69e0f9
Total jobs = 1
Launching Job 1 out of 1
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_1457173731478_0018, Tracking URL = http://bigdata:8088/proxy/application_1457173731478_0018/
Kill Command = /home/bigdata/hadoop/bin/hadoop job -kill job_1457173731478_0018
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-03-13 14:19:47,428 Stage-1 map = 0%,  reduce = 0%
2016-03-13 14:20:03,429 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.97 sec
2016-03-13 14:20:22,040 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.17 sec
MapReduce Total cumulative CPU time: 5 seconds 170 msec
Ended Job = job_1457173731478_0018
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1  Cumulative CPU: 5.17 sec  HDFS Read: 7914 HDFS Write: 4 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 170 msec
OK
1      1
Time taken: 53.359 seconds, Fetched: 1 row(s)
hives
```

Obtenemos **1** para EE. UU. y **1** para Reino Unido.

Ejercicio 3: HBASE



Objetivo: poner en práctica los conocimientos adquiridos sobre la herramienta para resolver el ejercicio.

HBASE

Partiendo de los datos de la discografía de álbumes de estudio de Pink Floyd (que se puede consultar en el siguiente enlace de Wikipedia: http://es.wikipedia.org/wiki/Anexo:Discograf%C3%ADa_de_Pink_Floyd) Año Álbum Ranking Certificación EE. UU. Ranking Certificación Reino Unido:

The screenshot shows a note-taking application with a red header bar. The main content area has a light gray background. On the left side, there is a vertical sidebar with a red header and a white body containing a large blue bookmark icon. The main content area starts with the title "Anotación: discografía Pink Floyd" in bold black font. Below the title is a bulleted list of albums, each with a small red arrow to its left. The list includes:

- 1967 The Piper at the Gates of Dawn 131 6
- 1968 A Saucerful of Secrets 999 9
- 1969 Music from the Film More 153 9
- 1969 Ummagumma (Disco 2) 74 5
- 1970 Atom Heart Mother 55 1
- 1971 Meddle 70 3
- 1972 Obscured by Clouds 46 6
- 1973 The Dark Side of the Moon 1 1
- 1975 Wish You Were Here 1 1
- 1977 Animals 3 2
- 1979 The Wall 1 3
- 1983 The Final Cut 6 1
- 1987 A Momentary Lapse of Reason 3 3
- 1994 The Division Bell 1 1
- 2014 The Endless River 3 1

Se pide resolver las siguientes cuestiones:

1. Crear un namespace llamado discografía.
2. Crear una tabla llamada pink_floyd que contenga solo dos columnas: nombre_disco, ranking (a su vez tendrá identificadores para EE. UU. y Reino Unido).
3. Cargar los datos en HBase (se pueden emplear los comandos put o una importación desde PIG). Nota: algunos años hay más de un disco por lo que la clave será el año y un secuencial, y no están disponibles todos los datos de *rankings*, por lo que esos datos nulos no será necesario cargarlos.
4. Escribir la instrucción y el resultado de consultar el álbum del año 1968.

Solución

Partiendo de los datos de la discografía de álbumes de estudio de Pink Floyd http://es.wikipedia.org/wiki/Anexo:Discograf%C3%ADa_de_Pink_Floyd

Año	Álbum	Ranking	
		Ranking Certificación EEUU	Ranking Certificación UK
1967	The Piper at the Gates of Dawn	131	6
1968	A Saucerful of Secrets	—	9
1969	Music from the Film More	153	9
1969	Ummagumma (Disco 2)	74	5
1970	Atom Heart Mother	55	1
1971	Meddle	70	3
1972	Obscured by Clouds	46	6
1973	The Dark Side of the Moon	1	1
1975	Wish You Were Here	1	1

Resolver las siguientes cuestiones:

1

1. Crear un namespace llamado discografía:

Arrancar HBASE:

bin/start-hbase.sh

jps

```
bigdata@bigdata:~/hbase$ bin/start-hbase.sh
starting master, logging to /home/bigdata/hbase/logs/hbase-bigdata-master-bigdata.out
bigdata@bigdata:~/hbase$ jps
2510 NameNode
6584 JobHistoryServer
3017 ResourceManager
2796 SecondaryNameNode
2632 DataNode
3142 NodeManager
915 HMaster
981 Jps
bigdata@bigdata:~/hbase$
```

Acceder a la consola y comprobar el estado:

bin/hbase shell

status

```
bigdata@bigdata:~/hbase$ bin/hbase shell
2016-03-13 09:46:44,876 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.2, rcc2b70cf03e3378800661ec5cab11eb43faf0fc, Wed Aug 26 20:11:27 PDT 2015

hbase(main):001:0> status
1 servers, 1 dead, 2.0000 average load
```

Crear el namespace:

```
create_namespace 'discografia'
```

```
hbase(main):002:0> create_namespace 'discografia'  
0 row(s) in 0.3000 seconds
```

2

2. Crear una tabla llamada pink_floyd que contenga solo dos columnas: nombre_disco, ranking (a su vez tendrá identificadores para EE. UU. y Reino Unido).

Crear la tabla con sus campos:

```
create 'discografia:pink_floyd', 'nombre_disco', 'ranking'
```

```
hbase(main):003:0> create 'discografia:pink_floyd', 'nombre_disco', 'ranking'  
0 row(s) in 2.5310 seconds  
=> Hbase::Table - discografia:pink_floyd
```

Comprobar que se ha creado correctamente con sus dos campos: nombre_disco y ranking.

```
describe 'discografia:pink_floyd'
```

```
hbase(main):005:0> describe 'discografia:pink_floyd'  
Table discografia:pink_floyd is ENABLED  
discografia:pink_floyd  
COLUMN FAMILIES DESCRIPTION  
{NAME => 'nombre_disco', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE',  
MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}  
{NAME => 'ranking', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_  
VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}  
2 row(s) in 0.4930 seconds
```

3

3. Cargar los datos en HBase (se pueden emplear los comandos put o una importación desde PIG). Nota: algunos años hay más de un disco, por lo que la clave será el año y un secuencial, y no están disponibles todos los datos de rankings por lo que esos datos nulos no será necesario cargarlos.

Cargar los datos en la tabla con el comando put:

```
put 'discografia:pink_floyd', '1967#1', 'nombre_disco', 'The Piper at the Gates of Dawn'
```

```
put 'discografia:pink_floyd', '1967#1', 'ranking:EEUU', 131
```

```
put 'discografia:pink_floyd', '1967#1', 'ranking:UK', 6
```

```
put 'discografia:pink_floyd', '1968#1', 'nombre_disco', 'A Saucerful of Secrets'
```

```
put 'discografia:pink_floyd', '1968#1', 'ranking:UK', 9
```

```
put 'discografia:pink_floyd', '1969#1', 'nombre_disco', 'Music from the Film More  
put 'discografia:pink_floyd', '1969#1', 'ranking:EEUU', 153  
put 'discografia:pink_floyd', '1969#1', 'ranking:UK', 9  
put 'discografia:pink_floyd', '1969#2', 'nombre_disco', 'Ummagumma (Disco 2)'  
put 'discografia:pink_floyd', '1969#2', 'ranking:EEUU', 74  
put 'discografia:pink_floyd', '1969#2', 'ranking:UK', 5  
put 'discografia:pink_floyd', '1970#1', 'nombre_disco', 'Atom Heart Mother'  
put 'discografia:pink_floyd', '1970#1', 'ranking:EEUU', 55  
put 'discografia:pink_floyd', '1970#1', 'ranking:UK', 1  
put 'discografia:pink_floyd', '1971#1', 'nombre_disco', 'Meddle'  
put 'discografia:pink_floyd', '1971#1', 'ranking:EEUU', 70  
put 'discografia:pink_floyd', '1971#1', 'ranking:UK', 3  
put 'discografia:pink_floyd', '1972#1', 'nombre_disco', 'Obscured by Clouds'  
put 'discografia:pink_floyd', '1972#1', 'ranking:EEUU', 46  
put 'discografia:pink_floyd', '1972#1', 'ranking:UK', 6  
put 'discografia:pink_floyd', '1973#1', 'nombre_disco', 'The Dark Side of the Moon'  
put 'discografia:pink_floyd', '1973#1', 'ranking:EEUU', 1  
put 'discografia:pink_floyd', '1973#1', 'ranking:UK', 1  
put 'discografia:pink_floyd', '1975#1', 'nombre_disco', 'Wish You Were Here'  
put 'discografia:pink_floyd', '1975#1', 'ranking:EEUU', 1  
put 'discografia:pink_floyd', '1975#1', 'ranking:UK', 1  
put 'discografia:pink_floyd', '1977#1', 'nombre_disco', 'Animals'  
put 'discografia:pink_floyd', '1977#1', 'ranking:EEUU', 3  
put 'discografia:pink_floyd', '1977#1', 'ranking:UK', 2  
put 'discografia:pink_floyd', '1979#1', 'nombre_disco', 'The Wall'
```

```

put 'discografia:pink_floyd', '1979#1', 'ranking:EEUU', 1
put 'discografia:pink_floyd', '1979#1', 'ranking:UK', 3
put 'discografia:pink_floyd', '1983#1', 'nombre_disco', 'The Final Cut'
put 'discografia:pink_floyd', '1983#1', 'ranking:EEUU', 6
put 'discografia:pink_floyd', '1983#1', 'ranking:UK', 1
put 'discografia:pink_floyd', '1987#1', 'nombre_disco', 'A Momentary Lapse of Reason'
put 'discografia:pink_floyd', '1987#1', 'ranking:EEUU', 3
put 'discografia:pink_floyd', '1987#1', 'ranking:UK', 3
put 'discografia:pink_floyd', '1994#1', 'nombre_disco', 'The Division Bell'
put 'discografia:pink_floyd', '1994#1', 'ranking:EEUU', 1
put 'discografia:pink_floyd', '1994#1', 'ranking:UK', 1
put 'discografia:pink_floyd', '2014#1', 'nombre_disco', 'The Endless River'
put 'discografia:pink_floyd', '2014#1', 'ranking:EEUU', 3
put 'discografia:pink_floyd', '2014#1', 'ranking:UK', 1

hbase(main):008:0> put 'discografia:pink_floyd', '1967#1', 'nombre_disco', 'The Piper at the Gates of Dawn'
0 row(s) in 0.2950 seconds

hbase(main):010:0> put 'discografia:pink_floyd', '1967#1', 'ranking:EEUU', 131
0 row(s) in 0.0560 seconds

hbase(main):011:0> put 'discografia:pink_floyd', '1967#1', 'ranking:UK', 6
0 row(s) in 0.0510 seconds

hbase(main):012:0> put 'discografia:pink_floyd', '1968#1', 'nombre_disco', 'A Saucerful of Secrets'
0 row(s) in 0.0560 seconds

hbase(main):013:0> put 'discografia:pink_floyd', '1968#1', 'ranking:UK', 9
0 row(s) in 0.0680 seconds

hbase(main):014:0> put 'discografia:pink_floyd', '1969#1', 'nombre_disco', 'Music from the Film More'
0 row(s) in 0.0330 seconds

hbase(main):015:0> put 'discografia:pink_floyd', '1969#1', 'ranking:EEUU', 153
0 row(s) in 0.0230 seconds

hbase(main):016:0> put 'discografia:pink_floyd', '1969#1', 'ranking:UK', 9
0 row(s) in 0.0600 seconds

hbase(main):017:0> put 'discografia:pink_floyd', '1969#2', 'nombre_disco', 'Ummagumma (Disco 2)'
0 row(s) in 0.0440 seconds

hbase(main):018:0> put 'discografia:pink_floyd', '1969#2', 'ranking:EEUU', 74
0 row(s) in 0.0320 seconds

hbase(main):019:0> put 'discografia:pink_floyd', '1969#2', 'ranking:UK', 5
0 row(s) in 0.0520 seconds

```

```
hbase(main):020:0> put 'discografia:pink_floyd', '1970#1', 'nombre_disco', 'Atom Heart Mother'  
0 row(s) in 0.0320 seconds  
  
hbase(main):021:0> put 'discografia:pink_floyd', '1970#1', 'ranking:EEUU', 55  
0 row(s) in 0.0300 seconds  
  
hbase(main):022:0> put 'discografia:pink_floyd', '1970#1', 'ranking:UK', 1  
0 row(s) in 0.0590 seconds  
  
hbase(main):023:0> put 'discografia:pink_floyd', '1971#1', 'nombre_disco', 'Meddle'  
0 row(s) in 0.0450 seconds  
  
hbase(main):024:0> put 'discografia:pink_floyd', '1971#1', 'ranking:EEUU', 70  
0 row(s) in 0.2080 seconds  
  
hbase(main):025:0> put 'discografia:pink_floyd', '1971#1', 'ranking:UK', 3  
0 row(s) in 0.0590 seconds  
  
hbase(main):026:0> put 'discografia:pink_floyd', '1972#1', 'nombre_disco', 'Obscured by Clouds'  
0 row(s) in 0.0730 seconds  
  
hbase(main):027:0> put 'discografia:pink_floyd', '1972#1', 'ranking:EEUU', 46  
0 row(s) in 0.0210 seconds  
  
hbase(main):028:0> put 'discografia:pink_floyd', '1972#1', 'ranking:UK', 6  
0 row(s) in 0.0290 seconds
```

```
hbase(main):029:0> put 'discografia:pink_floyd', '1973#1', 'nombre_disco', 'The Dark Side of the Moon'  
0 row(s) in 0.0250 seconds  
  
hbase(main):030:0> put 'discografia:pink_floyd', '1973#1', 'ranking:EEUU', 1  
0 row(s) in 0.0210 seconds  
  
hbase(main):031:0> put 'discografia:pink_floyd', '1973#1', 'ranking:UK', 1  
0 row(s) in 0.0330 seconds  
  
hbase(main):032:0> put 'discografia:pink_floyd', '1975#1', 'nombre_disco', 'Wish You Were Here'  
0 row(s) in 0.0350 seconds  
  
hbase(main):033:0> put 'discografia:pink_floyd', '1975#1', 'ranking:EEUU', 1  
0 row(s) in 0.0160 seconds  
  
hbase(main):034:0> put 'discografia:pink_floyd', '1975#1', 'ranking:UK', 1  
0 row(s) in 0.0220 seconds  
  
hbase(main):035:0> put 'discografia:pink_floyd', '1977#1', 'nombre_disco', 'Animals'  
0 row(s) in 0.0230 seconds  
  
hbase(main):036:0> put 'discografia:pink_floyd', '1977#1', 'ranking:EEUU', 3  
0 row(s) in 0.0170 seconds  
  
hbase(main):037:0> put 'discografia:pink_floyd', '1977#1', 'ranking:UK', 2  
0 row(s) in 0.0400 seconds
```

```

hbase(main):038:0> put 'discografia:pink_floyd', '1979#1', 'nombre_disco', 'The Wall'
0 row(s) in 0.0390 seconds

hbase(main):039:0> put 'discografia:pink_floyd', '1979#1', 'ranking:EEUU', 1
0 row(s) in 0.0240 seconds

hbase(main):040:0> put 'discografia:pink_floyd', '1979#1', 'ranking:UK', 3
0 row(s) in 0.0160 seconds

hbase(main):041:0> put 'discografia:pink_floyd', '1983#1', 'nombre_disco', 'The Final Cut'
0 row(s) in 0.0200 seconds

hbase(main):042:0> put 'discografia:pink_floyd', '1983#1', 'ranking:EEUU', 6
0 row(s) in 0.0170 seconds

hbase(main):043:0> put 'discografia:pink_floyd', '1983#1', 'ranking:UK', 1
0 row(s) in 0.0200 seconds

hbase(main):044:0> put 'discografia:pink_floyd', '1987#1', 'nombre_disco', 'A Momentary Lapse of Reason'
0 row(s) in 0.0110 seconds

hbase(main):045:0> put 'discografia:pink_floyd', '1987#1', 'ranking:EEUU', 3
0 row(s) in 0.0490 seconds

hbase(main):046:0> put 'discografia:pink_floyd', '1987#1', 'ranking:UK', 3
0 row(s) in 0.0160 seconds

```

```

hbase(main):047:0> put 'discografia:pink_floyd', '1994#1', 'nombre_disco', 'The Division Bell'
0 row(s) in 0.0230 seconds

hbase(main):048:0> put 'discografia:pink_floyd', '1994#1', 'ranking:EEUU', 1
0 row(s) in 0.0130 seconds

hbase(main):049:0> put 'discografia:pink_floyd', '1994#1', 'ranking:UK', 1
0 row(s) in 0.0460 seconds

hbase(main):050:0> put 'discografia:pink_floyd', '2014#1', 'nombre_disco', 'The Endless River'
0 row(s) in 0.0690 seconds

hbase(main):051:0> put 'discografia:pink_floyd', '2014#1', 'ranking:EEUU', 3
0 row(s) in 0.0530 seconds

hbase(main):052:0> put 'discografia:pink_floyd', '2014#1', 'ranking:UK', 1
0 row(s) in 0.1760 seconds

```

Consultar la información de la tabla:

scan 'discografia:pink_floyd'

```

hbase(main):054:0> scan 'discografia:pink_floyd'
ROW                                     COLUMN+CELL
1967#1                                   column=nombre_disco:, timestamp=1457859758629, value=The Piper at the Gates of Dawn
                                         column=ranking:EEUU, timestamp=145785983450, value=131
1967#1                                   column=ranking:UK, timestamp=1457859802306, value=6
1968#1                                   column=nombre_disco:, timestamp=1457860542608, value=A Saucerful of Secrets
                                         column=ranking:UK, timestamp=1457860555855, value=9
1969#1                                   column=nombre_disco:, timestamp=1457860566245, value=Music from the Film More
                                         column=ranking:EEUU, timestamp=1457860566416, value=153
1969#1                                   column=ranking:UK, timestamp=1457860574014, value=9
1969#2                                   column=nombre_disco:, timestamp=1457860589942, value=Ummagumma (Disco 2)
                                         column=ranking:EEUU, timestamp=1457860590138, value=74
1969#2                                   column=ranking:UK, timestamp=1457860592567, value=5
1970#1                                   column=nombre_disco:, timestamp=1457860659072, value=Atom Heart Mother
                                         column=ranking:EEUU, timestamp=1457860659248, value=55
1970#1                                   column=ranking:UK, timestamp=1457860660796, value=1
1971#1                                   column=nombre_disco:, timestamp=1457860673111, value=Meddle
                                         column=ranking:EEUU, timestamp=1457860673475, value=70
1971#1                                   column=ranking:UK, timestamp=1457860676259, value=3
1972#1                                   column=nombre_disco:, timestamp=1457860706591, value=Obscured by Clouds
                                         column=ranking:EEUU, timestamp=1457860706817, value=46
1972#1                                   column=ranking:UK, timestamp=1457860706966, value=6
1973#1                                   column=nombre_disco:, timestamp=1457860707145, value=The Dark Side of the Moon
                                         column=ranking:EEUU, timestamp=1457860707293, value=1
1973#1                                   column=ranking:UK, timestamp=1457860707505, value=1
1975#1                                   column=nombre_disco:, timestamp=1457860707665, value=Wish You Were Here
                                         column=ranking:EEUU, timestamp=1457860707832, value=1
1975#1                                   column=ranking:UK, timestamp=1457860707990, value=1

```

```
1977#1          column=nombre_disco:, timestamp=1457860708128, value=Animals
1977#1          column=ranking:EEUU, timestamp=1457860708287, value=3
1977#1          column=ranking:UK, timestamp=1457860708459, value=2
1979#1          column=nombre_disco:, timestamp=1457860708682, value=The Wall
1979#1          column=ranking:EEUU, timestamp=1457860708858, value=1
1979#1          column=ranking:UK, timestamp=1457860708984, value=3
1983#1          column=nombre_disco:, timestamp=1457860709217, value=The Final Cut
1983#1          column=ranking:EEUU, timestamp=1457860709359, value=6
1983#1          column=ranking:UK, timestamp=1457860709500, value=1
1987#1          column=nombre_disco:, timestamp=1457860709639, value=A Momentary Lapse of Reason
1987#1          column=ranking:EEUU, timestamp=1457860709788, value=3
1987#1          column=ranking:UK, timestamp=1457860709924, value=3
1994#1          column=nombre_disco:, timestamp=1457860710053, value=The Division Bell
1994#1          column=ranking:EEUU, timestamp=1457860710179, value=1
1994#1          column=ranking:UK, timestamp=1457860710367, value=1
2014#1          column=nombre_disco:, timestamp=1457860710880, value=The Endless River
2014#1          column=ranking:EEUU, timestamp=1457860711457, value=3
2014#1          column=ranking:UK, timestamp=1457860713244, value=1
15 row(s) in 1.1660 seconds
```

4

4. Escribir la instrucción y el resultado de consultar el álbum del año 1968:

Si queremos obtener toda la información del año 1968:

```
get 'discografia:pink_floyd','1968#1'
```

```
hbase(main):061:0> get 'discografia:pink_floyd','1968#1'
COLUMN           CELL
  nombre_disco:   timestamp=1457860542608, value=A Saucerful of Secrets
  ranking:UK      timestamp=1457860555855, value=9
2 row(s) in 0.1430 seconds
```

Obtenemos el nombre del disco '**A Saucerful of Secrets**' y el ranking de Reino Unido **9**

Si solo queremos obtener el título del álbum de 1968:

```
get 'discografia:pink_floyd','1968#1', 'nombre_disco'
```

```
hbase(main):060:0> get 'discografia:pink_floyd','1968#1', 'nombre_disco'
COLUMN           CELL
  nombre_disco:   timestamp=1457860542608, value=A Saucerful of Secrets
1 row(s) in 0.0530 seconds
```

El resultado es '**A Saucerful of Secrets**'.

Recursos

Documentos



Anexo I

[UD2_anexol.pdf](#)

Arquitectura



3.4.3 SAMPLE

[archivo_ejemplo.txt](#)

3.4.3 SAMPLE



Anexo II

[UD2_anexoll.pdf](#)

Configuración de un Clúster en Hadoop

Enlaces de Interés



<http://hbase.apache.org/book/book.html>

NoSQL Databases



<http://www.project-voldemort.com/voldemort/quickstart.html>

NoSQL Databases



<http://docs.basho.com/riak/latest/>

NoSQL Databases



<http://zookeeper.apache.org/doc/r3.3.4/index.html>

NoSQL Databases



<http://hortonworks.com/get-started/>

Mapreduce



<https://hive.apache.org/>

Mapreduce



<http://pig.apache.org/>

Mapreduce



<http://doc.mapr.com/display/MapR/Home>

Mapreduce



<http://docs.aws.amazon.com/AmazonS3/latest/gsg/GetStartedWithS3.html>

Almacenamiento



<http://wiki.apache.org/hadoop/GettingStartedWithHadoop>

Almacenamiento

Bibliografía

- ***Big Data Glossary.*** : Warden, Pete. Sebastopol, CA: O'Reilly Media; 2011.
- ***Big Data Now: Current Perspectives from O'Reilly Media.*** : O'Reilly Radar. Sebastopol, CA: O'Reilly Media; 2016.
- ***Cassandra: The Definitive Guide.*** : Hewitt, Eben. Sebastopol, CA: O'Reilly Media; 2010.
- ***Hadoop in Action.*** : Lam, Chuck. Shelter Island, NY: Manning; 2010.
- ***Hadoop: The Definitive Guide.*** : White, Tom. Sebastopol, CA: O'Reilly Media; 2012.
- ***HBase: The Definitive Guide.*** : George, Lars. Sebastopol, CA: O'Reilly Media; 2011
- ***Pro Hadoop.*** : Venner, Jason. Berkeley, CA: Apress; 2009.
- ***Professional NoSQL.*** : Tiwari, Shashank. Hoboken, NJ: Wrox; 2011.
- ***Redis Cookbook.*** : Macedo, Tiago y Oliveira, Fred. Sebastopol, CA: O'Reilly Media; 2011.
- **Bibliografía complementaria. Management Strategies for the Cloud Revolution.** : Babcock, Charles. Nueva York: McGraw Hill; 2010.
- **Bibliografía complementaria. Del cloud computing al big data.** : Torres i Viñals, Jordi. Barcelona: Universitat Oberta de Catalunya; 2012.
- **Bibliografía complementaria. Linux Pocket Guide.** : Barret, Daniel J. Sebastopol, CA: O'Reilly Media; 2012.
- **Bibliografía complementaria. Programming Amazon EC2.** : Van Vliet, Jurg y Paganelli, Flavia. Sebastopol, CA: O'Reilly Media; 2011.
- **Bibliografía complementaria. Practical Guide to Linux Commands, Editors, and Shell Programming.** : Sobell, Mark G. A. Nueva Jersey: Prentice Hall; 2012.
- **Bibliografía complementaria. Amazon Web Services 100 Success Secrets.** : Robinson, Donald. Sebastopol, CA: O'Reilly Media; 2008.
- **Bibliografía complementaria. Programación en Java 2.** : Joyanes, Luis y Zahonero, Ignacio. Madrid: McGraw Hill; 2002.
- **Bibliografía complementaria. Handbook of Cloud Computing.** : Furht, Borko y Escalante, Armando. Nueva York: Springer; 2010.

Glosario.

- **DAG:** En ciencias de la computación y matemáticas, un grafo acíclico dirigido o DAG (del inglés Directed Acyclic Graph) es un grafo dirigido que no tiene ciclos; esto significa que para cada vértice v no hay un camino directo que empiece y termine en v.
- **Nivel de log:** En IT, nivel de avisos que se escriben en un fichero log (fatal, error, warning, info, etc.).
- **Shell:** En informática, el shell (intérprete de órdenes o intérprete de comandos) es el programa informático que provee una interfaz de usuario para acceder a los servicios del sistema operativo.
- **Sistema relacional:** El modelo relacional, para el modelado y la gestión de bases de datos, es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. En este modelo todos los datos se almacenan en relaciones y, como cada relación es un conjunto de datos, el orden en el que estos se almacenen no tiene relevancia (a diferencia de otros modelos como el jerárquico y el de red).