EBook Gratis

APRENDIZAJE hive

Free unaffiliated eBook created from
Stack Overflow contributors.

#hive

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: hive

It is an unofficial and free hive ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official hive.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con la colmena

Observaciones

- Hive es una herramienta de almacenamiento de datos construida sobre Hadoop .
- Proporciona un lenguaje similar a SQL para consultar datos.
- Podemos ejecutar casi todas las consultas SQL en Hive, la única diferencia es que ejecuta un trabajo de reducción de mapa en el backend para obtener los resultados de Hadoop Cluster. Debido a esto, Hive a veces toma más tiempo para obtener el conjunto de resultados.

Examples

Ejemplo de conteo de palabras en Hive

Archivo de documentos (archivo de entrada)

María tenía un corderito

su vellón era blanco como la nieve

y por todas partes que fue María

el cordero estaba seguro de ir

Consulta de la colmena

```
CREATE TABLE FILES (line STRING);

LOAD DATA INPATH 'docs' OVERWRITE INTO TABLE FILES;

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

Salida de la tabla word_counts en Hive

Maria, 2

tenido, 1

a, 1

pequeño 1

cordero, 2

es, 1

vellón, 1

era, 2

blanco, 1

como, 1

nieve 1

y 1

en todas partes 1

que 1

fue, 1

el, 1

claro que 1

a 1

ve, 1

Instalación de Hive (linux)

Comience por descargar la última versión estable de https://hive.apache.org/downloads.html

-> Ahora untar el archivo con

$ tar -xvf hive-2.xy-bin.tar.gz

-> Crear un directorio en el / usr / local / con

$ sudo mkdir / usr / local / hive

-> Mueve el archivo a root con

$ mv ~ / Downloads / hive-2.xy / usr / local / hive

-> Editar variables de entorno para hadoop y hive en .bashrc

$ gedit ~ / .bashrc

Me gusta esto

export HIVE_HOME = / usr / local / hive / apache-hive-2.0.1-bin /

export PATH = $ PATH: $ HIVE_HOME / bin

export CLASSPATH = $ CLASSPATH: / usr / local / Hadoop / lib / * :.

export CLASSPATH = $ CLASSPATH: /usr/local/hive/apache-hive-2.0.1-bin/lib/* :.

-> Ahora, inicie hadoop si aún no se está ejecutando. Y asegúrese de que se está ejecutando y no está en modo seguro.

$ hadoop fs -mkdir / user / hive / warehouse

El directorio "almacén" es la ubicación para almacenar la tabla o los datos relacionados con la colmena.

$ hadoop fs -mkdir / tmp

El directorio temporal "tmp" es la ubicación temporal para almacenar el resultado intermedio del procesamiento.

-> Establecer permisos para leer / escribir en esas carpetas.

$ hadoop fs -chmod g + w / user / hive / warehouse

$ hadoop fs -chmod g + w / usuario / tmp

-> Ahora inicia HIVE con este comando en la consola

$ colmena

Instalación de Hive con Metastore Externo en Linux

Pre-requisitos:

1. Java 7
2. Hadoop (Consulte aquí para la instalación de Hadoop)
3. Servidor Mysql y Cliente

Instalación:

Paso 1: descarga el último archivo de Hive desde la página de descargas .

Paso 2: Extraiga el tarball descargado (***Supuesto:*** *El tarball se descarga en $ HOME*)

```
tar -xvf /home/username/apache-hive-x.y.z-bin.tar.gz
```

Paso 3: Actualice el archivo de entorno (`~/.bashrc`)

```
export HIVE_HOME=/home/username/apache-hive-x.y.z-bin
export PATH=$HIVE_HOME/bin:$PATH
```

fuente el archivo para establecer las nuevas variables de entorno.

```
source ~/.bashrc
```

Paso 4: Descargue el conector JDBC para mysql desde aquí y extráigalo.

```
tar -xvf mysql-connector-java-a.b.c.tar.gz
```

El directorio extraído contiene el archivo jar del conector `mysql-connector-java-abcjar` . Copiarlo en el `lib` de `$HIVE_HOME`

```
cp mysql-connector-java-a.b.c.jar $HIVE_HOME/lib/
```

Configuración:

Cree el archivo de configuración de `hive-site.xml` en el `$HIVE_HOME/conf/` y agregue las siguientes propiedades relacionadas con el metastore.

```
<configuration>
   <property>
      <name>javax.jdo.option.ConnectionURL</name>
      <value>jdbc:mysql://localhost/hive_meta</value>
      <description>JDBC connect string for a JDBC metastore</description>
   </property>

   <property>
      <name>javax.jdo.option.ConnectionDriverName</name>
      <value>com.mysql.jdbc.Driver</value>
      <description>Driver class name for a JDBC metastore</description>
   </property>

   <property>
       <name>javax.jdo.option.ConnectionUserName</name>
       <value>mysqluser</value>
       <description>username to use against metastore database</description>
   </property>

   <property>
       <name>javax.jdo.option.ConnectionPassword</name>
       <value>mysqlpass</value>
       <description>password to use against metastore database</description>
   </property>

   <property>
       <name>datanucleus.autoCreateSchema</name>
       <value>false</value>
   </property>

   <property>
       <name>datanucleus.fixedDatastore</name>
       <value>true</value>
   </property>
</configuration>
```

Actualice los valores de MySQL "nombre de usuario" y "contraseña" en consecuencia en las propiedades.

Crear el esquema de Metastore:

Los scripts de esquema de metastore están disponibles en `$HIVE_HOME/scripts/metastore/upgrade/mysql/`

Inicie sesión en MySQL y obtenga el esquema,

```
mysql -u username -ppassword

mysql> create database hive_meta;
mysql> use hive_meta;
mysql> source hive-schema-x.y.z.mysql.sql;
mysql> exit;
```

Metastore de partida:

```
hive --service metastore
```

Para ejecutarlo en segundo plano,

```
nohup hive --service metastore &
```

Iniciar HiveServer2: (usar si es necesario)

```
hiveserver2
```

Para ejecutarlo en segundo plano,

```
nohup hiveserver2 metastore &
```

Nota: estos ejecutables están disponibles en `$HIVE_HOME/bin/`

Conectar:

Utilizar cualquiera de `hive` , `beeline` o Hue a conectar con la colmena.

Hive CLI está en desuso, se recomienda usar Beeline o Hue.

Configuraciones adicionales para Hue:

Actualice este valor en `$HUE_HOME/desktop/conf/hue.ini`

```
[beeswax]
    hive_conf_dir=/home/username/apache-hive-x.y.z-bin/conf
```

Lea Empezando con la colmena en línea: https://riptutorial.com/es/hive/topic/1099/empezando-con-la-colmena

Capítulo 2: Creación de tablas de Hive a través de Sqoop

Introducción

Si tenemos una meta-tienda de Hive asociada con nuestro clúster HDFS, Sqoop puede importar los datos a Hive generando y ejecutando una sentencia CREATE TABLE para definir el diseño de los datos en Hive. Importar datos en Hive es tan simple como agregar la opción --hive-import a su línea de comandos de Sqoop.

Observaciones

Importar datos directamente desde RDBMS a HIVE puede resolver mucho tiempo. También podemos ejecutar una consulta de forma libre (una combinación o alguna consulta simple) y rellenarla en una tabla de nuestra elección directamente en Hive.

--hive-import le dice a Sqoop que el destino final es Hive y no HDFS.

- La opción de tabla activa ayuda a importar los datos a la tabla en la sección elegida por nosotros, de lo contrario, se nombrará como la tabla de origen que se importará desde RDBMS.

Examples

Importación de Hive con el nombre de la tabla de Destino en Hive

```
$ sqoop import --connect jdbc:mysql://10.0.0.100/hadooptest
--username hadoopuser -P
--table table_name --hive-import --hive-table hive_table_name
```

Lea Creación de tablas de Hive a través de Sqoop en línea:
https://riptutorial.com/es/hive/topic/10685/creacion-de-tablas-de-hive-a-traves-de-sqoop

Capítulo 3: Crear base de datos y declaración de tabla

Sintaxis

- CREAR [TEMPORAL] [EXTERNO] TABLA [SI NO EXISTE] [db_name.] Table_name

  [(col_name data_type [COMMENT col_comment], ...)] [COMMENT table_comment] [PARTITIONED BY (col_name data_type [COMMENT col_comment],))] [CLUSTERED BY (col_name, col_name, ...) [SORTED BY (col_name [ASC | DESC], ...)] INTO num_buckets BUCKETS] [SKEWED BY (col_name, col_name, ...) - (Nota: Disponible en Hive 0.10.0 y posterior)] ON ((col_value, col_value, ...), (col_value, col_value, ...), ...) [ALMACENADO COMO DIRECTORIOS] [[FORMATO DE FILA row_format] [STORED AS file_format] | ALMACENADO POR 'storage.handler.class.name' [CON SERDEPROPERTIES (...)]] [LOCATION hdfs_path] [TBLPROPERTIES (property_name = property_value, ...)] [AS select_statement];

- CREAR LA TABLA [TEMPORAL] [EXTERNA] [SI NO EXISTE] [nombre_bd.] Nombre_tabla LIKE existing_table_or_view_name [LOCATION hdfs_path];

- data_type: primitive_type, array_type, map_type, struct_type, union_type

- tipo primitivo: TINYINT, SMALLINT, INT, BIGINT, BOOLEAN, FLOTADOR, DOBLE, STRING, BINARY, TIMESTAMP, DECIMAL, DECIMAL (precisión, escala), DATE, VARCHAR, CHAR

- array_type: ARRAY <data_type>

- map_type: MAP <primitive_type, data_type>

- struct_type: STRUCT <col_name: data_type [COMMENT col_comment], ...>

- union_type: UNIONTYPE <data_type, data_type, ...>

- row_format: DELIMITED [CAMPOS TERMINADOS POR char [ESCAPED BY char]] [COLECCIÓN ARTÍCULOS TERMINADOS POR char] [MAP KEYS TERMINATED BY char] [LÍNEAS TERMINADAS POR char] [NULL DEFINED AS char]
  , SERDE serde_name [CON SERDEPROPERTIES (property_name = property_value, property_name = property_value, ...)]

- file_format:: SEQUENCEFILE, TEXTFILE, RCFILE, ORC, PARQUET, AVRO, INPUTFORMAT input_format_classname OUTPUTFORMAT output_format_classname

- CREATE (DATABASE | SCHEMA) [SI NO EXISTE] database_name [COMMENT database_comment] [LOCATION hdfs_path] [WITH DBPROPERTIES (property_name = property_value, ...)];

Observaciones

Cuando se trabaja con tablas y bases de datos en HIVE. Los siguientes puntos pueden ser útiles.

- Podemos cambiar la base de datos utilizando la `use database;` mando
- Para conocer la base de datos de trabajo actual podemos obtener utilizando `SELECT current_database()`
- Para ver el DDL utilizado para crear una declaración de tabla, podemos usar `SHOW CREATE TABLE tablename`
- Para ver todas las columnas de la tabla, utilice `DESCRIBE tablename` para mostrar detalles extendidos, como la ubicación que se usa y otros `DESCRIBE FORMATTED tablename` . DESCRIBIR también se puede abreviar como DESC.

Examples

Crear mesa

Creación de una tabla **administrada** con partición y almacenada como un archivo de secuencia. Se asume que el formato de datos en los archivos está delimitado por el campo por `Ctrl-A (^A)` y por la línea nueva delimitado por filas. La siguiente tabla se crea en el directorio del almacén de Hive especificado en valor para la clave `hive.metastore.warehouse.dir` en el archivo de configuración de Hive `hive-site.xml` .

```
CREATE TABLE view
(time INT,
id BIGINT,
url STRING,
referrer_url STRING,
add STRING COMMENT 'IP of the User')
COMMENT 'This is view table'
PARTITIONED BY(date STRING, region STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\001'
STORED AS SEQUENCEFILE;
```

Creación de una tabla **externa** con particiones y almacenada como un archivo de secuencia. Se supone que el formato de datos en los archivos está delimitado por el campo por `ctrl-A` y por la línea nueva delimitado por filas. La siguiente tabla se crea en la ubicación especificada y es útil cuando ya tenemos datos. Una de las ventajas de usar una tabla externa es que podemos eliminar la tabla sin eliminar los datos. Por ejemplo, si creamos una tabla y nos damos cuenta de que el esquema es incorrecto, podemos dejar la tabla de forma segura y recrear con el nuevo esquema sin preocuparnos por los datos. Otra ventaja es que si estamos usando otras herramientas como pig en los mismos archivos, Podemos seguir usándolos incluso después de eliminar la tabla.

```
CREATE EXTERNAL TABLE view
(time INT,
id BIGINT,
url STRING,
referrer_url STRING,
```

```
add STRING COMMENT 'IP of the User')
COMMENT 'This is view table'
PARTITIONED BY(date STRING, region STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\001'
STORED AS SEQUENCEFILE
LOCATION '<hdfs_location>';
```

Al crear una tabla utilizando la consulta de selección y **rellenando los** resultados de la consulta, estas declaraciones se conocen como **CTAS (Crear tabla como selección)** .

Hay dos partes en CTAS, la parte SELECT puede ser cualquier instrucción SELECT admitida por HiveQL. La parte CREAR del CTAS toma el esquema resultante de la parte SELECCIONAR y crea la tabla de destino con otras propiedades de la tabla como el SerDe y el formato de almacenamiento.

CTAS tiene estas restricciones:

- La tabla de destino no puede ser una tabla particionada.
- La tabla de destino no puede ser una tabla externa.
- La tabla de destino no puede ser una tabla de agrupación de listas.

```
CREATE TABLE new_key_value_store
ROW FORMAT SERDE "org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe"
STORED AS RCFile
AS
SELECT * FROM page_view
SORT BY url, add;
```

Crear tabla como:

La **forma LIKE de CREATE TABLE le** permite copiar una definición de tabla existente exactamente (sin copiar sus datos). A diferencia de CTAS, la siguiente declaración crea una nueva tabla cuya definición coincide exactamente con la tabla existente en todos los detalles, excepto el nombre de la tabla. La nueva tabla no contiene filas.

```
CREATE TABLE empty_page_views
LIKE page_views;
```

Crear base de datos

Creación de una base de datos en una ubicación particular. Si no especificamos ninguna ubicación para la base de datos, se creará en el directorio del almacén.

```
CREATE DATABASE IF NOT EXISTS db_name
COMMENT 'TEST DATABASE'
LOCATION /PATH/HDFS/DATABASE/;
```

Creación de tablas Hive ACID.

Las tablas ACID son compatibles desde la versión hive 0.14. Debajo de la tabla es compatible con ACTUALIZAR / BORRAR / INSERTAR

Debajo de los cambios de configuración requeridos en hive-site.xml.

```
hive.support.concurrency = true
hive.enforce.bucketing = true
hive.exec.dynamic.partition.mode = nonstrict
hive.txn.manager =org.apache.hadoop.hive.ql.lockmgr.DbTxnManager
hive.compactor.initiator.on = true
hive.compactor.worker.threads = 1
```

Actualmente solo el archivo orc es compatible con el formato.

Tabla de crear declaración.

```
create table Sample_Table(
col1 Int,
col2 String,
col3 String)
clustered by (col3) into 3 buckets
stored as orc
TBLPROPERTIES ('transactional'='true');
```

Integración HIVE_HBASE

La integración Hive-Hbase es compatible desde las versiones inferiores. Colmena: 0.11.0 HBase: 0.94.2 Hadoop: 0.20.2

```
CREATE TABLE hbase_hive
(id string,
 col1 string,
 col2 string,
 col3 int)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES
("hbase.columns.mapping" = ":key,cf1:col1,cf1:col2,cf1:col3")
TBLPROPERTIES ("hbase.table.name" = "hive_hbase");
```

Nota: la primera columna debe ser la columna clave.

Crear tabla utilizando las propiedades de tabla existentes.

```
CREATE TABLE new_table_name LIKE existing_table_name;
```

Lea Crear base de datos y declaración de tabla en línea:
https://riptutorial.com/es/hive/topic/3328/crear-base-de-datos-y-declaracion-de-tabla

Capítulo 4: Declaración SELECT

Sintaxis

- SELECCIONAR [TODO | DISTINCT] select_expr, select_expr, select_expr,….
- FROM table_reference
- [WHERE where_condition]
- [GRUPO POR col_list]
- [Teniendo condición]
- [ORDENAR POR col_list]
- [LIMITE n]

Examples

Seleccionar todas las filas

`SELECT` se utiliza para recuperar filas de datos de una tabla. Puede especificar qué columnas se recuperarán:

```
SELECT Name, Position
FROM Employees;
```

O simplemente use * para obtener todas las columnas:

```
SELECT *
FROM Employees;
```

Seleccionar filas específicas

Esta consulta devolverá todas las columnas de las `sales` de la tabla donde los valores en el `amount` la columna son mayores que 10 y los datos en la columna de la `region` en "US".

```
SELECT * FROM sales WHERE amount > 10 AND region = "US"
```

Puede usar *expresiones regulares* para seleccionar las columnas que desea obtener. La siguiente declaración obtendrá los datos del `name` columna y todas las columnas que comiencen con la `address` prefijo.

```
SELECT name, address.* FROM Employees
```

También puede usar la palabra clave `LIKE` (combinada con el carácter '%') para hacer coincidir las cadenas que comienzan con o terminan con una subcadena particular. La siguiente consulta devolverá todas las filas en las que la `city` la columna comience con "Nuevo"

```
SELECT name, city FROM Employees WHERE city LIKE 'New%'
```

Puede usar la palabra clave `RLIKE` para usar *expresiones regulares de* Java. La siguiente consulta devolverá filas cuyo `name` columna contiene las palabras "smith" o "son".

```
SELECT name, address FROM Employee WHERE name RLIKE '.*(smith|son).*'
```

Puede aplicar funciones a los datos devueltos. La siguiente oración devolverá todos los nombres en mayúsculas.

```
SELECT upper(name) FROM Employees
```

Puede utilizar diferentes funciones matemáticas , funciones de recopilación , funciones de conversión de tipos , funciones de fecha , funciones condicionales o funciones de cadena .

Para limitar el número de filas que se muestran en el resultado, puede usar la palabra clave `LIMIT` . La siguiente declaración devolverá sólo diez filas.

```
SELECT * FROM Employees LIMIT 10
```

Seleccionar: Proyecto columnas seleccionadas

Estructura de la tabla de muestra (por ejemplo, empleado)

Nombre de columna	Tipo de datos
CARNÉ DE IDENTIDAD	EN T
F_Nombre	CUERDA
L_Nombre	CUERDA
Teléfono	CUERDA
Dirección	CUERDA

Proyecta todas las columnas.

Usa el comodín `*` para proyectar todas las columnas. p.ej

```
Select * from Employee
```

Proyecto de columnas seleccionadas (digamos ID, Nombre)

Use el nombre de las columnas en la lista de proyecciones. p.ej

```
Select ID, Name from Employee
```

Descartar 1 columna de la lista de Proyecciones

Mostrar todas las columnas excepto 1 columna. p.ej

```
Select `(ID)?+.+` from Employee
```

Desechar columnas que coinciden con el patrón

Rechaza todas las columnas que coincidan con el patrón. por ejemplo, rechazar todas las columnas que terminan con `NAME`

```
Select `(.*NAME$)?+.+` from Employee
```

Lea Declaración SELECT en línea: https://riptutorial.com/es/hive/topic/4133/declaracion-select

Capítulo 5: Exportar datos en colmena

Examples

Característica de exportación en la colmena.

Exportando datos de la tabla de empleados a / tmp / ca_employees

INSERTAR SOBREESCRIBIR DIRECTORIO LOCAL '/ tmp / ca_employees' SELECCIONE el nombre, salario, dirección DE los empleados DONDE se.state = 'CA';

Exportación de datos de la tabla de empleados a varios directorios locales según una condición específica

La siguiente consulta muestra cómo se puede usar una construcción única para exportar datos a varios directorios en función de criterios específicos

```
FROM employees se INSERT OVERWRITE DIRECTORY '/tmp/or_employees' SELECT * WHERE se.cty = 'US'
and se.st = 'OR'
INSERT OVERWRITE DIRECTORY '/tmp/ca_employees' SELECT * WHERE se.cty = 'US' and se.st = 'CA'
INSERT OVERWRITE DIRECTORY '/tmp/il_employees' SELECT * WHERE se.cty = 'US' and se.st = 'IL';
```

Lea Exportar datos en colmena en línea: https://riptutorial.com/es/hive/topic/6530/exportar-datos-en-colmena

Capítulo 6: Formatos de archivo en HIVE

Examples

SEQUENCEFILE

Almacene los datos en SEQUENCEFILE si los datos necesitan ser comprimidos. Puede importar archivos de texto comprimidos con Gzip o Bzip2 directamente en una tabla almacenada como TextFile. La compresión se detectará automáticamente y el archivo se descomprimirá sobre la marcha durante la ejecución de la consulta.

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

ORC

El formato de archivo Optimized Row Columnar (ORC) proporciona una forma altamente eficiente de almacenar datos de Hive. Fue diseñado para superar las limitaciones de los otros formatos de archivo Hive. El uso de archivos ORC mejora el rendimiento cuando Hive lee, escribe y procesa datos. El archivo ORC puede contener índices ligeros y filtros de floración.

Consulte: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC

ORC es un formato recomendado para almacenar datos dentro de la distribución de HortonWorks.

```
CREATE TABLE tab_orc (col1 STRING,
                      col2 STRING,
                      col3 STRING)
STORED AS ORC
TBLPROPERTIES (
               "orc.compress"="SNAPPY",
               "orc.bloom.filter.columns"="col1",
               "orc.create.index" = "true"
              )
```

Para modificar una tabla para que las nuevas particiones de la tabla se almacenen como archivos ORC:

```
ALTER TABLE T SET FILEFORMAT ORC;
```

A partir de Hive 0.14, los usuarios pueden solicitar una combinación eficiente de pequeños archivos ORC al emitir un comando `CONCATENATE` en su tabla o partición. Los archivos se fusionarán a nivel de banda sin reserialización.

```
ALTER TABLE T [PARTITION partition_spec] CONCATENATE;
```

PARQUET

Parquet en formato de almacenamiento columnar en Hive 0.13.0 y posteriores. Parquet está construido desde cero con complejas estructuras de datos anidadas en mente, y utiliza el algoritmo de fragmentación y ensamblaje de registros descrito en el documento de Dremel. Creemos que este enfoque es superior al simple aplanamiento de espacios de nombres anidados.

Parquet está construido para soportar esquemas de compresión y codificación muy eficientes. Varios proyectos han demostrado el impacto en el rendimiento de la aplicación de la compresión correcta y el esquema de codificación a los datos. Parquet permite que los esquemas de compresión se especifiquen en un nivel por columna, y está preparado para el futuro para permitir agregar más codificaciones a medida que se inventan e implementan.

Se recomienda parquet Formato de archivo con tablas de impala en las distribuciones de Cloudera.

Ver: http://parquet.apache.org/documentation/latest/

```
CREATE TABLE parquet_table_name (x INT, y STRING) STORED AS PARQUET;
```

AVRO

Los archivos Avro se admiten en Hive 0.14.0 y versiones posteriores.

Avro es un marco de llamada a procedimiento remoto y serialización de datos desarrollado dentro del proyecto Hadoop de Apache. Utiliza JSON para definir tipos de datos y protocolos, y serializa los datos en un formato binario compacto. Su uso principal es en Apache Hadoop, donde puede proporcionar un formato de serialización para datos persistentes y un formato de conexión para la comunicación entre los nodos de Hadoop y de los programas del cliente a los servicios de Hadoop.

Especificación del formato AVRO: https://avro.apache.org/docs/1.7.7/spec.html

```
CREATE TABLE kst
PARTITIONED BY (ds string)
STORED AS AVRO
TBLPROPERTIES (
  'avro.schema.url'='http://schema_provider/kst.avsc');
```

También podemos usar la siguiente sintaxis sin usar el archivo de esquema.

```
CREATE TABLE kst (field1 string, field2 int)
PARTITIONED BY (ds string)
STORED AS AVRO;
```

En los ejemplos anteriores, la cláusula `STORED AS AVRO` es equivalente a:

```
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'
```

Archivo de texto

TextFile es el formato de archivo predeterminado, a menos que el parámetro de configuración hive.default.fileformat tenga una configuración diferente. Podemos crear una tabla en la sección utilizando los nombres de campo en nuestro archivo de texto delimitado. Digamos, por ejemplo, que nuestro archivo csv contiene tres campos (id, nombre, salario) y queremos crear una tabla en la colmena llamada "empleados". Usaremos el siguiente código para crear la tabla en la colmena.

```
CREATE TABLE employees (id int, name string, salary double) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',';
```

Ahora podemos cargar un archivo de texto en nuestra tabla:

```
LOAD DATA LOCAL INPATH '/home/ourcsvfile.csv' OVERWRITE INTO TABLE employees;
```

Mostrando el contenido de nuestra tabla en la colmena para verificar si los datos se cargaron exitosamente:

```
SELECT * FROM employees;
```

Lea Formatos de archivo en HIVE en línea: https://riptutorial.com/es/hive/topic/4513/formatos-de-archivo-en-hive

Capítulo 7: Funciones agregadas definidas por el usuario (UDAF)

Examples

UDAF significa ejemplo

- Cree una clase Java que amplíe `org.apache.hadoop.hive.ql.exec.hive.UDAF` Cree una clase interna que implemente el `UDAFEvaluator`
- Implementar cinco métodos.
    - `init()` : este método inicializa el evaluador y restablece su estado interno. Estamos utilizando la nueva columna () en el código siguiente para indicar que aún no se han agregado valores.
    - `iterate()` : este método se llama cada vez que se agrega un nuevo valor. El evaluador debe actualizar su estado interno con el resultado de realizar la agregación (estamos haciendo la suma, ver más abajo). Devolvemos true para indicar que la entrada era válida.
    - `terminatePartial()` : este método se llama cuando Hive desea un resultado para la agregación parcial. El método debe devolver un objeto que encapsule el estado de la agregación.
    - `merge()` : este método se llama cuando Hive decide combinar una agregación parcial con otra.
    - `terminate()` : este método se llama cuando se necesita el resultado final de la agregación.

```
public class MeanUDAF extends UDAF {
// Define Logging
static final Log LOG = LogFactory.getLog(MeanUDAF.class.getName());
public static class MeanUDAFEvaluator implements UDAFEvaluator {
/**
 * Use Column class to serialize intermediate computation
 * This is our groupByColumn
 */
public static class Column {
 double sum = 0;
 int count = 0;
 }
private Column col = null;
public MeanUDAFEvaluator() {
 super();
 init();
 }
// A - Initalize evaluator - indicating that no values have been
// aggregated yet.
public void init() {
 LOG.debug("Initialize evaluator");
 col = new Column();
```

```
 }
// B- Iterate every time there is a new value to be aggregated
 public boolean iterate(double value) throws HiveException {
 LOG.debug("Iterating over each value for aggregation");
 if (col == null)
 throw new HiveException("Item is not initialized");
 col.sum = col.sum + value;
 col.count = col.count + 1;
 return true;
 }
// C - Called when Hive wants partially aggregated results.
 public Column terminatePartial() {
 LOG.debug("Return partially aggregated results");
 return col;
 }
 // D - Called when Hive decides to combine one partial aggregation with another
 public boolean merge(Column other) {
 LOG.debug("merging by combining partial aggregation");
 if(other == null) {
 return true;
 }
 col.sum += other.sum;
 col.count += other.count;
 return true;
}
 // E - Called when the final result of the aggregation needed.
 public double terminate(){
 LOG.debug("At the end of last record of the group - returning final result");
 return col.sum/col.count;
 }
 }
}


hive> CREATE TEMPORARY FUNCTION <FUNCTION NAME> AS 'JAR PATH.jar';
hive> select id, mean_udf(amount) from table group by id;
```

Lea Funciones agregadas definidas por el usuario (UDAF) en línea:
https://riptutorial.com/es/hive/topic/5137/funciones-agregadas-definidas-por-el-usuario--udaf-

Capítulo 8: Funciones de tabla definidas por el usuario (UDTF)

Examples

Ejemplo UDTF y uso

Funciones de tabla definidas por el usuario representadas por la interfaz **org.apache.hadoop.hive.ql.udf.generic.GenericUDTF** . Esta función permite generar múltiples filas y múltiples columnas para una sola entrada.

Tenemos que sobrescribir los siguientes métodos:

```
 1.we specify input and output parameters
abstract StructObjectInspector initialize(ObjectInspector[] args)
                                                throws UDFArgumentException;

 2.we process an input record and write out any resulting records
abstract void process(Object[] record) throws HiveException;

3.function is Called to notify the UDTF that there are no more rows to process.
    Clean up code or additional output can be produced here.
abstract void close() throws HiveException;




import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.apache.hadoop.hive.ql.exec.UDFArgumentException;
import org.apache.hadoop.hive.ql.metadata.HiveException;
import org.apache.hadoop.hive.ql.udf.generic.GenericUDTF;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspectorFactory;
import org.apache.hadoop.hive.serde2.objectinspector.PrimitiveObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.StructObjectInspector;
import
org.apache.hadoop.hive.serde2.objectinspector.primitive.PrimitiveObjectInspectorFactory;

public class NameParserGenericUDTF extends GenericUDTF {
      private PrimitiveObjectInspector stringOI = null;

       //Defining input argument as string.
      @Override
      public StructObjectInspector initialize(ObjectInspector[] args) throws
UDFArgumentException {
        if (args.length != 1) {
          throw new UDFArgumentException("NameParserGenericUDTF() takes exactly one
argument");
        }

        if (args[0].getCategory() != ObjectInspector.Category.PRIMITIVE
```

```
            && ((PrimitiveObjectInspector) args[0]).getPrimitiveCategory() !=
PrimitiveObjectInspector.PrimitiveCategory.STRING) {
          throw new UDFArgumentException("NameParserGenericUDTF() takes a string as a
parameter");
        }

        // input
        stringOI = (PrimitiveObjectInspector) args[0];

        // output
        List<String> fieldNames = new ArrayList<String>(2);
        List<ObjectInspector> fieldOIs = new ArrayList<ObjectInspector>(2);
        fieldNames.add("name");
        fieldNames.add("surname");
        fieldOIs.add(PrimitiveObjectInspectorFactory.javaStringObjectInspector);
        fieldOIs.add(PrimitiveObjectInspectorFactory.javaStringObjectInspector);
        return ObjectInspectorFactory.getStandardStructObjectInspector(fieldNames, fieldOIs);
      }

      public ArrayList<Object[]> processInputRecord(String name){
            ArrayList<Object[]> result = new ArrayList<Object[]>();

            // ignoring null or empty input
            if (name == null || name.isEmpty()) {
              return result;
            }

            String[] tokens = name.split("\\s+");

            if (tokens.length == 2){
                result.add(new Object[] { tokens[0], tokens[1] });
            }else if (tokens.length == 4 && tokens[1].equals("and")){
                result.add(new Object[] { tokens[0], tokens[3] });
                result.add(new Object[] { tokens[2], tokens[3] });
            }

            return result;
      }

      @Override
      public void process(Object[] record) throws HiveException {
        final String name = stringOI.getPrimitiveJavaObject(record[0]).toString();
        ArrayList<Object[]> results = processInputRecord(name);

        Iterator<Object[]> it = results.iterator();

        while (it.hasNext()){
            Object[] r = it.next();
            forward(r);
        }
      }

      @Override
      public void close() throws HiveException {
        // do nothing
      }
}
```

Empaquete el código en el tarro y debe agregar el jar al contexto de la colmena.

```
hive> CREATE TEMPORARY FUNCTION process_names as 'jar.path.NameParserGenericUDTF';

Here we will pass input as full name and break it into first and last name.

hive> SELECT
   t.name,
   t.surname
FROM people
    lateral view process_names(name) t as name, surname;

Teena Carter
John Brownewr
```

Lea Funciones de tabla definidas por el usuario (UDTF) en línea:
https://riptutorial.com/es/hive/topic/6502/funciones-de-tabla-definidas-por-el-usuario--udtf-

Capítulo 9: Funciones definidas por el usuario de Hive (UDF)

Examples

Creación de UDF Hive.

Para crear un UDF, necesitamos extender la clase UDF (`org.apache.hadoop.hive.ql.exec.UDF`) e implementar el método de evaluación.

Una vez que se cumple con UDF y se construye JAR, debemos agregar el contexto jar to hive para crear una función temporal / permanente.

```
import org.apache.hadoop.hive.ql.exec.UDF;

class UDFExample extends UDF {

  public String evaluate(String input) {

    return new String("Hello " + input);
  }
}

hive> ADD JAR <JAR NAME>.jar;
hive> CREATE TEMPORARY FUNCTION helloworld as 'package.name.UDFExample';
hive> select helloworld(name) from test;
```

Hive UDF para recortar la cadena dada.

```
package MyHiveUDFs;

import org.apache.commons.lang.StringUtils;
import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.io.Text;

public class Strip extends UDF {

private Text result = new Text();
 public Text evaluate(Text str) {
 if(str == null) {
 return null;
 }
 result.set(StringUtils.strip(str.toString()));
 return result;
 }
}
```

exportar lo anterior al archivo jar

Vaya a la CLI de Hive y agregue el JAR UDF

```
hive> ADD jar /home/cloudera/Hive/hive_udf_trim.jar;
```

Verificar que JAR está en Hive CLI Classpath

```
hive> list jars;
/home/cloudera/Hive/hive_udf_trim.jar
```

Crear función temporal

```
hive> CREATE TEMPORARY FUNCTION STRIP AS 'MyHiveUDFs.Strip';
```

Salida UDF

```
 hive> select strip('   hiveUDF ') from dummy;
 OK
 hiveUDF
```

Lea Funciones definidas por el usuario de Hive (UDF) en línea:
https://riptutorial.com/es/hive/topic/4949/funciones-definidas-por-el-usuario-de-hive--udf-

Capítulo 10: Indexación

Examples

Estructura

```
CREATE INDEX index_name
ON TABLE base_table_name (col_name, ...)
AS 'index.handler.class.name'
[WITH DEFERRED REBUILD]
[IDXPROPERTIES (property_name=property_value, ...)]
[IN TABLE index_table_name]
[PARTITIONED BY (col_name, ...)]
[
 [ ROW FORMAT ...] STORED AS ...
 | STORED BY ...
]
[LOCATION hdfs_path]
[TBLPROPERTIES (...)]
```

Ejemplo:

```
CREATE INDEX inedx_salary ON TABLE employee(salary) AS
'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH DEFERRED REBUILD;
```

Alterar el índice

ALTER INDEX index_name ON nombre_tabla [PARTICIÓN (...)] RECONSTRUCCIÓN

Índice de caída

```
DROP INDEX <index_name> ON <table_name>
```

Si se especifica WITH DEFERRED REBUILD en CREATE INDEX, el índice creado recientemente estará inicialmente vacío (independientemente de si la tabla contiene algún dato).

El comando ALTER INDEX REBUILD se puede usar para construir la estructura del índice para todas las particiones o una sola partición.

Lea Indexación en línea: https://riptutorial.com/es/hive/topic/6365/indexacion

Capítulo 11: Insertar declaración

Sintaxis

- **Sintaxis estándar:**
- INSERT OVERWRITE TABLE tabla nombre1 [PARTICIÓN (partcol1 = val1, partcol2 = val2 ...) [SI NO EXISTE]] select_statement1 FROM from_statement;
- INSERTAR EN LA TABLA tablename1 [PARTITION (partcol1 = val1, partcol2 = val2 ...)] select_statement1 FROM from_statement;
- INSERTAR EN LA TABLA tablename1 [PARTITION (partcol1 = val1, partcol2 = val2 ...)] (z, y) select_statement1 FROM from_statement;
- **Extensión Hive (inserciones múltiples):**
- FROM from_statement
  INTRODUCIR LA TABLA DE SOBREESCRITO tablename1 [PARTITION (partcol1 = val1, partcol2 = val2 ...) [IF NOT EXISTS]] select_statement1
  [INSERTAR TABLA DE SOBREESCRITO tablename2 [PARTITION ... [IF NOT EXISTS]] select_statement2]
  [INSERTAR EN LA TABLA tablename2 [PARTITION ...] select_statement2] ...;
- FROM from_statement
  INSERTAR EN LA TABLA tablename1 [PARTITION (partcol1 = val1, partcol2 = val2 ...)] select_statement1
  [INSERTAR EN LA TABLA tablename2 [PARTITION ...] select_statement2]
  [INSERTAR TABLA DE SOBREESCRITO tablename2 [PARTITION ... [IF NOT EXISTS]] select_statement2] ...;
- **Extensión de Hive (inserciones dinámicas de partición):**
- INSERT OVERWRITE TABLE tablename PARTITION (partcol1 [= val1], partcol2 [= val2] ...) select_statement FROM from_statement;
- Insertar en la tabla nombre de tabla PARTITION (partcol1 [= val1], partcol2 [= val2] ...) select_statement FROM from_statement;

Observaciones

insertar sobrescribir
Una declaración de sobrescritura de inserción elimina cualquier archivo existente en la tabla o partición de destino antes de agregar nuevos archivos basados en la declaración de selección utilizada. Tenga en cuenta que cuando hay cambios de estructura en una tabla o en el DML utilizado para cargar la tabla, a veces los archivos antiguos no se eliminan. Al cargar en una tabla

utilizando particiones dinámicas, solo se sobrescribirán las particiones definidas por la instrucción de selección. Cualquier partición preexistente en el destino permanecerá y no se eliminará.

insertar en
Una inserción en la declaración agrega datos nuevos a una tabla de destino basada en la declaración de selección utilizada.

Examples

insertar sobrescribir

```
insert overwrite table yourTargetTable select * from yourSourceTable;
```

Insertar en la mesa

INSERT INTO se agregará a la tabla o partición, manteniendo intactos los datos existentes.

```
    INSERT INTO table yourTargetTable SELECT * FROM yourSourceTable;
```

Si una tabla está particionada, entonces podemos insertarla en esa partición en particular de forma estática como se muestra a continuación.

```
    INSERT INTO TABLE yourTargetTable PARTITION (state=CA, city=LIVERMORE)
    select * FROM yourSourceTable;
```

Si una tabla está particionada, podemos insertarla en esa partición en particular de forma dinámica, como se muestra a continuación. Para realizar inserciones dinámicas de partición, debemos establecer las siguientes propiedades.

```
    Dynamic Partition inserts are disabled by default. These are the relevant configuration
properties for dynamic partition inserts:
    SET hive.exec.dynamic.partition=true;
    SET hive.exec.dynamic.partition.mode=non-strict

    INSERT INTO TABLE yourTargetTable PARTITION (state=CA, city=LIVERMORE) (date,time)
    select * FROM yourSourceTable;
```

Inserciones múltiples en una tabla.

Extensión Hive (inserciones múltiples):

```
    FROM table_name

    INSERT OVERWRITE TABLE table_one SELECT table_name.column_one,table_name.column_two

    INSERT OVERWRITE TABLE table_two SELECT table_name.column_two WHERE table_name.column_one
== 'something'
```

Lea Insertar declaración en línea: https://riptutorial.com/es/hive/topic/1744/insertar-declaracion

Capítulo 12: Script de creación de tablas con datos de muestra

Examples

Tipos de fecha y hora

```
CREATE TABLE all_datetime_types(
  c_date date,
  c_timestamp timestamp
);
```

Valores de datos mínimos y máximos:

```
insert into all_datetime_types values ('0001-01-01','0001-01-01 00:00:00.000000001');
insert into all_datetime_types values ('9999-12-31','9999-12-31 23:59:59.999999999');
```

Tipos de texto

```
CREATE TABLE all_text_types(
  c_char char(255),
  c_varchar varchar(65535),
  c_string string
);
```

Data de muestra:

```
insert into all_text_type values ('some ****&&&%%% char value ','some $$$$#####@@@@ varchar
value','some !!~~~++ string value' );
```

Tipos numericos

```
CREATE TABLE all_numeric_types(
  c_tinyint tinyint,
  c_smallint smallint,
  c_int int,
  c_bigint bigint,
  c_decimal decimal(38,3)
);
```

Valores de datos mínimos y máximos:

```
insert into all_numeric_types values (-128,-32768,-2147483648,-9223372036854775808,-
99999999999999999999999999999999999.999);
insert into all_numeric_types values
(127,32767,2147483647,9223372036854775807,99999999999999999999999999999999999.999);
```

Tipos numéricos de punto flotante

```
CREATE TABLE all_floating_numeric_types(
  c_float float,
  c_double double
);
```

Valores de datos mínimos y máximos:

```
insert into all_floating_numeric_types values (-3.4028235E38,-1.7976931348623157E308);
insert into all_floating_numeric_types values (-1.4E-45,-4.9E-324);
insert into all_floating_numeric_types values (1.4E-45,4.9E-324);
insert into all_floating_numeric_types values (3.4028235E38,1.7976931348623157E308);
```

Tipos booleanos y binarios

```
CREATE TABLE all_binary_types(
  c_boolean boolean,
  c_binary binary
);
```

Data de muestra:

```
insert into all_binary_types values (0,1234);
insert into all_binary_types values (1,4321);
```

Nota:

- Para booleano, internamente se almacena como verdadero o falso.
- Para binario, almacenará valor codificado en base64.

Tipos complejos

FORMACIÓN

```
CREATE TABLE array_data_type(
  c_array array<string>)
 ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  COLLECTION ITEMS TERMINATED BY '&';
```

Crear `data.csv` con datos:

```
arr1&arr2
arr2&arr4
```

Ponga `data.csv` en la carpeta `/tmp` y cargue estos datos en Hive

```
LOAD DATA LOCAL INPATH '/tmp/data.csv' INTO TABLE array_data_type;
```

O puede poner este CSV en HDFS, por ejemplo, en `/tmp` . Cargar datos de CSV en HDFS usando

```
LOAD DATA INPATH '/tmp/data.csv' INTO TABLE array_data_type;
```

MAPA

```
CREATE TABLE map_data_type(
  c_map map<int,string>)
 ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  COLLECTION ITEMS TERMINATED BY '&'
  MAP KEYS TERMINATED BY '#';
```

archivo `data.csv` :

```
101#map1&102#map2
103#map3&104#map4
```

Cargar datos en la colmena:

```
LOAD DATA LOCAL INPATH '/tmp/data.csv' INTO TABLE map_data_type;
```

ESTRUCT

```
CREATE TABLE struct_data_type(
  c_struct struct<c1:smallint,c2:varchar(30)>)
 ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  COLLECTION ITEMS TERMINATED BY '&';
```

archivo `data.csv` :

```
101&struct1
102&struct2
```

Cargar datos en la colmena:

```
LOAD DATA LOCAL INPATH '/tmp/data.csv' INTO TABLE struct_data_type;
```

UNIONTYPE

```
CREATE TABLE uniontype_data_type(
  c_uniontype uniontype<int, double, array<string>)
 ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  COLLECTION ITEMS TERMINATED BY '&';
```

archivo `data.csv` :

```
0&10
1&10.23
2&arr1&arr2
```

Cargar datos en la colmena:

```
LOAD DATA LOCAL INPATH '/tmp/data.csv' INTO TABLE uniontype_data_type;
```

Lea Script de creación de tablas con datos de muestra en línea:
https://riptutorial.com/es/hive/topic/5067/script-de-creacion-de-tablas-con-datos-de-muestra

Creditos

S. No	Capítulos	Contributors
1	Empezando con la colmena	Bhavesh, Community, franklinsijo, johnnyaug, NeoWelkin
2	Creación de tablas de Hive a través de Sqoop	NeoWelkin
3	Crear base de datos y declaración de tabla	CodingInCircles, dev ツ, goks, Panther, Venkata Karthik
4	Declaración SELECT	Ambrish, dev, Jaime Caffarel, johnnyaug
5	Exportar datos en colmena	Prem Singh Bist
6	Formatos de archivo en HIVE	agentv, Alex, Community, johnnyaug, leftjoin, Mzzzzzz, NeoWelkin, tomek, Venkata Karthik
7	Funciones agregadas definidas por el usuario (UDAF)	dev ツ, Venkata Karthik
8	Funciones de tabla definidas por el usuario (UDTF)	Venkata Karthik
9	Funciones definidas por el usuario de Hive (UDF)	Ashok, Panther, Venkata Karthik
10	Indexación	Prem Singh Bist
11	Insertar declaración	Jared, johnnyaug, Venkata Karthik
12	Script de creación de tablas con datos de muestra	dev ツ