

Configuración e execución de traballos mapreduce con Hive en Hadoop

Índice

Introdución.....	2
Instalación e configuración de Apache Hive.....	3
Descarga e descompresión.....	3
Configuración de variables de entorno.....	4
Creación de directorios para Hive sobre HDFS.....	5
Probámolo?!!! hai que lembrarse deste cambio, ollo!.....	5
Revisa-lo ficheiro de configuración hive-site.xml.....	6
Inicia-la base de datos dos metadatos.....	9
Hive CLI.....	11
Comandos básicos da shell interactiva de hive.....	13
Tipos de datos.....	14
Bases de datos.....	16
Creación de base de datos.....	16
Táboas de Hive.....	18
Táboas internas e externas de Hive.....	23
Partitions e buckets das táboas Hive.....	28
Partitions.....	30
Buckets.....	32
Beeline CLI.....	34
Beeline en modo embebido.....	35
Beeline para Conectar a Hive en modo remoto.....	37
Introdución ó uso de Apache Hive.....	42
Exemplo 1. Creación e uso de bases de datos.....	42
Exemplo 2. Creación e carga de táboas.....	43
Exemplo 3. Consultas HiveQL SELECT WHERE.....	45
O problema das subconsultas en Hive.....	46
Operadores en Hive.....	49
Funcións en Hive.....	52
Exemplo 4. Consultas HiveQL SELECT ORDER BY.....	54
Exemplo 5. Consultas HiveQL SELECT GROUP BY.....	55
Exemplo 6. Consultas HiveQL SELECT JOIN.....	56
Exemplo 7. Saída/Output de Hive.....	60
Exercicio.....	61
EXEMPLO 8. Inserción de datos en táboas.....	62
Exercicio de aplicación de consultas Hive.....	65
EXEMPLO 9. Modificar datos en táboas.....	66

Introducción



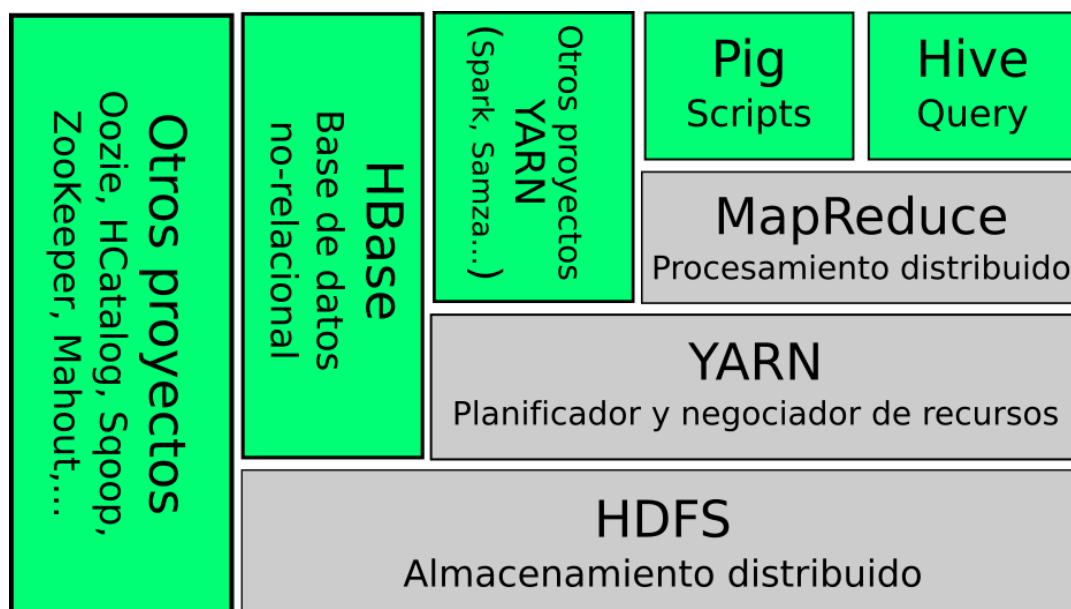
Para executar Apache Hive é **necesaria unha instalación de Hadoop**.

Hive traballa a un nivel superior a MapReduce, proporcionando unha linguaxe moi similar a SQL, que se traduce en execución a traballos mapreduce que executará Hadoop.

Funciona ó longo de todo o clúster grazas a Yarn.

Instalarase nun clúster Hadoop con hdfs e Yarn e o resultado de Hive tradúcese a MapReduce para repartirse a execución a través do clúster. Máis recentemente, Hive pódese executar usando DAG, en vez de MapReduce, mediante o uso dos frameworks Tez ou DAG-Spark, en vez de Yarn.

Con Yarn, o resultado de Hive tradúcese a MapReduce e repártese a execución a través do clúster. Non é necesario **instalar Hive máis que nun único nodo**, polo xeral será o nodo master ou aquel que teña o **ResourceManager**.



Hive 3 deixa de soportar MapReduce. Apache Tez reemprázao como o motor de execución por defecto, de xeito que mellora o rendemento e se executa sobre Hadoop Yarn, que encola e planifica os traballos no clúster. Ademais de Tez, Hive tamén pode utilizar Apache Spark como motor de execución.

Instalación e configuración de Apache Hive

Na web oficial de Apache atópase documentación sobre Hive e os lugares para a súa descarga:

- <https://cwiki.apache.org/confluence/display/HIVE>
<https://cwiki.apache.org/confluence/display/Hive/GettingStarted#GettingStarted-InstallationandConfiguration>
- <https://www.apache.org/dyn/closer.cgi/hive/>

Partiremos do escenario Hadoop con hdfs e Yarn que anteriormente montamos con alomenos 3 máquinas virtuais:

- Nodo Master: Ubuntu Desktop
- 2 x Nodo Worker: Ubuntu Server

Realizaremos esta práctica sobre o **nodo master**.

Descarga e descompresión

- Descarga-la última versión estable de Hive: <https://dlcdn.apache.org/hive/>



```
# Descarga da versión recente e estable do programa hive dende un mirror de Apache
```

```
$ wget https://dlcdn.apache.org/hive/stable-2/apache-hive-2.3.9-bin.tar.gz
```

- Descomprimi-lo ficheiro e movelo a

```
# Descomprimi-lo programa hive
```

```
$ tar -xvzf apache-hive-2.3.9-bin.tar.gz
```

```
# Move-lo programa hive
```

```
$ sudo mv apache-hive-2.3.9-bin /usr/share/hive
```

```
apache-hive-2.3.9-bin/hcatalog/sbin/update-hcatalog-env.sh
apache-hive-2.3.9-bin/hcatalog/sbin/webhcat_config.sh
apache-hive-2.3.9-bin/hcatalog/sbin/webhcat_server.sh
hduser@hadoop-master:~/Descargas$ ls
apache-hive-2.3.9-bin  hadoop-3.3.1.tar.gz  KEYS  pig-0.17.0.tar.gz
apache-hive-2.3.9-bin.tar.gz  hadoop-3.3.1.tar.gz.asc  movies-noheader.csv  ratings-noheader.csv
AverageRating.pig  hadoop-3.3.1.tar.gz.sha512  orders.tsv  returns.tsv
hduser@hadoop-master:~/Descargas$ sudo mv apache-hive-2.3.9-bin /usr/share/hive
[sudo] Contraseña de hduser:
hduser@hadoop-master:~/Descargas$ ls /usr/share/hive/
bin  binary-package-licenses  conf  examples  hcatalog  jdbc  lib  LICENSE  NOTICE  RELEASE_NOTES.txt  scripts
hduser@hadoop-master:~/Descargas$
```

Configuración de variables de entorno

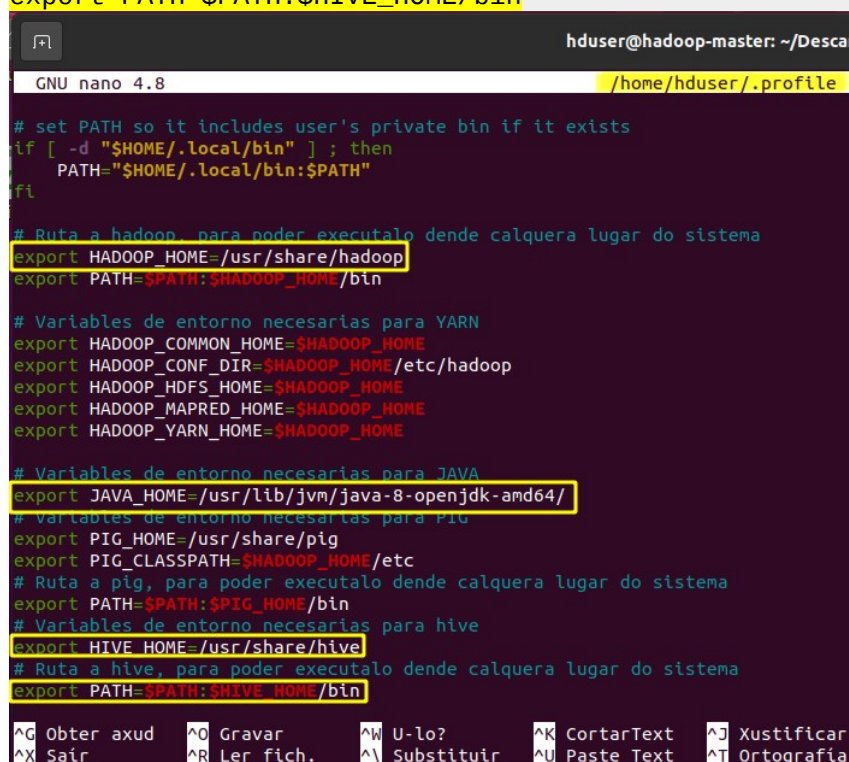
- Engadi-las seguintes variables de contorno no `co comando` :

```
$ nano ~/.profile
```

```
# Variables de entorno necesarias para HADOOP (hive usa hadoop)
export HADOOP_HOME=/usr/share/hadoop
# Variables de entorno necesarias para JAVA
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/
# Variables de entorno necesarias para hive
export HIVE_HOME=/usr/share/hive
```

- Extra: pódese engadi-lo directorio `de hive ó` para poder executar hive dende calquera lugar do sistema:

```
# Ruta a hive, para poder executalo dende calquera lugar do sistema
export PATH=$PATH:$HIVE_HOME/bin
```



```
GNU nano 4.8 /home/hduser/.profile

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi

# Ruta a hadoop, para poder executalo dende calquera lugar do sistema
export HADOOP_HOME=/usr/share/hadoop
export PATH=$PATH:$HADOOP_HOME/bin

# Variables de entorno necesarias para YARN
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME

# Variables de entorno necesarias para JAVA
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/
# Variables de entorno necesarias para PIG
export PIG_HOME=/usr/share/pig
export PIG_CLASSPATH=$HADOOP_HOME/etc
# Ruta a pig, para poder executalo dende calquera lugar do sistema
export PATH=$PATH:$PIG_HOME/bin
# Variables de entorno necesarias para hive
export HIVE_HOME=/usr/share/hive
# Ruta a hive, para poder executalo dende calquera lugar do sistema
export PATH=$PATH:$HIVE_HOME/bin

^G Obter axuda  ^O Gravar      ^W U-lo?       ^K CortarText  ^J Xustificar
^X Sair         ^R Ler fich.   ^_ Substituir  ^U Paste Text  ^T Ortografía
```

JAVA_HOME indica onde se atopa Java (no caso de hive non funcionan as versións posteriores á versión 8).

HIVE_HOME é a variable de entorno que apunta ó cartafol no que está instalado Apache Hive.

PATH é a variable de entorno á que se lle engade o cartafol `de Hive`, para poder executar dende calquera lugar do sistema.

Pódese forzar que se carguen as novas variables de entorno saíndo da sesión e volvendo a entrar nela, ou executando manualmente o script `da conta` .

```
$ source ~/.profile
```

Creación de directorios para Hive sobre HDFS

Hai que crear, no sistema HDFS, dous directorios separados para almacena-los datos. Polo tanto, previamente hai que arranca-lo sistema dfs:

```
$ /usr/share/hadoop/sbin/start-dfs.sh
```

- Crear un directorio provisional, `/tmp/hive`, para almacena-los resultados intermedios dos procesos de Hive, e dáselle permiso de escritura e execución sobre hdfs ó supergroup e a calquera.

```
$ hdfs dfs -mkdir -p /tmp/hive
$ hdfs dfs -chmod 777 /tmp/hive
```

- Crear un directorio `/user/hduser/warehouse`, que será o utilizado como almacén das táboas usado por Hive, e dáselle permiso de escritura e execución sobre hdfs ó supergroup e a calquera.

```
$ hdfs dfs -mkdir -p /user/hduser/warehouse
$ hdfs dfs -chmod 777 /user/hduser/warehouse
```

```
hduser@hadoop-master:/usr/share/hive/conf$ hdfs dfs -ls -d /tmp/hive
drwxrwxrwx - hduser supergroup          0 2022-03-14 19:09 /tmp/hive
hduser@hadoop-master:/usr/share/hive/conf$ hdfs dfs -ls -d /user/hduser/warehouse
drwxrwxrwx - hduser supergroup          0 2022-03-14 19:10 /user/hduser/warehouse
hduser@hadoop-master:/usr/share/hive/conf$
```

Este directorio será o que conterá os datos xestionados a través de Hive, así que se cadra é máis razoable crealo en `/user/hive` en vez de en `/user/hduser`:

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

Prob molo! """" hai #ue lembrarse deste cambio\$ ollo"

%evisa&lo ficheiro de configuración hive&site'xml

As distribucións de Apache Hive conteñen uns modelos ou plantillas de ficheiros de configuración por defecto. Estes ficheiros modelo están no directorio de Hive e teñen a extensión .

```
hduser@hadoop-master:~$ cd $HIVE_HOME/conf
hduser@hadoop-master:/usr/share/hive/conf$ ls
beeline-log4j2.properties.template  hive-exec-log4j2.properties.template  llap-cli-log4j2.properties.template
hive-default.xml.template           hive-log4j2.properties.template       llap-daemon-log4j2.properties.template
hive-env.sh.template               ivysettings.xml                      parquet-logging.properties
hduser@hadoop-master:/usr/share/hive/conf$
```

O ficheiro controla cada aspecto das operacións de Hive, e o modelo de exemplo, , contén unha cantidade de parámetros de configuración enorme (pódense consultar na documentación de configuración do sitio oficial Hive, en:

<https://cwiki.apache.org/confluence/display/Hive/AdminManual+Configuration#AdminManualConfiguration-HiveConfigurationVariables>).

No noso caso usaremos un ficheiro moito máis sinxelo, polo que creade o ficheiro de configuración en , como unha copia do ficheiro que se achega.

- Crear ficheiro

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:derby:/usr/share/hive/metastore_db;create=true</value>
    <description>Configuración do Metastore de Hive. Utilízase un ficheiro local usando a base de datos integrada Apache Derby</description>
  </property>
  <property>
    <name>hive.execution.engine</name>
    <value>mr</value>
    <description>Configura Hive para utilizar o motor de execución MapReduce</description>
  </property>
  <property>
    <name>hive.server2.enable.doAs</name>
    <value>>false</value>
    <description>Para simplificar a configuración, realiza todas as consultas como a conta de usuario/a coa que se executa hive (no noso sistema, hduser) en lugar de considerar usuarios/as individuais</description>
  </property>
  <property>
    <name>hive.server2.active.passive.ha.enable</name>
    <value>true</value>
    <description>Activa a configuración de alta dispoñibilidade para Thrift. O cliente beeline necesita esta configuración para poder conectarse con Hive.</description>
  </property>
  <property>
    <name>hive.server2.thrift.port</name>
    <value>10000</value>
    <description>Porto no que escoita o servidor Thrift</description>
  </property>
</configuration>
```

```

hduser@hadoop-master: /usr/share/hive
GNU nano 4.8 /usr/share/hive/conf/hive-site.xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>javax.ido.option.ConnectionURL</name>
    <value>jdbc:derby:/usr/share/hive/metastore_db;create=true</value>
    <description>Configuracion do Metastore de Hive. Utilizase un ficheiro local usando a base de datos integrada Apache Derby</description>
  </property>
  <property>
    <name>hive.execution.engine</name>
    <value>mr</value>
    <description>Configura Hive para utilizar o motor de execucion MapReduce</description>
  </property>
  <property>
    <name>hive.server2.enable.doAs</name>
    <value>>false</value>
    <description>Para simplifica-la configuracion, realiza todas as consultas como a conta de usuario/a coa que se executa hive en lugar de con
  </property>
  <property>
    <name>hive.server2.active.passive.ha.enable</name>
    <value>true</value>
    <description>Activa a configuracion de alta dispoñibilidade para Thrift. O cliente beeline necesita esta configuracion para poder conectars
  </property>
  <property>
    <name>hive.server2.thrift.port</name>
    <value>10000</value>
    <description>Porto no que escoita o servidor Thrift</description>
  </property>
</configuration>

```

\$sing Hive in a stand%alone mode rat&er t&an in a real%life 'pac&e Hadoop cluster is a safe option for ne (comers) *ou can configure t&e s+stem to use +our local storage rat&er t&an t&e H , - . la+er b+ setting t&e &ive)metastore) (are&ouse)dir parameter value to t&e location of +our Hive (are&ouse director+)/

Nota: ----->

Hai máis plantillas de ficheiros de configuración no directorio , do directorio de instalación de hive, como, por exemplo a do rexistro de logs en

A característica común de todas elas é que para crea-los correspondentes ficheiros de configuración a partir delas basta con copialas co nome adecuado:

```

$ cd /usr/share/hive/conf/
$ cp hive-default.xml.template hive-site.xml
$ cp hive-env.sh.template hive-env.sh
$ cp hive-exec-log4j2.properties.template hive-exec-log4j2.properties
$ cp hive-log4j2.properties.template hive-log4j2.properties

```

O ficheiro serve para incluír variables de entorno como as rutas de Hadoop e da configuración de Hive:

```

export HADOOP_HOME=/usr/share/hadoop
export HIVE_CONF_DIR=/usr/share/hive/conf
export HCAT_HOME=$HIVE_HOME/hcatalog

```

Tamén é posible configurar variables de entorno, como as do pat& para exporta-las rutas de Hive e do metastore:

ou

```
export HIVE_HOME=/usr/share/hive
export HIVE_CONF_DIR=$HIVE_HOME/conf
export HCAT_HOME=$HIVE_HOME/hcatalog
```

<-----

-----> A probar por min para usar MariaDB como metastore NON FACER POLO DE AGORA!!!

Para o metastore poderíamos usar un servidor de MariaDB correndo no propio hadoop-master, ou noutra máquina, e así permiti-la conexión de máis dunha sesión de forma simultánea

Metastore en MariaDB

Para crea-lo metastore en MariaDB:

```
$ /usr/share/hive/bin/schematool -dbType mysql -initSchema
```

```
<!-- novas propiedades -->
```

```
<property>
  <name>system:java.io.tmpdir</name>
  <value>/tmp/hive/java</value>
</property>
<property>
  <name>system:user.name</name>
  <value>${user.name}</value>
</property>
<property>
  <name>datanucleus.schema.autoCreateAll</name>
  <value>true</value>
</property>
```

```
<!-- propiedades existentes a modificar -->
```

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExist=true</value>
</property>
<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>com.mysql.jdbc.Driver</value>
</property>
<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>hduser</value>
</property>
<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>abc123.</value>
</property>
```

<-----

Inicia a base de datos dos metadatos

Apache Hive trae integrado un motor de base de datos chamado `DerbyDB` para almacena-los metadatos. Para contornos máis complexos (multiusuario) é posible usar outros sistemas como MySQL ou PostgreSQL, pero polo de agora só necesitamos unha única conexión para probar, así que manteremos DerbyDB como almacén xa que non require dunha instalación extra.

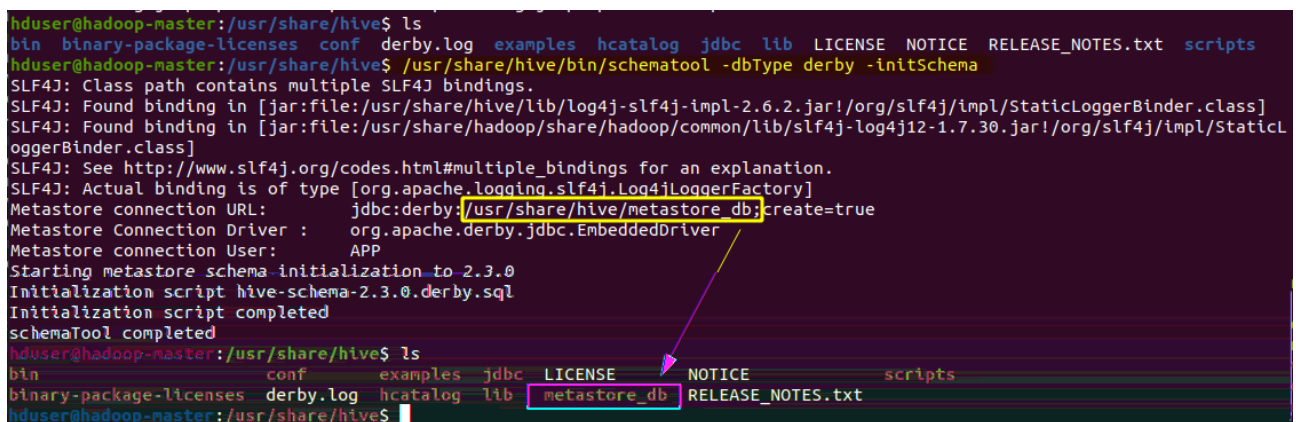
Observa a seguinte propiedade do ficheiro `hive-site.xml`, nela indícase o lugar no que se atopa o metastore de Derby para Hive.

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:derby:/usr/share/hive/metastore_db;create=true</value>
  <description>Configuración do Metastore de Hive. Utilízase un ficheiro local
usando a base de datos integrada Apache Derby</description>
</property>
```

Nota: inicializa-lo schema é un paso previo imprescindible para poder executar `hive` e `hivecli`.

- O seguinte comando `hive --initSchema` crea a base de datos no directorio actual, polo que debes situarte primeiro na ruta anteriormente especificada no ficheiro `hive-site.xml`.

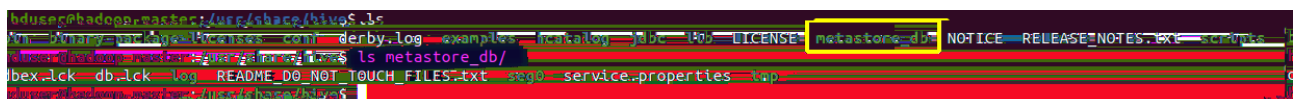
```
$ cd /usr/share/hive/
$ /usr/share/hive/bin/schematool -dbType derby -initSchema
```



The terminal screenshot shows the following output:

```
hduser@hadoop-master: /usr/share/hive$ ls
bin binary-package-licenses conf derby.log examples hcatalog jdbc lib LICENSE NOTICE RELEASE_NOTES.txt scripts
hduser@hadoop-master: /usr/share/hive$ /usr/share/hive/bin/schematool -dbType derby -initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/share/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/share/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.30.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:/usr/share/hive/metastore_db;create=true
Metastore Connection Driver : org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User: APP
Starting metastore schema initialization to 2.3.0
Initialization script hive-schema-2.3.0.derby.sql
Initialization script completed
schematool completed
hduser@hadoop-master: /usr/share/hive$ ls
bin conf examples jdbc LICENSE NOTICE scripts
binary-package-licenses derby.log hcatalog lib metastore_db RELEASE_NOTES.txt
hduser@hadoop-master: /usr/share/hive$
```

Tras isto, pódese comprobar que se creou un novo directorio `metastore_db` no directorio activo.



The terminal screenshot shows the following output:

```
hduser@hadoop-master: /usr/share/hive$ ls
bin binary-package-licenses conf derby.log examples hcatalog jdbc lib LICENSE metastore_db NOTICE RELEASE_NOTES.txt scripts
hduser@hadoop-master: /usr/share/hive$
```

E, se hdfs e yarn están correndo, xa se pode arranca-la consola de **Hive (CLI)** invocando o comando `hive` coa ruta completa da súa ubicación:

```
$ /usr/share/hive/bin/hive
```

Se anteriormente se cargaron as variables de entorno (saíndo da sesión e volvendo a entrar nela, ou executando manualmente o script da conta con source 2/))profile) pódese invoca-lo comando hive dende calquera ruta.

```
$ hive
```

```
hduser@hadoop-master: /usr/share/hive$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/share/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/share/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.30.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:/usr/share/hive/lib/hive-common-2.3.9.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> █
```

Pódese saír da consola con .

```
hive> quit;
```

Hive C(l)

Hive ven con Hive 345, un cliente baseado no protocolo de comunicación 'pac&e 6&rift, que usa os mesmos drivers que Hive, pero tamén incorpora Zeeline, un cliente baseado en JDBC para facer consultas en liña de comandos contra o compoñente Hive.erver, sen necesita-las dependenzas de Hive.

A consola Hive CLI está desfasada, en favor do uso de Hive .erver 8 Zeeline como cliente, pero aínda podemos utilizala para verifica-lo funcionamento de Hive.

```
hduser@hadoop-master: /usr/share/hive$ hive --version
Hive 2.3.9
Glt glt://chaos-nbp.lan/Users/chaoglt/hive -r 92dd0159f440ca7863be3232f3a683a510a62b9d
Compiled by chao on Tue Jun 1 14:02:14 PDT 2021
From source with checksum 0715a3ba850b746eebfbbec20d5a0187
hduser@hadoop-master: /usr/share/hive$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/share/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/share/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.30.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:/usr/share/hive/lib/hive-common-2.3.9.jar!/hive-log4j2.properties Async: true
Hive-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 3.X releases.
hive> show databases;
OK
default
Time taken: 4.104 seconds, Fetched: 1 row(s)
hive> quit;
hduser@hadoop-master: /usr/share/hive$
```

Lembrems que Hive execútase sobre hdfs e MapReduce, así que antes de lanzar hive, hai que segui-los seguintes pasos cando se inicia a máquina.

O primeiro que hay que facer é conectarse coa conta , iniciando sesión coa mesma ou executando o seguinte comando:

```
$ su - hduser
```

Logo xa se pode arranca-lo sistema dfs e os demos #arn:

```
$ /usr/share/hadoop/sbin/start-dfs.sh
$ /usr/share/hadoop/sbin/start-yarn.sh
```

E agora xa se pode entrar na consola de Hive co seguinte comando:

```
$ /usr/share/hive/bin/hive
```

Aparecerá o prompt de hive, á espera de ordes:

```
hive>
```

Para probar que todo funciona correctamente podemos executa-la seguinte sentencia HiveQL (os comandos no CLI deben rematar con ; (punto e coma).

```
hive> SHOW DATABASES;
```

Isto fai saia por pantalla a lista de bases de datos existentes en Hive, na que inicialmente só hai unha base de datos por defecto, a “default”:

```
OK
default
Time taken: 4.104 seconds, Fetched: 1 row(s)
```

Para saír da consola, basta con executa-la sentenza `quit` ou `exit`.

```
hive> quit;
```

Dende a liña de comandos do sistema, sen entrar na shell de hive, tamén é posible executar sentenzas hive usando o comando `hive` coa opción `-e`; por exemplo, para obte-los datos da táboa clientes da base de datos negocio faríase:

```
$ $HIVE_HOME/bin/hive -e 'select * from negocio.clientes'
```

- Exemplo de execución dunha consulta dende liña de comandos:

```
$ $HIVE_HOME/bin/hive -e 'select a.col from tab1 a'
```

- Exemplo de obtención nun ficheiro de texto, do resultado dunha consulta usando o modo silencioso:

```
$ $HIVE_HOME/bin/hive -S -e 'select a.col from tab1 a' > a.txt
```

Para executar un script, bastaría con usa-lo propio comando `hive` coa opción `-f`:

```
$ $HIVE_HOME/bin/hive -f script.sql
```

- Exemplo de execución non interactiva, dun script tomado do disco local:

```
$ $HIVE_HOME/bin/hive -f /home/my/hive-script.sql
```

- Exemplo de execución non interactiva, dun script tomado do sistema dfs:

```
$ $HIVE_HOME/bin/hive -f hdfs://<namenode>:<port>/hive-script.sql
```

Comandos básicos da shell interactiva de hive

Os comandos de hive deben rematar con (punto e coma) ó final.

Os comentarios, nos scripts, precédense de 2 guións ó comezo da liña, por exemplo:

```
-- Isto é un comentario nun script Hive
```

- Comandos **quit** e **exit**.

Saír da liña de comandos.

A súa sintaxe é a seguinte:

```
quit  
exit
```

Por exemplo:

```
hive> quit;
```

- Comando **!**

Permite executar comando da shell de Linux

A súa sintaxe é a seguinte:

```
!<comando linux>
```

Por exemplo:

```
hive> !pwd;
```

- Comando **dfs**

Permite executar comandos dfs

A súa sintaxe é a seguinte:

```
dfs <comando dfs>
```

Por exemplo:

```
hive> dfs -ls /;
```

- Comando **source**

Permite executar un script HiveQL dende dentro do CLI.

A súa sintaxe é a seguinte:

```
source <ruta-e-nome-do-script>
```

Por exemplo:

```
hive> source /test/scripts/calcula_vendas.hql;
```

Tipos de datos

Similares ós utilizados pola DDL de SQL.

1. Numéricos enteiros de diferentes tamaños, que se especifican mediante un carácter de sufixo.

- TINYINT (1 byte, de -128 a 127):
- SMALLINT (2 bytes, de -32.768 a 32.767):
- INTEGER (4 bytes, de -2.147.483.648 to 2.147.483.647):
- BIGINT (8 bytes, de -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807):

2. Numéricos reais

- FLOAT (4-byte single precision floating point number)
- DOUBLE (8-byte double precision floating point number)
- DOUBLE PRECISION (alias for DOUBLE)
- DECIMAL ou NUMERIC (inicialmente cunha precisión de 38 díxitos (de -10^{-38} a 10^{38}), pero actualmente cunha precisión e escala definibles polo/a usuario/a; p.e., indica unha precisión de 5 díxitos (incluídos os da parte fraccionaria) e 2 díxitos despois da coma decimal.

3. Cadeas de caracteres

- STRING, poden expresarse entre comiñas simples (') ou dobres ("). Hive usa a “barra inclinada inversa, backslash” (\), ó “estilo C”, para “escapar” dentro das cadeas.
- VARCHAR (de 1 a 65.355 caracteres)
- CHAR (ata 255 caracteres)

4. Datas

- TIMESTAMP, no formato tradicional Unix con precisión opcional de nanosegundos: “YYYY-MM-DD HH:MM:SS[.ffffff]”
- DATE, na forma YYYY-MM-DD.
- INTERVAL, permite definir intervalos con distintas unidades de tempo (SECOND / MINUTE / DAY / MONTH / YEAR):
(1 día de intervalo),
(1 ano + 2 meses de intervalo).

5. Miscelánea:

- BOOLEAN, ou
- BINARY, representan secuencias de bytes, é dicir, datos binarios sen formato. Este tipo de datos utilízase para almacenar datos que non están nun formato de texto lexible, como imaxes, arquivos comprimidos, datos en formato binario dalgún sistema externo etc. Hive

almacena estes datos na súa forma binaria orixinal e non realiza ningunha interpretación do seu contido. Cando se traballa con datos binarios en Hive, é importante ter en conta que non se poden realizar operacións de consulta ou manipulación directamente no contido dos datos binarios a través de SQL estándar o habitual é utilizar funcións de usuario/a personalizadas ou UDFs (User Defined Functions) para manipular e procesar estes datos binarios. Os datos binarios son útiles cando necesitas almacenar información non estruturada ou datos que non se axustan a un formato de texto lexible.

6. Tipos compostos

- ARRAY, lista que agrupa elementos do mesmo tipo, e ós que se accede mediante un índice;
= Para acceder ó elemento da 2ª posición:
- MAP, para definir parellas de chave-valor, nas que a cada elemento accédese mediante a súa chave;
= Para acceder ó contrasinal de Ricardo:
- STRUCT, son datos complexos con elementos de distintos tipos; a cada elemento accédese mediante a notación "dot":
= Para acceder á propiedade cantidade:

```
CREATE TABLE datos_complexos
(
  mascotas      ARRAY<string>,
  contrasinais   MAP<string,string>,
  bichos         STRUCT<animal:string, cantidade:integer, tipo:string>
);
```

7. NULL: o valor especial NULL pódese usar cando falte un valor.

Fonte: <https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=82706456#LanguageManualTypes-bigint>

Nota: para realizar unha conversión explícita de tipos, por exemplo dun tipo texto a un numérico, hai que utiliza-la función CAST:

```
# converte o valor da cadea "123" a tipo enteiro
SELECT CAST('123' AS INT);

# converte os valores de cadea da columna "id" da táboa, nun valor numérico enteiro
SELECT CAST(id AS INT) FROM taboaProdutos;
```

Operadores e funcións de Hive en:

[https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF#LanguageManualUDF-HiveOperatorsandUser-DefinedFunctions\(UDFs\)](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF#LanguageManualUDF-HiveOperatorsandUser-DefinedFunctions(UDFs))

*ases de datos

Hive permite definir bases de datos e táboas para analizar datos estruturados. A análise de datos estruturados requires de almacena-los datos de xeito tabular, e facer consultas para analizalos.

Hive ven cunha base de datos por defecto chamada `default`.

- Creación de base de datos

Unha base de datos en Hive é un namespace ou unha colección de táboas. A sintaxe da sentenza coa que crear bases de datos é a seguinte, puidendo usar `CREATE DATABASE` ou `CREATE SCHEMA`, indistintamente:

```
CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>
```

A cláusula opcional `[IF NOT EXISTS]`, permite garantir que a sentenza non se execute se a base de datos xa existe.

A seguinte sentenza crea unha base de datos chamada `userdb`:

```
hive> CREATE DATABASE IF NOT EXISTS userdb;
```

ou

```
hive> CREATE SCHEMA userdb;
```

- Comprobar que bases de datos existen

Para ve-la lista de bases de datos existentes úsase a seguinte sentenza:

```
SHOW DATABASES
```

```
hive> SHOW DATABASES;  
default  
userdb
```

- Usar bases de datos

Para abrir/conectar unha base de datos coa que operar úsase a seguinte sentenza:

```
USE <database name>
```

```
hive> USE userdb;  
OK  
Time taken: 0.09 seconds
```

Se non usamo-lo comando `USE`, utilizarase a base de datos `default`, que se atopa como raíz en `/user`, en HDFS.

- Borrar bases de datos

A sintaxe para borrar tódalas táboas dunha base de datos, xunto coa mesma é a seguinte, puidendo usar `DROP DATABASE` ou `DROP SCHEMA`, indistintamente:

```
DROP DATABASE|SCHEMA [IF EXISTS] database_name [RESTRICT|CASCADE]
```


A cláusula IF EXISTS comproba se existe a base de datos antes de tentar borrarla.

```
hive> DROP DATABASE IF EXISTS userdb;
```

A cláusula CASCADE forza o borrado das táboas, antes da eliminación da propia base de datos.

```
hive> DROP DATABASE IF EXISTS userdb CASCADE;
```

A cláusula RESTRICT só permite a eliminación da base de datos, se está baleira, é dicir, se non ten táboas, vistas ou calquera obxecto.

```
hive> DROP DATABASE IF EXISTS userdb RESTRICT;
```

Información sobre o DDL de Hive:

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL>

) boas de Hive

A sintaxe básica da sentenza que permite crear táboas en Hive é a seguinte:

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.] table_name
[(col_name data_type [COMMENT col_comment], ...)]
[COMMENT table_comment]
[ROW FORMAT row_format]
[FIELDS TERMINATED BY char]
[STORED AS file_format]
location '[location]'
```

A sintaxe completa pódese consultar na documentación on line de Hive:

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-CreateTable>

- Os nomes das táboas **non distinguen entre maiúsculas e minúsculas**, pero en caso de que existise xa unha táboa co mesmo nome no sistema, a sentenza de creación provocaría un erro, se non se incluíse a cláusula `IF NOT EXISTS`.
- `COMMENT`. Ó igual que para os nomes das táboas, **os nomes de columna tampouco son case insensitive**. Os tipos de columna son valores como int, c&ar, string, etc.
- `ROW FORMAT`. A ringleiras usan formatos nativos ou formatos `serialization`, e `deserialization` (`serialization`, `deserialization`). Se non se indica nada, ou se se especifica como DELIMITED, indica que el formato de **cada ringleira está delimitado cun salto de liña e ademais utilizará o formato por defecto** `serialization`, e;

```
DEFAULT_ESCAPE_CHARACTER \
DEFAULT_QUOTE_CHARACTER "
DEFAULT_SEPARATOR ,
```

- `TERMINATED BY`. Indica o carácter que se usa para separa-los valores da táboa nunha liña (é o carácter delimitador dos campos).
- `STORED AS`. Permite especifica-lo formato de almacenamento en HDFS, como `TEXT`, `SEQUENCEFILE`, etc.
- LOCATION**. Permite especifica-la ubicación do directorio HDFS do ficheiro que contén a táboa de datos (normalmente se esta é externa).

Característica SerDe de Hive: <https://cwiki.apache.org/confluence/display/Hive/SerDe>

Os Serializadores e Deserializadores (`Serializer`) en Hive son compoñentes fundamentais para a lectura e escritura de datos en formatos específicos. Hive necesita SerDes para interpreta-los datos

almacenados nun formato específico cando se len dende o almacenamento subxacente e para convertelos nun formato recoñecible cando se escriben no almacenamento.

En esencia, un SerDe é unha peza de código que manexa a serialización (conversión de datos estruturados nun formato específico nunha secuencia de bytes) e deserialización (conversión dunha secuencia de bytes en datos estruturados) dos datos. Hive utiliza SerDes para traballar con diferentes formatos de arquivos, como `Text`, `SequenceFile`, `Avro`, `Parquet`, etc.

Cando se crea unha táboa en Hive, pódese especificar un SerDe para indicar a Hive como ler e escribir os datos nesa táboa nun formato específico.

Para algúns formatos, como o `Text` ou `SequenceFile`, Hive manexa automaticamente a serialización e deserialización dos datos, polo que non é necesario especificar un SerDe explicitamente, xa que Hive ten integrado o soporte para eles. Pero noutros caso, como Avro, é necesario especifica-lo SerDe.

Por exemplo, para definir unha táboa en Hive cun SerDe de `Avro` para manexa-la serialización e deserialización de datos na táboa habería que especifica-los formatos de entrada e saída para a táboa:

```
hive> CREATE TABLE taboa (  
    column1 INT,  
    column2 STRING  
)  
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';
```

O formato de arquivo Parquet, que está altamente optimizado para consultas analíticas en Hive e outras ferramentas de big data, non precisa de que se especifique un SerDe explicitamente, o que simplifica a creación e administración de táboas en Hive cando se utiliza Parquet como formato de arquivo.

```
Hive> CREATE TABLE taboa (  
    column1 INT,  
    column2 STRING  
)  
STORED AS PARQUET;
```

Exemplo de creación dunha táboa chamada <mpregada cos seguintes campos e tipos:

id	int
nome	string
soldo	float
posto	string

Inclúese na creación un comentario descritivo e cales son os caracteres delimitadores de fin de campo e de fin de liña, así como o tipo de ficheiro.

```
COMMENT 'Detalles das persoas empregadas'
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED IN TEXT FILE
```

A sentenza completa sería a seguinte:

```
hive> CREATE TABLE IF NOT EXISTS empregada
(id INT, nome STRING, soldo FLOAT, posto STRING)
COMMENT 'Detalles das persoas empregadas'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

Dado que se inclúe a cláusula , a táboa só se crearía se non existise previamente.

```
OK
Time taken: 5.905 seconds
hive>
```

- Comprobar que táboas existen

Para ve-la lista de táboas existentes úsase a seguinte sentenza:

SHOW TABLES

```
hive> SHOW TABLES;
empregada
ok
Time taken: 1.1 seconds
hive>
```

- Ve-lo esquema dunha táboa

Para ve-los campos e tipos dunha táboa, úsase a seguinte sentenza:

DESCRIBE table_name

```
hive> DESCRIBE empregada;
OK
id                int
nome              string
soldo             float
posto             string
Time taken: 0.073 seconds, Fetched: 4 row(s)
hive>
```

- Borrar táboas

A sintaxe para borrar táboas dunha base de datos, é a mesma, independentemente de que se trate dunha táboa interna ou externa:

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

A cláusula IF EXISTS comproba se existe a táboa antes de tentar borrarla.

```
hive> DROP TABLE IF EXISTS empregada;
```

- Cambia-los atributos das táboas.

Pódense muda-los nomes de táboa e columnas, así como engadir, borrar e reemprazar columnas.

A sintaxe cambia dependendo de que atributos da táboa se modifican:

```
ALTER TABLE name RENAME TO new_name
```

```
ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])
```

```
ALTER TABLE name CHANGE column_name new_name new_type
```

```
ALTER TABLE name REPLACE COLUMNS (col_spec[, col_spec ...])
```

A cláusula `RENAME TO` úsase para renomea-la táboa.

```
hive> ALTER TABLE empregada RENAME TO persoal_empregado;
```

A cláusula `CHANGE` úsase para cambia-los nomes e tipos das columnas.

Por exemplo, a primeira sentenza cambia o nome dunha columna, e a segunda o tipo de dato que contén (os tipos a cambiar deben ser compatibles, p.e., non se pode cambiar un STRING a INT):

```
hive> ALTER TABLE persoal_empregado CHANGE posto cargo STRING;  
hive> ALTER TABLE persoal_empregado CHANGE soldo soldo DOUBLE;
```

A cláusula `ADD COLUMNS` úsase para engadir columnas á táboa.

```
hive> ALTER TABLE persoal_empregado ADD COLUMNS (departamento STRING COMMENT  
'Nome do Departamento');
```

Curiosamente non existe unha cláusula `REMOVE COLUMNS` para eliminar columnas da táboa, no seu lugar habería que usar `REPLACE COLUMNS` indicando as columnas a conservar e omitindo as que se quixesen borrar; os datos das columnas a conservar non se borran.

```
hive> ALTER TABLE persoal_empregado REPLACE COLUMNS (id INT, nome STRING, soldo  
DOUBLE); #elimina as columnas cargo e departamento
```

A cláusula `REPLACE COLUMNS` úsase para reempraza-los nomes e/ou tipos das columnas existentes por outros nomes e/ou tipos. As novos nomes e tipos de columnas reemprazan ás existentes previamente, segundo a súa orde, conservando os datos que teñen, e se se definenn menos columnas que as existentes, elimínanse as restantes.

```
hive> ALTER TABLE persoal_empregado REPLACE COLUMNS (id INT, nome_e_apelidos
STRING, soldo STRING, cargo STRING);
```

```
hive> CREATE TABLE IF NOT EXISTS empregada (id INT, nome STRING, soldo FLOAT,
posto STRING)
> COMMENT 'Detalles das persoas empregadas'
> ROW FORMAT DELIMITED
> FIELDS TERMINATED BY '\t'
> LINES TERMINATED BY '\n'
> STORED AS TEXTFILE;
```

OK

Time taken: 0.218 seconds

```
hive> DESCRIBE empregada;
```

OK

id	int
nome	string
soldo	float
posto	string

Time taken: 0.073 seconds, Fetched: 4 row(s)

```
hive> INSERT INTO empregada VALUES
```

```
(1, 'Manuela', 1000, 'Xefa'),
(2, 'Ricardo', 500, 'Peón'),
(3, 'Bárcenas', 2000, 'Contable');
```

OK

Time taken: 35.14 seconds

```
hive> SELECT * FROM empregada;
```

OK

1	Manuela	1000.0	Xefa
2	Ricardo	500.0	Peón
3	Bárcenas	2000.0	Contable

Time taken: 0.271 seconds, Fetched: 3 row(s)

```
hive> ALTER TABLE empregada REPLACE COLUMNS (id STRING, nome STRING, soldo
DOUBLE);
```

OK

Time taken: 0.207 seconds

```
hive> DESCRIBE empregada;
```

OK

id	string
nome	string
soldo	double

Time taken: 0.073 seconds, Fetched: 3 row(s)

```
hive> SELECT * FROM empregada;
```

OK

1	Manuela	1000.0
2	Ricardo	500.0
3	Bárcenas	2000.0

Time taken: 0.265 seconds, Fetched: 3 row(s)

```
hive>
```

) boas internas e externas de Hive

Hive ten unha base de datos relacional no nodo principal que utiliza para realizar un seguimento do estado. Por exemplo, o seguinte esquema de táboa almacénase na base de datos, na ruta por defecto :

```
hive> CREATE TABLE ficheiro(nome string);
```

Pero tamén poderíamos indicarlle, á nosa discreción, unha ubicación física distinta á predeterminada, usando o parámetro :

```
hive> CREATE TABLE ficheiro2(nome string) LOCATION 'hdfs:///probas/';
```

Se, por exemplo a táboa se particionase, as particións (coma o resto dos metadatos) almacénanse na base de datos (isto permítelle a Hive usar listas de particións sen ir ó sistema de arquivos e atopalas, p.e.).

- Cando se borra (DROP) unha **táboa interna**, descártanse os datos e tamén os metadatos: o ficheiro bórrase.
- En cambio, cando se borra unha **táboa externa**, só se descartan os metadatos; iso implica que Hive non toca os datos en si (o ficheiro non se borra).

As **táboas** de Hive pódense crear como **EXTERNAS** ou **INTERNAS** (tamén chamadas ou). Que se escolla un tipo ou outro afecta á forma en que se cargan, controlan e administran os datos.

- Para as táboas externas, **Hive almacena os datos na ubicación especificada durante a creación da táboa** (dentro de HDFS). **Se se eliminase a táboa externa, elimínanse os metadatos** (o esquema asociado) **da táboa, pero non os arquivos de datos.**
- Para as táboas internas, **Hive almacena tanto os datos como o esquema da táboa no seu directorio de almacén** (que pode ser simplemente unha ubicación de arquivo en HDFS). **Se se eliminase a táboa interna, eliminaranse tanto os metadatos como os datos e a propia táboa.**

Táboas externas

- A táboa externa almacena arquivos no servidor HDFS, pero as táboas non están vinculadas ó arquivo de orixe por completo.
- Se se elimina unha táboa externa, o arquivo aínda permanece no servidor HDFS.

Por exemplo, se se crea unha táboa externa chamado "taboa=test" en HIVE usando HIVEQL e se vincula a táboa ó arquivo "expediente%x)txt", logo de eliminar "taboa=test" de HIVE non se eliminará "expediente%x)txt" de HDFS.

- Os arquivos de táboas externos son accesibles para calquera persoa que teña acceso á estrutura de arquivos HDFS e, por tanto, a seguridade debe administrarse a nivel de arquivo/cartafol HDFS.

- Os metadatos mantéñense no nodo principal e, ó eliminar unha táboa externa de HIVE, só elimínanse os metadatos, non os datos/arquivos.

Táboas internas

- As táboas internas almacénanse nun directorio que por defecto é `"/user/hive/warehouse"`; pódese configurar por parte do/a administrador/a, actualizando a ubicación no arquivo de configuración `hive-site.xml`, na propiedade de configuración `hive.metastore.warehouse.dir`.
- Ó elimina-la táboa, elimínanse os metadatos e os datos do nodo principal e HDFS, respectivamente.
- A seguridade do arquivo de táboa interno contrólase unicamente a través de HIVE. A seguridade debe xestionarse dentro de HIVE, probablemente a nivel de esquema (depende da organización).

Utilizaranse táboas EXTERNAS cando:

- Os datos tamén se utilizan fóra de Hive. Por exemplo, os arquivos de datos son lidos e procesados por un programa existente que non bloquea os arquivos.
- Os datos deben permanecer na localización subxacente incluso despois dunha DROP TABLE. Isto pode aplicarse se se teñen varios esquemas (táboas ou vistas) a un só conxunto de datos ou se está alternando entre varios esquemas posibles.
- Quérese utilizar unha localización personalizada aliea ó sistema hdfs.
- Hive non debe posuír datos e configuracións de control, directorios, etc., porque tense outro programa ou proceso que fará esas cousas.
- Non se está a crear unha táboa baseada nunha táboa existente (é dicir, non se crea a partir dun SELECT).

Utilizaranse táboas INTERNAS cando:

- Os datos son temporais.
- Quérese que Hive administre por completo o ciclo de vida da táboa e os datos.

Para identifica-lo tipo de táboa en HIVE (administrada ou externa) hai que ve-la saída de:

```
DESCRIBE EXTENDED <nome de táboa>
```

No caso das táboas internas verase:

```
... tableType: MANAGED_TABLE)
```

Para táboas externa verase o seguinte:

```
... tableType: EXTERNAL_TABLE)
```

Máis información: <https://cwiki.apache.org/confluence/display/Hive/Managed+vs.+External+Tables>

Por exemplo.

- Lanzamo-lo interprete de comandos **hive** executando:

```
$ /usr/share/hive/bin/hive
```

- Creamos unha táboa de tipo externo usando o seguinte comando:

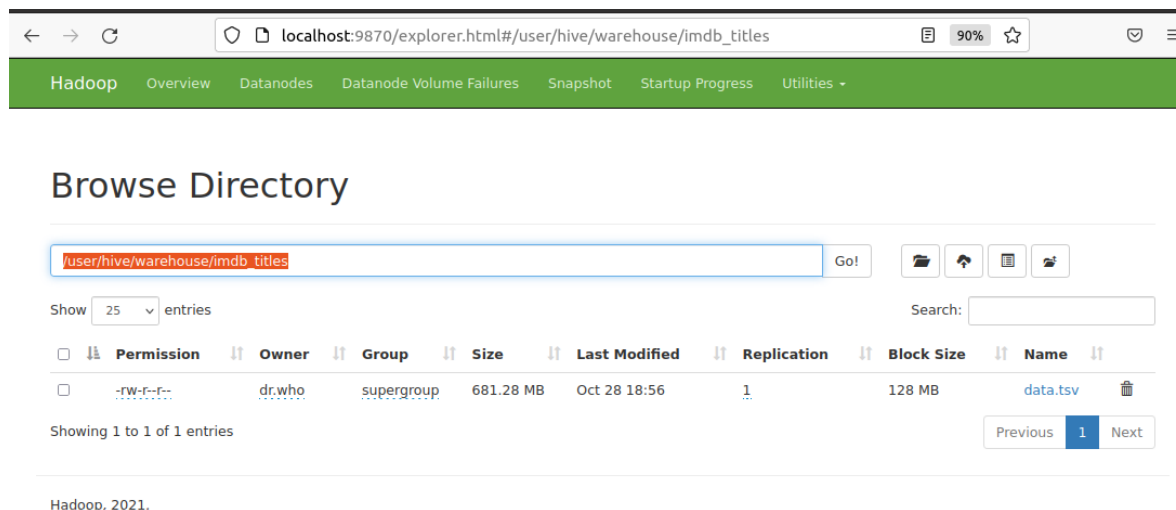
```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS imdb_titles
(
  tconst STRING,
  titleType STRING,
  primaryTitle STRING,
  originalTitle STRING,
  isAdult INT,
  startYear int,
  endYear int,
  runtimeMinutes int,
  genres STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
TBLPROPERTIES ("skip.header.line.count"="1");
```

- Cargamo-lo ficheiro title)basics)tsv, previamente cargado na raíz do sistema hdfs (este ficheiro foi obtido do dataset de [imdb](#)), na táboa:

```
hive> LOAD DATA INPATH '/data.tsv' INTO TABLE imdb_titles;
```

É curioso onde vai garda-la táboa cargada, xa que lévaa ata o directorio hdfs

/user/hive/warehouse/imdb_titles/data.tsv



Hadoop

Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Browse Directory

/user/hive/warehouse/imdb_titles Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	dr.who	supergroup	681.28 MB	Oct 28 18:56	1	128 MB	data.tsv

Showing 1 to 1 of 1 entries Previous 1 Next

Hadoop, 2021.

Se posteriormente quixeramos volver a cargar nunha táboa o ficheiro, a ruta tería que ser a de
> xa que aló moveu o arquivo dende a **raíz** do sistema hdfs.

```
hive> LOAD DATA INPATH '/user/hive/warehouse/imdb_titles/data.tsv' INTO TABLE
imdb_titles2;
```

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS imdb_titles (
  > tconst STRING,
  > titleType STRING,
  > primaryTitle STRING,
  > originalTitle STRING,
  > isAdult INT,
  > startYear int,
  > endYear int,
  > runtimeMinutes int,
  > genres STRING)
  > ROW FORMAT DELIMITED
  > FIELDS TERMINATED BY '\t'
  > STORED AS TEXTFILE
  > TBLPROPERTIES ("skip.header.line.count"="1");
OK
Time taken: 3.872 seconds
hive> LOAD DATA INPATH '/data.tsv' INTO TABLE imdb_titles;
FAILED: SemanticException Line 1:17 Invalid path '/data.tsv': No files matching path hdfs://hadoop-name:9000/data.tsv
```

A táboa xa a creara anteriormente polo que esta sentenza xa non fai nada

Nesta 2ª ocasión que pedín carga-la táboa, aparéceme o erro indicando que non se atopa o arquivo na raíz, porque hive xa o movera anteriormente. Realmente non me fai falta volver a cargala, xa está dispoñible e podo lanzar sentencias SQL sobre ela con normalidade.

```
hive> SELECT * FROM imdb_titles LIMIT 10;
hive> SELECT COUNT(*) FROM imdb_titles;
```

```
hive> select count(*) from imdb_titles
> ;
Query ID = administrador_20211028200310_3e58c541-a07d-4247-9dcd-4d074f2f85eb
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_1635441888784_0002, Tracking URL = http://hadoop-name:8088/proxy/application_1635441888784_0002/
... (Command) ... User (hadoop) hadoop@centos:~/all$ job_1635441888784_0002
```

As consultas hive son convertidas a aplicacións MapReduce para a súa execución e polo tanto pódense seguir dende o interface web de Yarn.

The screenshot shows the Hadoop YARN web interface at localhost:8088/cluster. The top section displays 'Cluster Metrics' with various counters like Apps Submitted, Pending, Running, Completed, etc. Below this, 'Cluster Node Metrics' shows the number of active and decommissioning nodes. The 'Scheduler Metrics' section provides details about the scheduler type and resources. The bottom section, 'Show 20 entries', lists running applications with columns for ID, User, Name, Application Type, Queue, Priority, Start Time, Launch Time, Finish Time, and State. Two applications are visible: a Hive application (select count(*) from imdb_titles) and a Spark application (pysparkshell).

Probamos que pasa no caso de crea-la táboa como interna?

Ó carga-los datos mediante LOAD DATA dende HDFS, os datos de orixe elimínanse? E que pasa se se cargan dende o sistema local con LOAD DATA LOCAL?

- Creamos unha nova táboa para ser almacenada en formato `PARQUET` :

```
hive> CREATE TABLE IF NOT EXISTS imdb_titles_pq
(
  tconst STRING,
  titleType STRING,
  primaryTitle STRING,
  originalTitle STRING,
  isAdult INT,
  startYear int,
  endYear int,
  runtimeMinutes int,
  genres STRING
)
STORED AS PARQUET;
```

- Cargamo-la nova táboa cos datos da outra táboa previamente usada `imdb=titles`:

```
hive> INSERT INTO TABLE imdb_titles_pq SELECT * FROM imdb_titles;
```

- Comparamo-lo rendemento dunha mesma consulta que, por exemplo, conte a cantidade de películas que hai en cada unha das 2 táboas iguais:

```
hive> SELECT * FROM imdb_titles_pq;
hive> SELECT * FROM imdb_titles;
```

Por que tarda unha máis ca outra? Acaso non teñen os mesmos datos?

e das táboas Hive

As **particións** son un xeito de dividir unha gran táboa en táboas máis pequenas baseadas nos valores dunha columna (unha **partition** para cada valor distinto desa columna), mentres que os **bucketing** son unha técnica para dividi-los datos dun xeito manexable (pódense especificar cantos buckets se queren).

Ambas opcións están pensadas para mellora-lo rendemento eliminando a lectura completo dunha táboa cun conxunto grande de datos nun sistema de ficheiros HDFS. A principal diferenza entre Partitioning e Bucketing está no xeito en que se dividen os datos.

O **Partitioning** é un xeito de organizar táboas grandes en táboas lóxicas máis pequenas baseadas en valores de columnas; unha táboa lóxica (partition) para cada valor distinto.

As táboas en Hive son creadas como directorios en HDFS; unha táboa pode ter unha ou máis particións que corresponden a un subdirectorio para cada partición dentro dun directorio de táboa.

Por exemplo, considerando unha táboa co censo de España que contivese os códigos postais, cidades, provincias, etc, etc; crear unha partición en base ás provincias, dividiría a táboa en 52 trozos (50 provincias + 2 cidades autónomas), e isto permitiría obter resultados de busca moito máis rápidos en consultas como a dun código postal nunha provincia (Provincia = 'Pontevedra' e CP ='36203'), dado que se buscaría nunha sóa partición de directorio, a desa provincia.

En todo caso hai que ser cautos co número de particións creadas, xa que se hai demasiadas particións, créanse demasiados subdirectorios nun directorio de táboa, o que conleva unha sobrecarga innecesaria do **namenode**, en tanto que ten que manter tódolos metadatos para o sistema de ficheiros en memoria.

Exemplo:

```
hive> CREATE TABLE codigos_postais(  
  Id int,  
  Comunidade_autonoma string,  
  Cidade string,  
  Cp int)  
PARTITIONED BY(provincia string)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ',';
```

Asemade, as particións pódense incorporar ás táboas con posterioridade á súa creación, así como renomealas, ou eliminalas:

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-AlterTable/Partition/Column>

O é unha técnica para dividi-los datos en ficheiros pequenos máis manexables (especificando o número de buckets a crear). O valor de columna de bucketing será &as&eado a partir do número de buckets escollido polo/a usuario/a.

Pódense crear buckets só para unha columna da táboa, ou tamén pódense crear nuha táboa particionada para unha división maior dos datos co obxecto de mellora-lo rendemento das consultas na táboa particionada.

Cada bucket almacénase como un ficheiro dentro do directorio da táboa ou dos directorios das particións. Olo, lembra que a partición crea un directorio e que podes ter unha partición en base a unha ou máis columnas.

Continuando co anterior exemplo, no que xa se tiña unha partición sobre a columna Provincia, que implicaba a creación de 52 subdirectorios no directorio da táboa, se se crease un bucketing con valor de 10 na coluna de 3ódigo Postal, crearíanse 10 ficheiros para cada subdirectorio particionado.

Exemplo:

```
hive> CREATE TABLE codigos_postais(  
Id int,  
Comunidade_autonoma string,  
Cidade string,  
Cp int)  
PARTITIONED BY(provincia string)  
CLUSTERED BY Cp INTO 10 BUCKETS  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ',';
```

Observacións e diferencias entre o particionado e o bucketing:

- Crear particións en base ó código postal, non sería unha boa práctica, posto que implicaría a creación de case 12.000 directorios en HDFS (en España hai 11.752 códigos postais).
- Dado que o bucketing usa hashing, se os datos non están distribuídos equilibradamente entre &as&es, resultarían ficheiros desiguais, o que pode conlevar problemas de rendemento.

Créase un directorio sobre HDFS para cada partición	Créase un ficheiro sobre HDFS para cada bucket
Pódense ter unha ou máis columnas de particionado	Sóo se pode ter unha columna de Bucketing
Non se pode escolle-lo número de particións a crear	Pódese o número de buckets a crear
	Pódese facer Bucketing sobre unha táboa particionada
Utiliza a cláusula PARTITIONED BY	Utiliza a cláusula CLUSTERED BY

A particións de Hive son un xeito de organiza-las táboas dividíndoas en partes diferentes baseadas en claves de partición.

A partición é útil cando a táboa ten unha ou máis **chaves de partición** (as claves de partición son elementos básicos para determinar como se almacenan os datos na táboa).

Exemplo con detalles: <https://www.guru99.com/hive-partitions-buckets-example.html>

“Caso no que se teñen os datos completos de comercio electrónico correspondentes a operacións na India (os datos están almacenados na súa totalidade con indicación de a que estado -dos 38 estados existentes- corresponde cada operación). Se se toma a columna estatal como chave de partición pódense obter 38 particións, de tal xeito que os datos de cada estado se vexan separadamente en táboas de partición.”

Nas seguintes capturas de pantalla amósase o seguinte:

1. Creación dunha táboa para os estados da India con 3 columnas: state, district, e enrolments.

```
hive> create table allstates(state string, District string, Enrolments string)
row format delimited fields terminated by ',';
```

```
hive> create table allstates(state string, District string, Enrolments string)
```

```
1 > row format delimited
> fields terminated by ',';
```

creation of table "allstates"

2. Carga dos datos dos estados da India na táboa.

```
hive> Load data local inpath '/home/hduser/Desktop/AllStates.csv' into table
allstates;
```

3. Creación da táboa de particións coa columna state como chave de partición.

```
hive> create table state_part(District string, Enrolments string) PARTITIONED
BY(state string);
```

```
hive> load data local inpath '/home/hduser/AllStates.csv' into table allstates;
```

```
Loading data to table default.allstates
```

```
Table default.allstates stats: [numFiles=1, totalSize=36913]
```

```
OK
```

```
Time taken: 1.459 seconds
```

```
hive> create table state_part(District string, Enrolments string) PARTITIONED BY(state string);
```

```
OK
```

```
Time taken: 0.199 seconds
```

Loading Data into "allstates"

creation of partition table "state_part"

4. Para poder usar particións hai que activa-lo modo de partición dinámico, configurando como "nonstrict" este modo.

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

5. Carga dos datos na táboa de particións.

```
hive> INSERT OVERWRITE TABLE state_part PARTITION(state) SELECT district,
enrolments, state from allstates;
```


6. Como consecuencia, créanse as táboas de partición baseadas nos valores da columna state como chave de partición.

```
hive> set hive.exec.dynamic.partition.mode=nonstrict
>
hive> insert overwrite table state part PARTITION(state) SELECT district,enrolments,state from allstates;
Query ID = hduser_20151104161604_ce10a013-7e6e-4545-94b9-b26dcaad8879
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_201511041430_0001, Tracking URL = http://localhost:50030/jobdetails
Kill Command = /usr/local/hadoop-1.2.1/libexec/./bin/hadoop job -kill job_201511041430_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2015-11-04 16:16:13,677 Stage-1 map = 0%, reduce = 0%
2015-11-04 16:16:18,699 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.48 sec
2015-11-04 16:16:19,702 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.48 sec
MapReduce Total cumulative CPU time: 1 seconds 480 msec
Ended Job = job_201511041430_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:54310/user/hive/warehouse/state_part/.hive-staging_hive_2015-11-04_16-16-04_2
Loading data to table default.state_part partition (state=null)
Time taken for load dynamic partitions : 5282
Loading partition {state=Haryana}
Loading partition {state=Uttarakhand}
Loading partition {state=Daman_and_Diu}
Loading partition {state=Puducherry}
Loading partition {state=Uttar_Pradesh}
Loading partition {state=Assam}
Loading partition {state=Others}
Loading partition {state=Arunachal_Pradesh}
Loading partition {state=Lakshadweep}
Loading partition {state=West_Bengal}
Loading partition {state=Sikkim}
Loading partition {state=Himachal_Pradesh}
Loading partition {state=Jharkhand}
Loading partition {state=Tripura}
Loading partition {state=Punjab}
Loading partition {state=Tamil_Nadu}
Loading partition {state=Gujarat}
```

4

5

Making partition based on "state" field

6

Creation Partition tables using "state" as partition key during Map reduce process

7. As 38 particións pódense observar no sistema de almacenamento HDFS con nomes de ficheiro correspondentes a cada estado da India.

```
hduser@datamatics-Ubuntu: /usr/local/hadoop-1.2.1/bin$ ./hadoop dfs -ls /user/hive/warehouse/state_part
Warning: SHADOOP_HOME is deprecated.

Found 38 items
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Andaman_and_Nicobar_Island
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Andhra_Pradesh
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Arunachal_Pradesh
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Assam
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Bihar
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Chandigarh
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Chhattisgarh
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Dadra_and_Nagar_Haveli
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Daman_and_Diu
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Delhi
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Goa
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Gujarat
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Haryana
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Himachal_Pradesh
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Jammu_and_Kashmir
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Jharkhand
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Karnataka
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Kerala
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Lakshadweep
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Madhya_Pradesh
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Maharashtra
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Manipur
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Meghalaya
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Mizoram
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Nagaland
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Odisha
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Others
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Puducherry
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Punjab
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Rajasthan
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Sikkim
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Tamil_Nadu
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Tripura
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Uttar_Pradesh
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Uttarakhand
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=West_Bengal
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=california
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=newyork
```

7

Total 38 states present in table

38 partition tables stored in HDFS system

Os Buckets (literalmente traducido sería caldeiros ou baldes) utilízanse para dividi-los datos das táboas en múltiples ficheiros, para optimiza-las consultas.

Os datos que se atopan troceados nas particións aínda poden dividirse máis en buckets.

A división faise en base a un Hash sobre as columnas determinadas que se seleccionan dunha táboa; os buckets utilizan por detrás algún algoritmo de Hashing para ler cada rexistro e colocalos nos diferentes buckets.

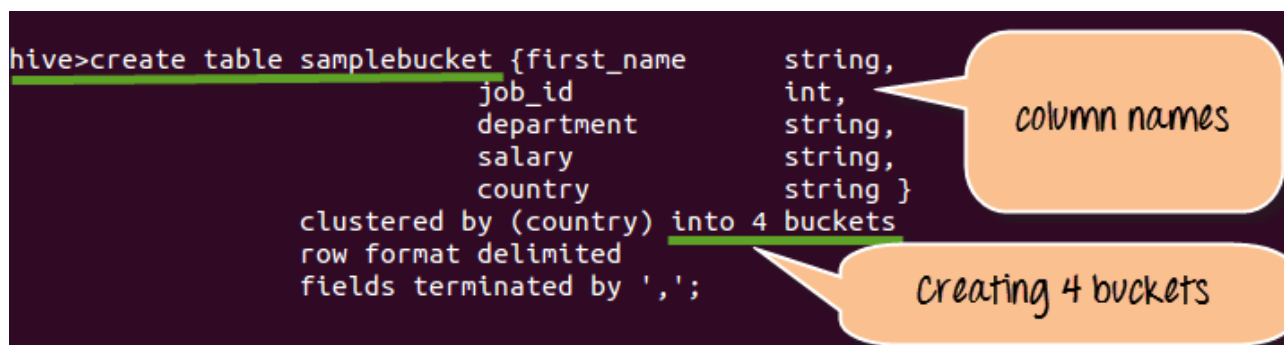
Exemplo con detalles: <https://www.guru99.com/hive-partitions-buckets-example.html>

Nas seguintes capturas de pantalla amósase o seguinte:

1. Para poder usa-los buckets en Hive, hai que habilitalos, activando este modo do seguinte xeito:

```
hive> set.hive.enforce.bucketing=true;
```

2. Creación de 4 buckets sobre unha táboa con varias columnas; a medida que se cargan os datos, estes colocaranse automaticamente nos distintos buckets, en base ó hash xerado da columna country.

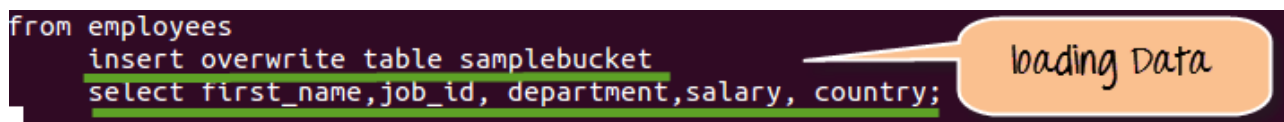


```
hive>create table samplebucket {first_name      string,
                                job_id          int,
                                department       string,
                                salary           string,
                                country          string }
                                clustered by (country) into 4 buckets
                                row format delimited
                                fields terminated by ',';
```

The screenshot shows a Hive command to create a table named 'samplebucket'. The table has five columns: first_name (string), job_id (int), department (string), salary (string), and country (string). It is clustered by the 'country' column into 4 buckets. The row format is delimited and fields are terminated by commas. Two orange callout boxes are present: one pointing to the column definitions labeled 'column names', and another pointing to 'into 4 buckets' labeled 'Creating 4 buckets'.

3. Carga dos datos nos 4 buckets da táboa. Partido da existencia previa dunha táboa “employees” xa cargada no sistema Hive, e coas columnas desexadas, cárganse os datos da táboa de empregados nos buckets da táboa samplebucket:

```
hive> INSERT OVERWRITE TABLE samplebucket SELECT first_name,job_id,department,
salary, country FROM employees;
```

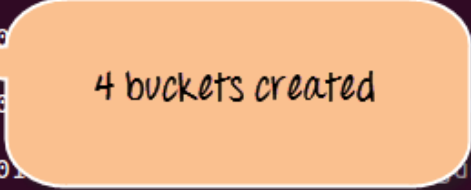


```
from employees
insert overwrite table samplebucket
select first_name,job_id, department,salary, country;
```

The screenshot shows the continuation of the Hive command to load data from the 'employees' table into the 'samplebucket' table. The command is 'insert overwrite table samplebucket select first_name,job_id, department,salary, country;'. An orange callout box labeled 'loading Data' points to the 'insert overwrite' part of the command.

4. O resultado da carga dos datos sería a creación dos ficheiros correspondentes a cada un dos 4 buckets.


```
guru99hive@ubuntu:~/Hadoop_YARN/hadoop-2.2.0/bin$ ./hadoop fs -ls /user/hive/
Found 3 items
--rwx---- 1 guru guru 4602 2015-11-02 09:30 /user/hive/guru99db/samplebucket/
000000 0
--rwx---- 1 guru guru 4602 2015-11-02 09:30 /user/hive/guru99db/samplebucket/
000000_1
--rwx---- 1 guru guru 4602 2015-11-02 09:30 /user/hive/guru99db/samplebucket/
000000 2
--rwx---- 1 guru guru 4602 2015-11-02 09:30 /user/hive/guru99db/samplebucket/
000000 3
```



4 buckets created

* Beeline CLI

Beeline pode correr en modo embebido e en modo remoto. En modo embebido é similar a Hive CLI, pero correndo un servidor HiveServer2, namentres que en modo remoto, conéctase a un servidor remoto HiveServer2 utilizando o protocolo Thrift.

Unha das diferencias máis evidentes entre Beeline e Hive CLI, é que este último proporciona autentificación: non se pode conectar a Hive CLI sen credenciais de conta e contrasinal.

Algunhas vantaxes de Beeline fronte a Hive CLI:

- Cliente de acceso remoto para executar consultas contra o servidor Hive.
- Concurrency de múltiples clientes e autentificación de usuarios/as.
- Mellor soporte para clientes API como JDBC e ODBC

Diferencias entre Hive CLI e Hive Server + Beeline (cliente JDBC):

https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.6.5/bk_data-access/content/beeline-vs-hive-cli.html

Nota: inicializa-lo esquema é un paso previo imprescindible para poder executar consultas. Como xa o fixemos anteriormente, durante a instalación, agora non precisamos repetilo:

```
$ /usr/share/hive/bin/schematool -dbType derby -initSchema
```

Para poder usar Beeline ten que estar correndo o servidor HS2 (HiveServer2), polo que é necesario arrancalo co comando `hiveserver2`, que quedará á escoita no porto 10000.

```
$ /usr/share/hive/bin/hiveserver2
```

* eeline en modo embebido

Unha forma rápida de conectar con Hive para facer consultas HiveQL usando Beeline, de forma similar a Hive CLI, é a través do modo embebido.

No modo embebido, lánzase internamente o servizo HiveServer2, polo que non é recomendable para uso de produción, senón **unicamente para probas, xa que e e corren no mesmo proceso.**

Para arrancar Beeline en modo embebido úsase o seguinte comando:

```
$ /usr/share/hive/bin/beeline
```

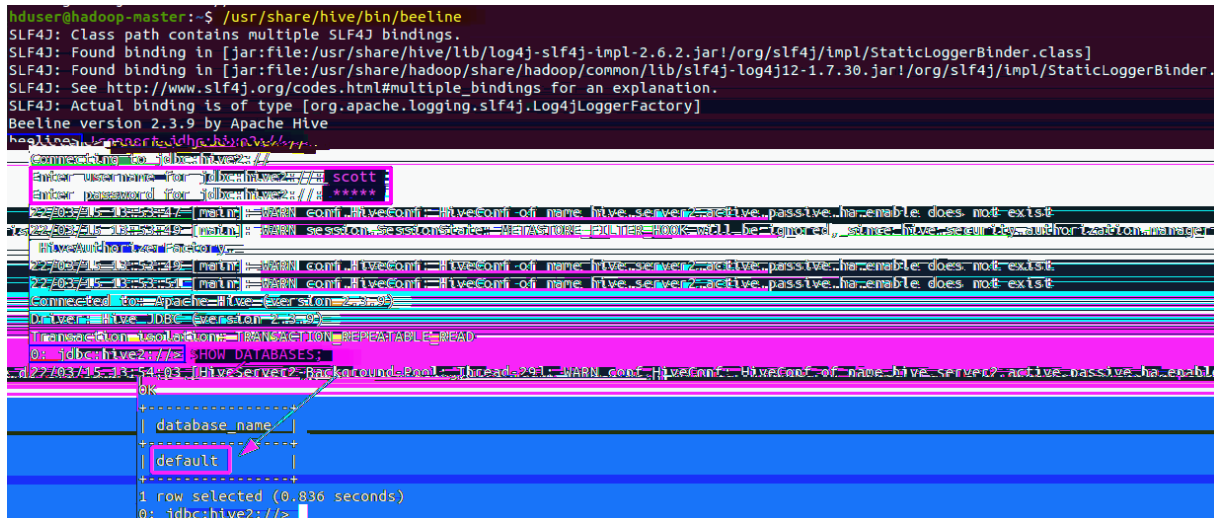
E unha vez na consola de beeline, para conectar a Hive, hai que usa-lo seguinte comando:

```
beeline> !connect jdbc:hive2://
```

Nota: de xeito predeterminado, asúmesse , polo que o anterior enderezo será como se houbésemos posto:

Inmediatamente pídense por pantalla as credencias (nome de conta e contrasinal) que están establecidas por defecto no servidor HiveServer2 como:

- conta de usuario/a:
- contrasinal:



De xeito alternativo tamén se poden pasa-las credenciais e o URL embebido, xuntos no propio comando CLI de conexión, cos parámetros:

- -u url de conexión
- -n nome de conta
- -p contrasinal

```
$ /usr/share/hive/bin/beeline -u jdbc:hive2:// -n scott -p tiger
```

```

hadoop@hadoop-master:~$ /usr/share/hive/bin/beeline -u jdbc:hive2:// -n scott -p tiger
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/share/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/share/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.30.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]
Connecting to jdbc:hive2://
22/03/15 13:04:33 [main]: WARN conf.HiveConf: HiveConf of name hive.server2.active.passive.ha.enable does not exist
22/03/15 13:04:33 [main]: WARN conf.HiveConf: HiveConf of name hive.server2.active.passive.ha.enable does not exist
22/03/15 13:04:37 [main]: WARN conf.HiveConf: HiveConf of name hive.server2.active.passive.ha.enable does not exist
22/03/15 13:04:51 [main]: WARN conf.HiveConf: HiveConf of name hive.server2.active.passive.ha.enable does not exist
Connected to: Apache Hive (version 2.3.9)
Driver: Hive JDBC (version 2.3.9)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.3.9 by Apache Hive
0: jdbc:hive2://> SHOW DATABASES;
22/03/15 13:05:09 [HiveServer2-Background-Pool: Thread-30]: WARN conf.HiveConf: HiveConf of name hive.server2.active.passive.ha.enable does not exist
OK
+-----+
| database_name |
+-----+
| default      |
+-----+
1 row selected (1.312 seconds)
0: jdbc:hive2://>

```

Para saír da consola, basta con executa-la sentenza

```

0: jdbc:hive2://> !quit
Closing: 0: jdbc:hive2://

```

Para ve-los comandos dispoñibles basta con executa-la sentenza

```

0: jdbc:hive2://> help

```

Ademais dos comandos do cliente &hive, tamén aparecen os comandos beeline, que son os que comezan polo símbolo de exclamación !

* eeline para Conectar a Hive en modo remoto

En modo Remoto, o proceso Hive.server está correndo nun cluster remoto. Podémosnos conectar remotamente a Hive usando Beeline proporcionando o enderezo IP e porto, no URL de conexión JDBC.

```
$ $HIVE_HOME/bin/hiveserver2
$ $HIVE_HOME/bin/beeline -u jdbc:hive2://$HS2_HOST:$HS2_PORT
```

Logicamente, para poder conectarnos ó servidor remoto mediante Beeline, o primeiro que se necesita é que estea correndo o servizo HiveServer2 no servidor remoto. Se non estivese xa correndo, hai que iniciar HiveServer2 con:

```
$ /usr/share/hive/bin/hiveserver2
```

The screenshot shows a terminal window on a Linux system where the user 'hduser' is running 'hiveserver2' and 'jps' commands. The 'jps' command output lists several processes, including 'HiveServer2' (PID 8368) and 'HiveMetaStore' (PID 8369). The 'netstat -ltn' command output shows that the 'HiveServer2' process is listening on port 10000. A red arrow points from the 'HiveServer2' entry in the 'jps' output to the 'HiveServer2' entry in the 'netstat' output. To the right of the terminal window, a web browser window displays the 'HiveServer2' status page. The page shows 'Active Sessions' (0), 'Open Queries' (0), and 'Last Max 25 Closed Queries' (0). The 'Software Attributes' section shows the 'Hive Version' as '2.3.9' and the 'Hive Compiled' date as 'Tue Jun 1 14:02:14 PDT 2021, chao'.

Nota; Por defecto Hive.server corre no ...) < o interface (eb no

Para conectar a un Hive que estea correndo nun cluster remoto, hai que pasa-lo enderezo IP e o porto no comando de conexión JDBC.

```
beeline> !connect jdbc:hive2://hadoop-master:10000 scott tiger
```

```
hduser@hadoop-master:~$ /usr/share/hive/bin/beeline
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/share/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/share/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Beeline version 2.3.9 by Apache Hive
beeline> !connect jdbc:hive2://hadoop-master:10000 scott tiger
Connecting to jdbc:hive2://hadoop-master:10000
Connected to: Apache Hive (version 2.3.9)
Driver: Hive JDBC (version 2.3.9)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://hadoop-master:10000> show databases;
+-----+
| database_name |
+-----+
| default       |
+-----+
1 row selected (1.379 seconds)
0: jdbc:hive2://hadoop-master:10000>
```

En caso de que se execute localmente o servidor, tamén pódese usar localhost, o &ostname, ou 1 7)0)1 en vez do enderezo IP remoto.

O porto por defecto de HiveServer2 é o **porto 10000**, pero pode cambiarse na propiedade do ficheiro de configuración .

E o interface web por defecto atópase no **porto 10002**, dende o que se proporciona información das sesións activas e da configuración, métricas e logs.

<http://hadoop-master:10002>

The screenshot shows the HiveServer2 web interface at localhost:10002. The interface includes a navigation bar with links for Home, Local logs, Metrics Dump, Hive Configuration, Stack Trace, and Umap Daemons. The main content area displays three sections:

- Active Sessions:** A table showing one active session for user 'scott' at IP '10.0.2.8'. The table has columns for User Name, IP Address, Operation Count, Active Time (s), and Idle Time (s). Below the table, it states 'Total number of sessions: 1'.
- Open Queries:** A table showing no open queries. The table has columns for User Name, Query, Execution Engine, State, Opened Timestamp, Opened (s), Latency (s), and Drilldown Link. Below the table, it states 'Total number of queries: 0'.
- Last Max 25 Closed Queries:** A table showing the last closed query for user 'scott'. The table has columns for User Name, Query, Execution Engine, State, Opened (s), Closed Timestamp, Latency (s), and Drilldown Link. The query 'show databases' is shown as 'FINISHED' at 'Tue Mar 15 16:18:08 CET 2022'. Below the table, it states 'Total number of queries: 1'.

Software Attributes

Tamén pódese executa-lo servidor en segundo plano do seguinte xeito, que creará, no cartafol dende onde se invoque, un ficheiro cos logs da execución do servizo:

```
$ nohup /usr/share/hive/bin/hiveserver2 &
```

- O operador de control `&` ó final dun comando, executa ese comando nunha subshell en segundo plano, polo que a shell non espera ó remate do comando e presenta o prompt inmediatamente.
- O comando `nohup` permite mante-la execución dun comando (que se lle pasa como un argumento) pese a pecha-la terminal (`Ctrl+C` ou `Ctrl+D`), xa que fai que se execute de forma independente á sesión. Basicamente, o que fai é ignora-lo sinal `SIGINT` (sinal que se envía a un proceso cando a terminal que o controla se pecha), isto implica que aínda que se peche a terminal, o proceso siga executándose. Por defecto, a saída do comando, que normalmente aparecería directamente na terminal, sácase a un ficheiro chamado `nohup.out` que aparecerá na ruta dende a que se execute o comando.
- Tipicamente úsase `nohup` en combinación con `(nohup /unscript)s& @`; deste xeito a execución en segundo plano dun script calquera, grazas ó comando `nohup` permite a continuidade da execución en caso de calquera problema coa sesión, shell de execución, etc:

```
hduser@hadoop-master: /usr/share/hive$ jps
3217 SecondaryNameNode
3011 NameNode
13397 Jps
8935 RunJar
3565 ResourceManager
hduser@hadoop-master: /usr/share/hive$ ls
bin          conf          examples     jdbc         LICENSE      nohup.out    RELEASE_NOTES.txt
binary-package-licenses  derby.log    hcatalog     lib          metastore_db NOTICE      scripts
hduser@hadoop-master: /usr/share/hive$ cat nohup.out
2022-03-15 15:59:21: Starting HiveServer2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/share/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/share/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.30.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]
OK
hive$
```

Por último, tamén pódese lanza-lo servidor Hive Server2 como servizo en segundo plano do seguinte xeito:

```
$ hive --service hiveserver2
```

Curiosamente para lanza-lo hive CLI faise de xeito similar, pero sen invocar a hiveserver2, claro:

```
$ hive --service
```

Para dete-lo servidor Hive Server 2 dende a liña de comandos envíase un sinal de parada ó proceso hiveserver2, mediante o seguinte comando:

```
$ hive --service hiveserver2 --stop
```



```

hadoop@hadoop-master: /usr/share/hive$ hive --service hiveserver2
2022-03-16 00:38:05: Starting HiveServer2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/share/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/share/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.30.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]
hadoop@hadoop-master: /usr/share/hive$ hive --service hiveserver2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/share/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/share/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.30.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:/usr/share/hive/lib/hive-common-2.3.9.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 or tez) or using Hive 1.X releases.
hive>

```

-----> A probar por min para Flume NON FACER POLO DE AGORA!!!

necesitamos configura-lo metastore en remoto para poder acceder a el remotamente na práctica de Flume

Metastore remoto

Para configura-lo metastore en remoto, necesitamos modifica-la propiedade

hive.metastore.uris do arquivo **hive-site.xml** situado en **\$HIVE_HOME/conf**:

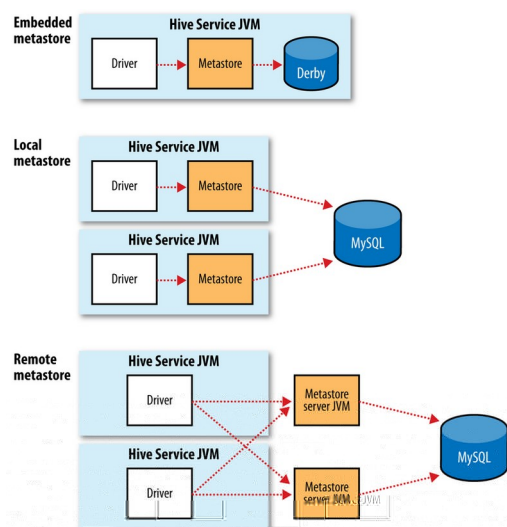
```

hive-site.xml
<property>
  <name>hive.metastore.uris</name>
  <value>thrift://hadoop-master:9083</value>
</property>

```

Unha vez configurado, volvemos a arrancar Hive mediante **hiveserver2** e a continuación, cada vez que arranquemos Hive, necesitamos tamén arranca-lo Metastore, xa que agora funciona como un servizo totalmente desacoplado:

```
$ hive --service metastore
```



<-----

-----> A probar por min para SQOOP NON FACER POLO DE AGORA!!!

Para que funcione a inxesta de datos en Hive mediante Sqoop, necesitamos engadir unha librería a Sqoop:

```
$ cp $HIVE_HOME/lib/hive-common-3.1.2.jar $SQOOP_HOME/lib  
/usr/share/hive/ /usr/share/sqoop/
```

<-----

Introducción ó uso de Apache Hive

- Lanza-la consola de Hive:

```
$ /usr/share/hive/bin/hive
```

ou, se se engadiu a ruta ós binarios de hive ó PATH:

```
$ hive
```

+xemplo , ' Creación e uso de bases de datos

- Crear unha base de datos.

A sintaxe para crear unha base de datos é a seguinte:

```
CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>
```

```
hive> CREATE DATABASE compras_bd;
```

- Ve-la lista de bases de datos existentes.

A sintaxe para ve-la lista de bases de datos é a seguinte:

```
SHOW DATABASES|SCHEMAS
```

```
hive> SHOW DATABASES;
```

- Usar unha base de datos

A sintaxe para abri-la base de datos a usar é a seguinte.

```
USE <database name>
```

```
hive> USE compras_bd;
```

+xemplo - ' Creación e carga de t boas

Utilizaremos o ficheiro de vendas de exercicios anteriores: `purc&ases.txt`, con información sobre as vendas en diferentes tendas.

En Hive, unha táboa é un conxunto de datos que utiliza un esquema para ordena-los datos por unha serie de identificadores dados.

A sintaxe para a creación de táboas é a seguinte:

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.] table_name  
[(col_name data_type [COMMENT col_comment], ...)]  
[COMMENT table_comment]  
[ROW FORMAT row_format]  
[FIELDS TERMINATED BY char]  
[STORED AS file_format]
```

- Crear unha táboa cunha estrutura axeitada ós datos, que se cargarán a continuación:

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS compras (  
data STRING,  
hora STRING,  
tenda STRING,  
categoría STRING,  
venda FLOAT,  
pago STRING)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY '\t'  
STORED AS TEXTFILE;
```

A sintaxe para ve-la lista das táboas e vistas da base de datos actual, se non se explicita outra, é a seguinte:

```
SHOW TABLES [IN db_name]
```

- Podemos consulta-las táboas existentes na base de datos :

```
hive> SHOW TABLES;
```

A sintaxe para amosa-lo esquema dunha táboa é a seguinte:

```
DESCRIBE [EXTENDED|FORMATTED] [db_name.]table_name
```

- Podemos consulta-la estrutura da táboa creada, e incluso obter moita máis información, como a súa ubicación física e se é externa ou interna:

```
hive> DESCRIBE compras;  
hive> DESCRIBE FORMATTED compras;
```

A sintaxe para carga-los datos dun ficheiro dende o sistema de arquivos a unha táboa, na que o formato dos datos é o mesmo en ambas (o .er , e da táboa debe coincidir coa estrutura do ficheiro de orixe), é a seguinte:

```
LOAD DATA [LOCAL] INPATH file_path [OVERWRITE] INTO TABLE table_name [PARTITION (partcol1=val1, partcol2=val2 ...)]
```

- **LOCAL** úsase se a ruta fose ó sistema de arquivos local, en vez do hdfs.
- **OVERWRITE** permite sobrescribi-los datos que puidera haber xa na táboa.

- Podemos carga-los datos dende o sistema de ficheiros local:

```
hive> LOAD DATA LOCAL INPATH '/home/hduser/Descargas/purchases.txt' INTO TABLE compras;
```

- De xeito predeterminado Hive buscará en HDFS (nese caso non se usa LOCAL):

```
hive> LOAD DATA INPATH 'purchases.txt' INTO TABLE compras;
```

Para verifica-los datos cargados pódese facer unha consulta co comando :

```
hive> SELECT * FROM compras;
```

Tamén pódese escolle que se amosen só os datos correspondentes ás columnas desexadas:

```
hive> SELECT tenda, venda FROM compras;
```

Nota: **CREATE TABLE table_name [AS select_statement]**

Ás veces necesítase almacena-la saída dunha consulta Hive nunha nova táboa. Mediante a cláusula (non funciona con táboas externas), as definicións das columnas da nova táboa derívase das columnas recuperadas na consulta:

```
hive> CREATE TABLE novas_compras AS SELECT * from compras;
```

Nota: **CREATE TABLE table_name LIKE existing_table_or_view_name**

Outra opción consiste en crear unha nova táboa baleira, pero coa mesma estrutura que a dunha táboa xa existente mediante a cláusula :

```
hive> CREATE TABLE novas_compras2 LIKE compras;
```

+xemplo . ' Consultas Hive / (S+ (+C) 0H+%+

Unha vez que definimos unha estrutura e indicamo-la fonte dos datos podemos definir consultas en HiveQL, unha variante NON-ANSI de SQL. Estas consultas SQL, antes da execución, son transformadas a MapReduce (ou máis recentemente a Tez ou DAG-Spark), é dicir, convértense en programas paralelizables e executables por un clúster sobre os datos almacenados sobre HDFS.

A sentenza `SELECT` úsase para recupera-los datos dunha táboa.

A cláusula `WHERE` permite indicar unha condición, usando diferentes operadores e funcións integradas que xeran expresións.

A combinación de ambas, filtra os datos que cumpren a condición e dá un resultado.

A sintaxe é a seguinte:

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

Máis información: <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Select>

- Pódense realizar unha consulta básica que amose o contido de toda a táboa.

```
hive> SELECT * FROM compras;
```

- Tamén pódese escolle que se amosen só os datos correspondentes ás columnas desexadas:

```
hive> SELECT tenda, venda FROM compras;
```

- Pódense realizar un par de consultas usando funcións de agregación/resumo:

```
hive> SELECT COUNT(*) FROM compras;
```

```
hive> SELECT MAX(venda) FROM compras;
```

- Tamén pódese realizar unha consulta usando unha condición (na que interveñen operadores), por exemplo para obte-los datos das vendas por riba dun importe mínimo:

```
hive> SELECT * FROM compras WHERE venda > 1000;
```

1 problema das subconsultas en Hive

A seguinte sentenza HIVE non funcionaría:

```
hive> SELECT * FROM compras WHERE venda > (SELECT AVG(venda) FROM compras);
```

A sentenza devolvería o seguinte erro:

```
FAILED: SemanticException Line 0:-1 Unsupported SubQuery Expression 'venda':  
Only SubQuery expressions that are top level conjuncts are allowed  
Time taken: 36.418 seconds, Fetched: 1 row(s)  
hive> SELECT * FROM compras WHERE venda > (SELECT AVG(venda) FROM compras);  
FAILED: SemanticException Line 0:-1 Unsupported SubQuery Expression 'venda': Only SubQuery expressions th  
at are top level conjuncts are allowed
```

Sen embargo, a mesma consulta especificando explicitamente o nome de columna venda na cláusula SELECT de nivel superior, si que funcionaría:

```
hive> SELECT venda, venda FROM compras WHERE venda > (SELECT AVG(venda) FROM  
compras);
```

As subconsultas en Hive teñen varias restricións para o seu uso:

- https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.0.1/using-hiveql/content/hive_hive_subquery_limitations.html
 - https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.2.4/bk_dataintegration/content/hive-013-feature-subqueries-in-where-clauses.html
-
- A subconsulta debe aparecer no lado dereito da expresión
 - Non admite subconsultas anidadas.
 - Unha mesma consulta só pode ter unha expresión de subconsulta.
 - O predicado da subconsulta debe aparecer como unha conectiva de nivel superior (top-level conjuncts).
 - As subconsultas admiten catro operadores lóxicos nos predicados das consultas: `<`, `>`, `=` e `<=>`.
 - Os operadores lóxicos `<` e `>` só poden seleccionar unha columna na subconsulta.
 - Os operadores `=` e `<=>` deben ter alomenos un predicado relacionado.
 - O lado esquerdo da subconsulta debe de “cualificar” (Bualif+) tódalas referencias ás columnas da táboa.
 - Só se permite facer referencia a columnas da consulta principal na cláusula `WHERE` da subconsulta.

- Os predicados da subconsulta que fan referencia a columnas da consulta principal deben utiliza-lo operador de predicado igual (=).
- O predicado da subconsulta non pode referirse unicamente ás columnas da consulta principal.
- As subconsultas correlacionadas con sentenzas `GROUP BY` só poden devolver unha ringleira.
- Tódalas referencias non cualificadas ás columnas da subconsulta deben resolverse nas táboas da subconsulta.
- As subconsultas correlacionadas non pode conter cláusulas “ (`into` (`ing`” (<https://bigdataprogrammers.com/windowing-functions-in-hive/>).

Asemade, Hive non soporta máis que un par de tipos moi concretos de subconsultas: usando os operadores `IN` e `EXISTS` na cláusula `WHERE`:

- <https://cwiki.apache.org/confluence/display/Hive/Subqueries+in+SELECT>
- <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+SubQueries>
- [https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.0.1/using-hiveql/content/hive use a subquery in a hive table.html](https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.0.1/using-hiveql/content/hive+use+a+subquery+in+a+hive+table.html)

Para a sentenza do exemplo, **unha posible solución alternativa sería crear unha vista, ou unha táboa temporal** (que se borra automaticamente ó pecha-la sesión) e logo utiliza-lo seu contido nunha nova consulta sen subconsulta.

- Creación de táboas temporais: [https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.0.1/using-hiveql/content/hive create a hive temporary table.html](https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.0.1/using-hiveql/content/hive+create+a+hive+temporary+table.html)

1. Crear unha táboa temporal cunha columna de tipo string.

```
CREATE TEMPORARY TABLE tmp1 (tname varchar(64));
```

2. Crear unha táboa temporal usando a sentenza `CREATE TABLE AS SELECT`.

```
CREATE TEMPORARY TABLE tmp2 AS SELECT c2, c3, c4 FROM mytable;
```

3. Crear unha táboa temporal usando a sentenza `CREATE TEMPORARY TABLE LIKE`.

```
CREATE TEMPORARY TABLE tmp3 LIKE tmp1;
```

- Crear unha táboa temporal co cálculo da media das vendas.

```
hive> CREATE TEMPORARY TABLE media AS SELECT AVG(venda) AS media_vendas FROM
compras;
```

- Facer un **INNER JOIN** das compras coa táboa temporal para obter só as que teñan un valor de venda superior á media.

```
hive> SELECT * FROM compras AS c INNER JOIN media AS m ON c.venda >= m.media_vendas;
```

```
hive> DESCRIBE media;
OK
media_vendas          double
Time taken: 0.075 seconds, Fetched: 1 row(s)
hive> SELECT * FROM media;
OK
225.19639966011047
Time taken: 0.261 seconds, Fetched: 1 row(s)
hive> SELECT * FROM compras AS c INNER JOIN media AS m ON c.venda >= m.media_vendas;
FAILED: SemanticException Cartesian products are disabled for safety reasons. If you know what you are doing, please set hive.strict.checks.cartesian.product to false and that hive.mapred.mode is not set to 'strict' to proceed. Note that if you may get errors or incorrect results if you make a mistake while using some of the unsafe features.
```

Previamente debemos de habilitar que HIVE permita realizar operacións de produto cartesiano coa cláusula `set hive.strict.checks.cartesian.product=false`, e, en caso necesario, habilita-lo modo non estricto, coa sentença `set hive.mapred.mode=nonstrict`.

Habilitar expresións con produto cartesiano: <https://blog.spacepatroldelta.com/a?ID=01600-6c29f541-c4cb-4747-b74f-896ec2862aa6>

```
hive> set hive.strict.checks.cartesian.product=false;
hive> SELECT data, hora, tenda, categoria, venda, pago FROM compras AS c INNER JOIN media AS m ON c.venda >= m.media_vendas;
```

Poderíase evitar crea unha táboa auxiliar facendo directamente o JOIN dos dous SELECT, pero penso que fai máis complicada a comprensión da consulta:

```
hive> SELECT data, hora, tenda, categoria, venda, pago FROM compras AS c INNER JOIN (SELECT AVG(venda) AS media_vendas FROM compras) AS m ON c.venda >= m.media_vendas;
```

Como referencia o resultado desta consulta sobre o ficheiro de compras de 2004 executada nun cluster cun namenode e catro datanodes;

```
Time taken: 194.455 seconds, Fetched: 2068497 row(s)
```

Exemplo de uso de JOIN con SELECTs en vez de subconsultas:

<https://stackoverflow.com/questions/68011840/only-subquery-expressions-that-are-top-level-conjuncts-are-allowed-error>

Operadores en Hive

Hai catro tipos de operadores en Hive.

Máis información:

<https://cwiki.apache.org/confluence/display/hive/languagemanual+udf#LanguageManualUDF-Built-inOperators>

- **Operadores relacionais:** para comparar dous operandos

Operador	Operando	Descrición
A = B	calquera tipo primitivo	TRUE if expression A is equivalent to expression B otherwise FALSE.
A != B	calquera tipo primitivo	TRUE if expression A is not equivalent to expression B otherwise FALSE.
A < B	calquera tipo primitivo	TRUE if expression A is less than expression B otherwise FALSE.
A <= B	calquera tipo primitivo	TRUE if expression A is less than or equal to expression B otherwise FALSE.
A > B	calquera tipo primitivo	TRUE if expression A is greater than expression B otherwise FALSE.
A >= B	calquera tipo primitivo	TRUE if expression A is greater than or equal to expression B otherwise FALSE.
A IS NULL	calquera tipo	TRUE if expression A evaluates to NULL otherwise FALSE.
A IS NOT NULL	calquera tipo	FALSE if expression A evaluates to NULL otherwise TRUE.
A LIKE B	Strings	TRUE if string pattern A matches to B otherwise FALSE.
A RLIKE B	Strings	NULL if A or B is NULL, TRUE if any substring of A matches the Java regular expression B, otherwise FALSE.
A REGEXP B	Strings	Same as RLIKE.

```
hive> SELECT * FROM compras WHERE tenda = 'San Jose';
```

```
hive> SELECT * FROM compras WHERE venda >= 1000;
```

- **Operadores aritméticos:** para realizar cálculos aritméticos que devuelven como resultado un número.

Operador	Operando	Descripción
A + B	calquera tipo numérico	Gives the result of adding A and B.
A - B	calquera tipo numérico	Gives the result of subtracting B from A.
A * B	calquera tipo numérico	Gives the result of multiplying A and B.
A / B	calquera tipo numérico	Gives the result of dividing B from A.
A DIV B	calquera tipo numérico	Gives the integer part resulting from dividing A by B. E.g 17 div 3 results in 5.
A % B	calquera tipo numérico	Gives the reminder resulting from dividing A by B.
A & B	calquera tipo numérico	Gives the result of bitwise AND of A and B.
A B	calquera tipo numérico	Gives the result of bitwise OR of A and B.
A ^ B	calquera tipo numérico	Gives the result of bitwise XOR of A and B.
~A	calquera tipo numérico	Gives the result of bitwise NOT of A.

Por exemplo, para obte-los importes das vendas en milleiros:

```
hive> SELECT venda / 1000 FROM compras;
```

- **Operadores lógicos:** expresións lóxicas que devolven `TRUE` ou `FALSE`, ou `NULL`.

Operador	Operando	Descripción
! A	boolean	Same as NOT A.
[NOT] EXISTS (subquery)		TRUE if the the subquery returns at least one row.
A AND B	boolean	TRUE if both A and B are TRUE, otherwise FALSE. NULL if A or B is NULL.
A IN (val1, val2, ...)	boolean	TRUE if A is equal to any of the values.
A NOT IN (val1, val2, ...)	boolean	TRUE if A is not equal to any of the values.
A OR B	boolean	TRUE if either A or B or both are TRUE, FALSE OR NULL is NULL, otherwise FALSE.
NOT A	boolean	TRUE if A is FALSE or NULL if A is NULL. Otherwise FALSE.

```
hive> SELECT * FROM compras WHERE tenda = 'San Jose' AND venda >= 1000;
```

- **Operadores complejos:** proporcionan mecanismos para acceder a tipos complejos

Operador	Operando	Descripción
A[n]	A is an Array and n is an int	Returns the nth element in the array A. The first element has index 0. For example, if A is an array comprising of ['foo', 'bar'] then A[0] returns 'foo' and A[1] returns 'bar'.
M[key]	M is a Map<K, V> and key has type K	Returns the value corresponding to the key in the map. For example, if M is a map comprising of {'f' -> 'foo', 'b' -> 'bar', 'all' -> 'foobar'} then M['all'] returns 'foobar'.
S.x	S is a struct	Returns the x field of S. For example for the struct foobar {int foo, int bar}, foobar.foo returns the integer stored in the foo field of the struct.

Operador de cadeas	Operando	Descripción
A B	strings	Concatenates the operands - shorthand for concat(A,B)

Funcións en Hive

Hai unha chea de funcións: aritméticas, de cadeas, de datas, de agregación,...

Máis información:

<https://cwiki.apache.org/confluence/display/hive/languagemanual+udf#LanguageManualUDF-Built-inFunctions>

Tipo devolto	Función	Descrición
BIGINT	round(double a)	It returns the rounded BIGINT value of the double.
BIGINT	floor(double a)	It returns the maximum BIGINT value that is equal or less than the double.
BIGINT	ceil(double a)	It returns the minimum BIGINT value that is equal or greater than the double.
double	rand(), rand(int seed)	It returns a random number that changes from row to row.
string	concat(string A, string B,...)	It returns the string resulting from concatenating B after A.
string	substr(string A, int start)	It returns the substring of A starting from start position till the end of string A.
string	upper(string A)	It returns the string resulting from converting all characters of A to upper case.
string	ucase(string A)	Same as above.
string	lower(string A)	It returns the string resulting from converting all characters of B to lower case.
string	lcase(string A)	Same as above.
string	trim(string A)	It returns the string resulting from trimming spaces from both ends of A.
string	ltrim(string A)	It returns the string resulting from trimming spaces from the beginning (left hand side) of A.
string	rtrim(string A)	rtrim(string A) It returns the string resulting from trimming spaces from the end (right hand side) of A.
string	regexp_replace(string A, string B, string C)	It returns the string resulting from replacing all substrings in B that match the Java regular expression syntax with C.
int	size(Map<K,V>)	It returns the number of elements in the map type.
int	size(Array<T>)	It returns the number of elements in the array type.
value of <type>	cast(<expr> as <type>)	It converts the results of the expression expr to <type> e.g. cast('1' as BIGINT) converts the string '1' to it integral representation. A NULL is returned if the conversion does not succeed.
	from_unixtime(int unixtime)	convert the number of seconds from Unix epoch (1970-01-01 00:00:00 UTC) to a string representing the timestamp of that moment in the current system time zone in the format of "1970-01-01 00:00:00"
string	to_date(string timestamp)	It returns the date part of a timestamp string: to_date("1970-01-01 00:00:00") = "1970-01-01"
int	year(string date)	It returns the year part of a date or a timestamp string: year("1970-01-01 00:00:00") = 1970, year("1970-01-01") = 1970
int	month(string date)	It returns the month part of a date or a timestamp string: month("1970-11-01 00:00:00") = 11, month("1970-11-01") = 11
int	day(string date)	It returns the day part of a date or a timestamp string: day("1970-11-01 00:00:00") = 1, day("1970-11-01") = 1
string	get_json_object(string json_string, string path)	It extracts json object from a json string based on json path specified, and returns json string of the extracted json object. It returns NULL if the input json string is invalid.

```
hive> SELECT UPPER(tenda) FROM compras WHERE venda >= 1000;
```

Tipo devuelto	Función	Descripción
BIGINT	count(*), count(expr), count(DISTINCT expr[, expr...])	count(*) - Returns the total number of retrieved rows, including rows containing NULL values. count(expr) - Returns the number of rows for which the supplied expression is non-NULL. count(DISTINCT expr[, expr]) - Returns the number of rows for which the supplied expression(s) are unique and non-NULL. Execution of this can be optimized with hive.optimize.distinct.rewrite
DOUBLE	sum(col), sum(DISTINCT col)	Returns the sum of the elements in the group or the sum of the distinct values of the column in the group.
DOUBLE	avg(col), avg(DISTINCT col)	Returns the average of the elements in the group or the average of the distinct values of the column in the group.
DOUBLE	min(col)	Returns the minimum of the column in the group.
DOUBLE	max(col)	Returns the maximum value of the column in the group.
DOUBLE	variance(col), var_pop(col)	Returns the variance of a numeric column in the group.
DOUBLE	var_samp(col)	Returns the unbiased sample variance of a numeric column in the group.
DOUBLE	stddev_pop(col)	Returns the standard deviation of a numeric column in the group.
DOUBLE	stddev_samp(col)	Returns the unbiased sample standard deviation of a numeric column in the group.
DOUBLE	covar_pop(col1, col2)	Returns the population covariance of a pair of numeric columns in the group.
DOUBLE	covar_samp(col1, col2)	Returns the sample covariance of a pair of a numeric columns in the group.
DOUBLE	corr(col1, col2)	Returns the Pearson coefficient of correlation of a pair of a numeric columns in the group.
DOUBLE	percentile(BIGINT col, p)	Returns the exact pth percentile of a column in the group (does not work with floating point types). p must be between 0 and 1. NOTE: A true percentile can only be computed for integer values. Use PERCENTILE_APPROX if your input is non-integral.
array<double>	percentile(BIGINT col, array(p1 [, p2]...))	Returns the exact percentiles p1, p2, ... of a column in the group (does not work with floating point types). pi must be between 0 and 1. NOTE: A true percentile can only be computed for integer values. Use PERCENTILE_APPROX if your input is non-integral.
DOUBLE	percentile_approx(DOUBLE col, p [, B])	Returns an approximate pth percentile of a numeric column (including floating point types) in the group. The B parameter controls approximation accuracy at the cost of memory. Higher values yield better approximations, and the default is 10,000. When the number of distinct values in col is smaller than B, this gives an exact percentile value.
array<double>	percentile_approx(DOUBLE col, array(p1 [, p2]...) [, B])	Same as above, but accepts and returns an array of percentile values instead of a single one.
double	regr_avgx(independent, dependent)	Equivalent to avg(dependent).
double	regr_avgy(independent, dependent)	Equivalent to avg(independent).
double	regr_count(independent, dependent)	Returns the number of non-null pairs used to fit the linear regression line.
double	regr_intercept(independent, dependent)	Returns the y-intercept of the linear regression line, i.e. the value of b in the equation dependent = a * independent + b.
double	regr_r2(independent, dependent)	Returns the coefficient of determination for the regression.
double	regr_slope(independent, dependent)	Returns the slope of the linear regression line, i.e. the value of a in the equation dependent = independent * a + b.
double	regr_sxx(independent, dependent)	Equivalent to regr_count(independent, dependent) * var_pop(dependent).
double	regr_sxy(independent, dependent)	Equivalent to regr_count(independent, dependent) * covar_pop(independent, dependent).
double	regr_syy(independent, dependent)	Equivalent to regr_count(independent, dependent) * var_pop(independent).
array<struct {x,y}>	histogram_numeric(col, b)	Computes a histogram of a numeric column in the group using b non-uniformly spaced bins. The output is an array of size b of double-valued (x,y) coordinates that represent the bin centers and heights
array	collect_set(col)	Returns a set of objects with duplicate elements eliminated.
array	collect_list(col)	Returns a list of objects with duplicates.
INTEGER	ntile(INTEGER x)	Divides an ordered partition into x groups called buckets and assigns a bucket number to each row in the partition. This allows easy calculation of tertiles, quartiles, deciles, percentiles and other common summary statistics.

```
hive> SELECT tenda, venda FROM compras WHERE venda > (SELECT AVG(venda) FROM compras);
```

+xemplo 2' Consultas Hive / (S+ (+C) 1%D+% * 3

A cláusula ORDER BY permite ordear-lo resultado dunha consulta en base á orde ascendente ou descendente dunha ou varias columnas.

Sintaxe:

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...  
FROM table_reference  
[WHERE where_condition]  
[GROUP BY col_list]  
[HAVING having_condition]  
[ORDER BY col_list [(ASC | DESC)]]  
[LIMIT number]
```

Máis información: <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+SortBy>

- Obte-las vendas ordeadas polos seus importes, de maior a menor.

```
hive> SELECT * FROM compras ORDER BY venda DESC;
```

+xemplo 4' Consultas Hive / (S+ (+C) 5%16P * 3

A cláusula GROUP BY permite agrupa-los rexistros por conxuntos de resultados, en base ó valor dunha ou varias columnas.

Sintaxe:

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...  
FROM table_reference  
[WHERE where_condition]  
[GROUP BY col_list]  
[HAVING having_condition]  
[ORDER BY col_list [(ASC | DESC)]]  
[LIMIT number]
```

Máis información: <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+GroupBy>

- Obte-lo número de vendas e o importes totais das mesmas, para cada tenda.

```
hive> SELECT tenda, COUNT(*), SUM(venda) FROM compras GROUP BY tenda;
```

+xemplo 7' Consultas Hive / (S+ (+C) 8119

A cláusula JOIN úsase para combinar campos de 2 táboas que teñan valores en común. Serve par combinar tuplas de 2 ou máis táboas na base de datos.

Sintaxe:

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[HAVING having_condition]
[ORDER BY col_list [(ASC | DESC)]]
[LIMIT number]
```

Onde table=reference ó referirse a join=table:

join_table:

```
table_reference [INNER] JOIN table_factor [join_condition]
| table_reference {LEFT|RIGHT|FULL} [OUTER] JOIN table_reference join_condition
| table_reference LEFT SEMI JOIN table_reference join_condition
| table_reference CROSS JOIN table_reference [join_condition]
```

table_reference:

```
table_factor
| join_table
```

table_factor:

```
tbl_name [alias]
| table_subquery alias
| ( table_references )
```

join_condition:

```
ON expression
```

Hai 4 diferentes tipos de JOIN:

- JOIN (equivalente ó INNER JOIN de SQL)
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

De seguido van uns exemplos descritivos, partindo das seguintes táboas

customers:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00

7	Muffy	24	Indore	10000.00
---	-------	----	--------	----------

orders:

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-08 00:00:00	3	3000
100	2009-10-08 00:00:00	3	1500
101	2009-11-20 00:00:00	2	1560
103	2008-05-20 00:00:00	4	2060

- **JOIN**

Combina as ringleiras de ambas táboas e devolve os rexistros correspondentes á intersección de ambas segundo as chaves primarias e secundarias delas.

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

ID	NAME	AGE	AMOUNT
3	kaushik	23	3000
3	kaushik	23	1500
2	Khilan	25	1560
4	Chaitali	25	2060

- **LEFT OUTER JOIN**

Devolve tódalas ringleiras da táboa da esquerda, mesmo se non hai coincidencia de chave coa táboa da dereita. Isto significa, que se non hai ningún rexistro que emparelle coa táboa da dereita, o JOIN aínda devolve unha ringleira no resultado, mais con NULL en cada columna da táboa dereita.

En definitiva, un LEFT JOIN devolve tódolos valores da táboa esquerda, máis os valores emparellados da táboa dereita, ou NULL en caso de non ter parella.

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

ID	NAME	AMOUNT	DATE
1	Ramesh	NULL	NULL
2	Khilan	1560	2009-11-20 00:00:00
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00
5	Hardik	NULL	NULL
6	Komal	NULL	NULL
7	Muffy	NULL	NULL

+-----+-----+-----+-----+-----+-----+

- **RIGHT OUTER JOIN**

Devolve tódalas ringleiras da táboa dereita, mesmo se non hai ningunha coincidencia na táboa esquerda. Se a cláusula ON emparella 0 (zero) rexistros na táboa esquerda, o JOIN aínda devolve unha ringleira no resultado, mais con NULL en cada columna da táboa esquerda.

En definitiva, un RIGHT JOIN devolve tódolos valores da táboa dereita, máis os valores emparellados da táboa esquerda, ou NULL en caso de non ter parella.

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

ID	NAME	AMOUNT	DATE
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
2	Khilan	1560	2009-11-20 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00

- **FULL OUTER JOIN**

Combina as ringleiras de ambas táboas, que cumpren a condición JOIN. A táboa resultante contén tódolos rexistros de ámba-las táboas, ou enche de valores NULL, as columnas correspondentes de cada lado para non perder emparellamentos.

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

ID	NAME	AMOUNT	DATE
1	Ramesh	NULL	NULL
2	Khilan	1560	2009-11-20 00:00:00
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00
5	Hardik	NULL	NULL
6	Komal	NULL	NULL
7	Muffy	NULL	NULL
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
2	Khilan	1560	2009-11-20 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00

Máis información:

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Joins#LanguageManualJoins-JoinOptimization>

- Obte-lo número de puntos promocionais obtidos, en cada compra.

Para facer un exemplo rápido, usaremos-la sentenza JOIN para engadi-los puntos promocionais obtidos para cada compra, en base ó medio de pago utilizado. Para elo necesitamos primeiro cargar no sistema o ficheiro medios=pago.tsv adxunto.

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS medios_pago
(
  abreviatura STRING,
  nome_ampliado STRING,
  puntos_promocionais INTEGER
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE;
```

```
hive> LOAD DATA INPATH '/home/hduser/Descargas/medios_pago.tsv' INTO TABLE
medios_pago;
```

```
hive> SELECT c.*,m.puntos_promocionais FROM compras AS c LEFT OUTER JOIN
medios_pago AS m ON (c.pago = m.abreviatura);
```

Obtén o número de puntos promocionais obtidos en total, por cada tenda, ordenados polas tendas con máis a menos puntos.

+ exemplo : ' Saída output de Hive

Podemos escribi-los resultados a HDFS coa seguinte sintaxe:

```
INSERT OVERWRITE [LOCAL] DIRECTORY directory1
[ROW FORMAT row_format] [STORED AS file_format]
SELECT ... FROM ...
```

Formato de saída:

```
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]
```

Por exemplo, para escribir a disco:

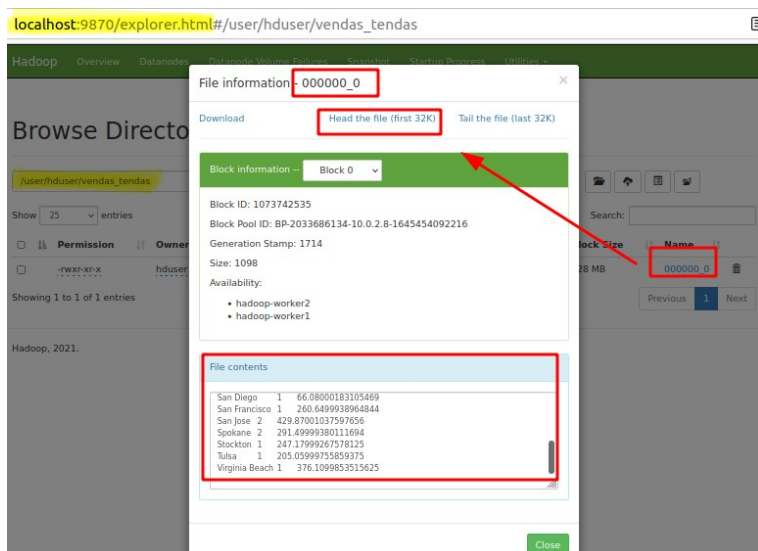
```
INSERT OVERWRITE LOCAL DIRECTORY '/home/hduser/YourTableDir'
SELECT * FROM table WHERE ...;
```

Podemos especificar algunhas cuestións como os separadores:

```
INSERT OVERWRITE LOCAL DIRECTORY '/home/hduser/YourTableDir'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
SELECT * FROM table WHERE ...;
```

- Gardar no sistema hdfs o resultado da consulta anterior do número de vendas e o importes totais das mesmas, para cada tenda.

```
hive> INSERT OVERWRITE DIRECTORY '/user/hduser/vendas_tendas' ROW FORMAT
DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE SELECT tenda, COUNT(*),
SUM(venda) FROM compras GROUP BY tenda;
```



Para ve-lo contido da táboa pódese usa-lo comando hdfs:

```
$ hdfs dfs -cat /user/hduser/vendas_tendas/000000_0
```

+xercicio

Escribe a disco a saída das seguintes consultas, sobre o ficheiro purchases.txt:

1. Guarda a HDFS os datos das vendas de xoguetes con valores maiores de 200. Comproba o ficheiro de saída.
2. Repite a consulta pero coa saída separada por ';'.
3. Guarda a disco local os datos das vendas pagadas con Visa ou MasterCard.

+=+>P(1 ?' Inserción de datos en t boas

Os resultados das consultas pódense escribir no sistema HDFS.

Con Hive, temos dous tipos de comandos de inserción para cargar datos en táboas e particións:

- `INSERT INTO` : úsase para engadir datos a unha táboa ou partición, sen modifica-los datos xa existentes.

```
INSERT INTO destino SELECT col1, col2 FROM orixe;
```

- `INSERT OVERWRITE` : úsase para reempraza-lo datos existentes na táboa ou partición, baleirándose previamente antes de volver a rechearse coa inserción de novos rexistros.

```
INSERT OVERWRITE destino SELECT col1, col2 FROM orixe;
```

Por defecto o destino é o sistema `HDFS` :

```
INSERT OVERWRITE DIRECTORY 'ventas_x_tenda' SELECT tenda, COUNT(*), SUM(venda)  
FROM compras GROUP BY tenda;
```

Pero pódese gardar no sistema de arquivos `local` usando a cláusula `LOCAL DIRECTORY` :

```
INSERT OVERWRITE LOCAL DIRECTORY '/home/hduser/ventas_x_tenda' SELECT tenda,  
COUNT(*), SUM(venda) FROM compras GROUP BY tenda;
```

Tamén pódense especificar cousas como os **separadores** ou o **formato do arquivo** de saída:

```
INSERT OVERWRITE LOCAL DIRECTORY '/home/hduser/dirSaidaTaboa'  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY '\t'  
STORED AS TEXTFILE  
SELECT * FROM table WHERE ...;
```

Por exemplo, para que o arquivo estea en formato de texto co “punto e coma” como separador de campos:

```
INSERT OVERWRITE DIRECTORY 'ventas_x_tenda' ROW FORMAT DELIMITED FIELDS  
TERMINATED BY ';' SELECT tenda, COUNT(*), SUM(venda) FROM compras GROUP BY  
tenda;
```

Caso distinto é o de ter un ficheiro que se queira cargar na táboa, para o que se usa o comando `LOAD` anteriormente visto de `Hive` .

Asemade, diferéncianse 3 tipos de forma de carga de datos `INSERT`, dependendo de se se fai nunha táboa de Hive existente, nunha táboa de Hive usando unha sintaxe similar ó SQL estándar, ou nun directorio HDFS directamente:

1. *5? . <96 5?6D Hive tables from Bueries*: para insertar datos calculados ou filtrados por unha consulta SQL nunha táboa de Hive existente. Por exemplo, para que os resultados da consulta SELECT se insiran na táboa taboa1:

```
INSERT INTO taboa1
SELECT col1, col2 FROM taboa2 WHERE col3 > 100;
```

2. *5? . <96 5?6D Hive tables from . E4*: para inserir datos de forma atómica directamente nunha táboa de Hive utilizando unha sentenza SQL de base de datos estándar. Por exemplo, para inserir valores específicos nas columnas col1 e col da táboa taboa1:

```
INSERT INTO table1 (col1, col2)
VALUES (valor1, valor2), (valor3, valor4);
```

3. *5? . <96 5?6D directories from Bueries*: para inseri-los resultados dunha consulta SQL nun directorio de HDFS (Hadoop Distributed File System). Por exemplo, para que os resultados da consulta SELECT se escriban no directorio especificado en HDFS:

```
INSERT OVERWRITE DIRECTORY '/user/hduser/saida'
SELECT col1, col2 FROM taboa1 WHERE col3 = 'valor';
```

Tamén existe a posibilidade de inserir datos en múltiples táboas á vez:

- No caso de necesitar inseri-los datos en varias táboas á vez, é mellor usa-la sentenza , xa que o rendemento é mellor ó só realizarse unha única lectura.

```
FROM orixe
INSERT OVERWRITE TABLE destino1
SELECT col1, col2
INSERT OVERWRITE TABLE destino2
SELECT col1, col3
```

Exemplos:

Por exemplo, para inserir un ou varios rexistros nunha táboa, coa sentenza

:

```
INSERT INTO compras VALUES
('2023-05-06', '09:00', 'Pittsburgh', 'Pet Supplies', 91.80, 'Discover'),
('2023-05-06', '09:10', 'Pittsburgh', 'Pet Supplies', 1.99, 'Visa');
```

Para inserir datos en columnas seleccionadas (nas restantes columnas insírese o valor nulo), coa sentenza

:

```
INSERT INTO compras(data,tenda,venda) VALUES
('2023-05-06', 'San Diego', 101.10),
('2023-05-06', 'San Diego', 101.99);
```

Tamén pódese usar unha sentenza para inseri-lo seu resultado nunha táboa.

```
CREATE TABLE compras_tmp LIKE compras;
INSERT INTO compras_tmp SELECT * FROM compras;
```

Para insertar datos nunha partición hai que usa-la cláusula . Por exemplo, supoñendo que houbera unha columna *3P* como chave de partición (esa columna ten que se-la última):

```
INSERT INTO compras PARTITION(cp=36206) VALUES
('2023-05-06', '09:00', 'Pittsburgh', 'Pet Supplies', 91.80, 'Discover');

INSERT INTO compras PARTITION(cp) VALUES
('2023-05-06', '09:00', 'Pittsburgh', 'Pet Supplies', 91.80, 'Discover', 36206);
```

Un exemplo de como borraría tódolos datos da táboa Hive e inserta unha ringleira específica sería:

```
INSERT OVERWRITE TABLE compras VALUES
('2023-05-07', 'San Diego', 102.20);
```

Para crear con un par de táboas coa mesma estrutura da táboa de compras, pero que cada unha delas garde os datos correspondentes a dunha determinada localidade:

```
CREATE TABLE compras_pittsburgh LIKE compras;
CREATE TABLE compras_sandiego LIKE compras;
```

E, de seguido encheríanse as táboas cos valores correspondentes ás súas localidades:

```
FROM compras
INSERT OVERWRITE TABLE compras_pittsburgh
  SELECT data, hora, tenda, categoria, venda, pago WHERE tenda = "Pittsburgh"
INSERT OVERWRITE TABLE compras_sandiego
  SELECT data, hora, tenda, categoria, venda, pago WHERE tenda = "San Diego";
```


+xercicio de aplicación de consultas Hive

Utiliza HiveQL para dar resposta ás seguintes preguntas sobre purc&ases.txt:

1. Cal é o número total de vendas?
2. Cal é o importe total de vendas?
3. Cal é a suma total e número de vendas por tenda?
4. Cal é a venda mínima por tipo de pago?
5. Cal é a media das vendas por tenda?
6. Que tendas están por riba da media en canto a vendas?
7. Cal é o tipo de pago máis utilizado?
8. Datos das 25 vendas máis altas (data, tenda, venda, categoría)
9. Listaxe das tendas con maior número de vendas en cada categoría
10. Bonus: mellora da anterior consulta: limitar o número de resultados por grupo usando a función RANK (trátase do típico problema do TOP N por grupos). Axuda: <https://learnsql.com/blog/how-to-rank-rows-in-sql/>
11. Garda a HDFS os datos das vendas de xoguetes con valores maiores de 200 (verifica o ficheiro de saída)
12. Repite a consulta pero coa saída separada por ';'.
13. Garda a disco local os datos das vendas pagadas con Visa ou MasterCard con saída separada por tabuladores.

Modificar datos en táboas

Ademais de engadir datos, tamén é posible realizar operacións `DELETE` e `UPDATE` sobre as táboas dunha base de datos.

HDFS non se deseñou pensando nas modificacións de arquivos, polo que os cambios resultantes das insercións, modificacións e borrados almacénanse en `hdfs` (unha especie de rexistro incremental dos cambios). Por cada transacción, créase un conxunto de arquivos delta que altera a táboa (ou partición). Os ficheiros delta fúndense periodicamente cos ficheiros base das táboas mediante traballos `MapReduce` que o `metastore` executa en segundo plano.

Para poder modificar ou borrar os datos, Hive necesita traballar nun contexto transaccional, polo que necesitamos activa-las seguintes variables:

```
set hive.support.concurrency=true;
set hive.enforce.bucketing=true;
set hive.exec.dynamic.partition.mode=nonstrict;
set hive.txn.manager=org.apache.hadoop.hive.q1.lockmgr.DbTxnManager;
```

No seguinte exemplo, unha vez configuradas as variables, créase unha táboa de clientes co formato ORC, organizando os datos mediante buckets e finalmente indícase mediante `INDEX` que se trata dunha táboa transaccional (para que permita as operacións a nivel de ringleira de `DELETE` e `UPDATE`).

```
CREATE TABLE clientes
(
  idCliente INT,
  nome STRING,
  enderezo STRING,
)
CLUSTERED BY (idCliente) INTO 4 BUCKETS
STORED AS ORC
TBLPROPERTIES ('transactional' = 'true');
```

Unha vez creada a táboa cárganse os datos a partir da de `customers`:

```
INSERT INTO clientes
SELECT id, name, address FROM customers;
```

E unha vez cargados os datos modifícase o enderezo “AP” por “Madhya Pradesh”:

```
UPDATE clientes SET address="Madhya Pradesh" WHERE address="MP";
```

Fonte: <https://www.sparkcodehub.com/hive/unleashing-transactional-tables-in-hive>

Tutoriais de Hive:

- <https://cwiki.apache.org/confluence/display/Hive>

Máis información:

- <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DML>