APACHE SQOOP

Sqoop - (apache.org)

1. INTRODUCCIÓN

Apache Sqoop es una **herramienta de línea de comandos** desarrollada para transferir grandes volúmenes de datos de bases de datos relacionarles a Hadoop, de ahí su nombre que viene de la fusión de SQL y Hadoop. Concretamente transforma datos relacionarles en Hive o HBase en una dirección y en la otra de HDFS a datos relacionares como **MySQL, Oracle, Postgress** o a un **data warehouse**.

El proceso de transferencia consiste leer fila por fila cada tabla de la base de datos e importarlas a HDFS, la salida de estos es un conjunto de ficheros que puede estar en **formato CSV, Avro, binario o de secuencia.**

CARACTERÍSTICAS

- Proporciona una **API Java** para realizar el procesamiento en la ingesta. Permitiendo programar aplicaciones que realicen algunas tipo de transformación sobre los datos.
- Proporciona comandos para listar tablas y esquemas.
- Soporta cargas incrementales de datos.
- Proporciona multitud de conectores como FTP, JDBC, Kafka, SFTP

2. Importación de datos

Este tema está dedicado a la transferencia de datos desde una base de datos relacional o sistema de almacén al ecosistema Hadoop. Cubriremos los casos de uso básicos de Sqoop, describiendo varias situaciones en las que tienes datos en una única tabla en un sistema de base de datos (por ejemplo, MySQL u Oracle) que quieres transferir al ecosistema Hadoop.

Describiremos varias características de Sqoop a través de ejemplos que puedes copiar y pegar en la consola y luego ejecutar. Para ello, primero tendrás que configurar tu base de datos relacional. Para los ejemplos, usaremos una base de datos MySQL con la cuenta *sqoop* y la contraseña *sqoop*. Nos conectaremos a una base de datos llamada *sqoop*.

Como Sqoop se centra principalmente en la transferencia de datos, necesitamos tener algunos datos ya disponibles en la base de datos antes de ejecutar los comandos de Sqoop. Para tener algo con lo que empezar, hemos creado la tabla ciudades que contiene algunas ciudades de todo el mundo.

Tabla Cities

id	country	city
1	USA	Palo Alto
2	Czech Republic	Brno
3	USA	Sunnyvale

Transferencia de una tabla completa

Problema

Se tiene una tabla en una base de datos relacional (por ejemplo, MySQL) y se necesita transferir el contenido de la tabla al Sistema de Archivos Distribuidos de Hadoop (HDFS).

Solución

Importar una tabla con Sqoop es muy sencillo: ejecuta el comando Sqoop import y especifica las credenciales de la base de datos y el nombre de la tabla a transferir.

```
sqoop import \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --table cities \
  --split-by id
```

Importar una tabla entera es uno de los casos de uso más comunes y sencillos de Sqoop. El resultado de este comando será un archivo CSV separado por comas en el que cada fila se almacena en una sola línea.

El parámetro `--split-by` indica el campo índice de la tabla que Sqoop utilizará para hacer las particiones de la misma, por defecto 4. Solamenta ha de indicarse si la tabla no tiene un campo índice. Se puede poner en su lugar el parámetro `-m 1` para habilitar un único mapeador o `-m n` para fijar otro número.

Tener en cuenta que este archivo CSV se creará en HDFS (en lugar de en el sistema de archivos local). Se puede inspeccionar el contenido de los archivos creados utilizando el siguiente comando:

```
% hadoop fs -cat cities/parte-m-*
```

En este ejemplo, de Sqoop fue llamado con un par de parámetros, así que vamos a discutir todos ellos con más detalle. El primer parámetro después del ejecutable `sqoop` es **import**, que especifica la herramienta apropiada. La herramienta de importación se utiliza cuando se desea transferir datos desde la base de datos relacional a Hadoop. Más adelante discutiremos la herramienta de exportación, que se utiliza para transferir datos en la dirección opuesta. El siguiente parámetro, **--connect**, contiene la URL JDBC a su base de datos. La sintaxis de la URL es específica para cada base de datos, por lo que se debe consultar el manual de la base de

datos para conocer el formato adecuado. A la URL le siguen dos parámetros, **--username** y **--password**, que son las credenciales que Sqoop debe utilizar al conectarse a la base de datos. Finalmente, el último parámetro, **--table**, contiene el nombre de la tabla a transferir.

Echemos un vistazo más de cerca para ver lo que sucederá después de ejecutar este comando. En primer lugar, Sqoop se conectará a la base de datos para obtener los metadatos de la tabla: el número de columnas de la tabla, sus nombres y los tipos de datos asociados. Por ejemplo, para la tabla ciudades, Sqoop recuperará información sobre las tres columnas: `id, country` y `city`, con `int, VARCHAR` y `VARCHAR` como sus respectivos tipos de datos. Dependiendo del sistema de base de datos concreto y de la tabla en sí, también se pueden recuperar otros metadatos útiles (por ejemplo, Sqoop puede determinar si la tabla está particionada o no). En este punto, Sqoop no está transfiriendo ningún dato entre la base de datos y su máquina, sino que está consultando las tablas y vistas del catálogo. Basándose en los metadatos recuperados, Sqoop generará una clase Java y la compilará utilizando el JDK y las librerías Hadoop disponibles en la máquina.

A continuación, Sqoop se conectará al clúster Hadoop y enviará un trabajo MapReduce. Cada *mapp*er del trabajo transferirá una porción de los datos de la tabla. Como MapReduce ejecuta varios *mapeadores* al mismo tiempo, Sqoop transferirá los datos en paralelo para lograr el mejor rendimiento posible utilizando el potencial de su servidor de base de datos. Cada *mapper* transfiere los datos de la tabla directamente entre la base de datos y el clúster Hadoop. Para evitar convertirse en un cuello de botella de la transferencia, el cliente Sqoop actúa como el supervisor en lugar de como un participante activo en la transferencia de los datos. Este es un principio clave del diseño de Sqoop.

Especificación de un directorio de destino

Por defecto, Sqoop creará un directorio con el mismo nombre que la tabla importada dentro de tu directorio home en HDFS e importará todos los datos allí. Por ejemplo, cuando el usuario *jarcec* importa la tabla ciudades, se almacenará en **/user/jarcec/cities**. Este directorio puede ser cambiado a cualquier directorio arbitrario en su HDFS usando el parámetro **--target-dir**. El único requisito es que este directorio no debe existir antes de ejecutar el comando Sqoop.
Si se desea especificar el directorio en el que los datos deben ser importados.

```
sqoop import \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --table cities \
  --target-dir /etl/input/cities
```

Importar sólo un subconjunto de datos

En lugar de importar una tabla completa, se necesita transferir sólo un subconjunto de filas en función de varias condiciones que puede expresar en forma de sentencia SQL con una cláusula WHERE.

Solución
Utilizamos el parámetro de línea de comandos `--where` para especificar una condición SQL que deben cumplir los datos importados. Por ejemplo, para importar sólo ciudades de EE.UU. de la tabla *cities*, se puede ejecutar el siguiente comando de Sqoop:

```
sqoop import \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --table cities \
  --where "country = 'USA'"
```

Cuando utilices el parámetro `--where`, hay que tener en cuenta la naturaleza paralela de las transferencias de Sqoop. Los datos serán transferidos en varias tareas concurrentes. Cualquier llamada a una función costosa supondrá una carga de rendimiento significativa para el servidor de base de datos. Las funciones avanzadas podrían bloquear ciertas tablas, impidiendo que Sqoop transfiera datos en paralelo. Esto afectará negativamente al rendimiento de la transferencia. Para un filtrado avanzado eficiente, es mejor ejecutar la consulta de filtrado en la base de datos antes de la importación, guardar su resultado en una tabla temporal y ejecutar Sqoop para importar la tabla temporal a Hadoop sin el parámetro `--where`.

Protección de la contraseña
Escribir la contraseña en la interfaz de línea de comandos es inseguro. Puede ser fácilmente recuperada de la lista de procesos en ejecución del sistema operativo.

Hay dos opciones además de especificar la contraseña en la línea de comandos con el parámetro `--password`. La primera opción es usar el parámetro `-P` que le indicará a Sqoop que lea la contraseña desde la entrada estándar. Alternativamente, puedes guardar la contraseña en un archivo y especificar la ruta a este archivo con el parámetro `--password-file`.

A continuación, se muestra una ejecución de Sqoop que leerá la contraseña de la entrada estándar:

```
sqoop import \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --table cities \
  -P
```

He aquí un ejemplo de lectura de la contraseña desde un fichero:

```
sqoop import \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --table cities \
  --password-file my-sqoop-password
```

Uso de un formato de archivo distinto de CSV

Problema

Sqoop admite formatos de archivo diferentes; uno de ellos es de texto y los otros dos son binarios. Los formatos binarios son Avro y Hadoop's SequenceFile. Puedes activar la importación a SequenceFile utilizando el parámetro `--as-sequencefile`:

```
sqoop import \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --table cities \
  --as-sequencefile
```

Avro puede activarse especificando el parámetro --as-avrodatafile:

```
sqoop import \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --table cities \
  --as-avrodatafile
```

Además, soporta los formatos que se pueden ver en el siguiente enlace:
Sqoop User Guide (v1.4.7) (apache.org)

Comprimir datos importados

Si se desea disminuir el tamaño total ocupado en HDFS utilizando compresión para los archivos generados debemos utilizare el parámetro `--compress` para activar la compresión:

```
sqoop import \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --table cities \
  --compress
```

Por defecto, cuando se utiliza el parámetro `--compress`, los archivos de salida se comprimirán utilizando el códec GZip, y todos los archivos terminarán con una extensión `.gz`. Se puede elegir cualquier otro códec utilizando el parámetro `--compression-codec`. El siguiente ejemplo utiliza el códec BZip2 en lugar de GZip (los archivos en HDFS tendrán la extensión .bz2):

```
sqoop import --compress
  --compression-codec org.apache.hadoop.io.compress.BZip2Codec
```

Aceleración de las transferencias

Sqoop es una gran herramienta y procesa muy bien las transferencias masivas. ¿Puede Sqoop funcionar más rápido?
Para algunas bases de datos se puede aprovechar el modo directo usando el parámetro `--direct`:

```
sqoop import
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop
  --table cities
  --direct
```

En lugar de utilizar la interfaz JDBC para transferir datos, el modo directo delega el trabajo de transferencia de datos a las utilidades nativas proporcionadas por el proveedor de la base de datos. En el caso de MySQL, se utilizarán `mysqldump` y `mysqlimport` para recuperar datos del servidor de base de datos o mover datos de vuelta. En el caso de PostgreSQL, Sqoop aprovechará la utilidad `pg_dump` para importar datos. El uso de utilidades nativas mejorará en gran medida el rendimiento, ya que están optimizadas para proporcionar la mejor velocidad de transferencia posible, al tiempo que suponen una menor carga para el servidor de bases de datos. Hay varias limitaciones que vienen con esta importación más rápida. Por un lado, no todas las bases de datos disponen de utilidades nativas. Este modo no está disponible para todas las bases de datos soportadas. Sqoop tiene soporte directo sólo para MySQL y PostgreSQL.

Importación de todas las tablas

Nos gustaría importar todas las tablas de la base de datos a la vez usando un comando en lugar de importar las tablas una a una.

En lugar de utilizar la herramienta de importación para una tabla, se puede utilizar la herramienta de importación de todas las tablas. Por ejemplo, para importar todas las tablas de nuestra base de datos de ejemplo, utilizarías el siguiente comando Sqoop:

```
sqoop import-all-tables \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop
```

Si se necesita importar todas las tablas menos algunas, se puede utilizar el parámetro `--exclude-tables` que acepta una lista separada por comas de nombres de tablas que deben excluirse de la importación masiva. Por ejemplo, si se necesita importar todas las tablas de la base de datos excepto *cities* y *countries*, utilizaría el siguiente comando:

```
sqoop import-all-tables \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
```

```
  --exclude-tables cities,countries
```

Importar datos de dos tablas

En lugar de utilizar la importación de tablas, se utiliza la importación de consultas. En este modo, Sqoop permitirá especificar cualquier consulta para importar los datos. En lugar del parámetro `--table`, se utiliza el parámetro `--query` con la consulta completa para obtener los datos que desea transferir.

Veamos un ejemplo con las tablas *normcities* y *countries*. Se podría utilizar el siguiente comando Sqoop:

```
sqoop import \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --query 'SELECT normcities.id, \
                  countries.country, \
                  normcities.city \
                  FROM normcities \
                  JOIN countries USING(country_id) \
                  WHERE $CONDITIONS' \
  --split-by id \
  --target-dir cities
```

Sqoop realiza transferencias de datos altamente eficientes heredando el paralelismo de Hadoop. Para ayudar a Sqoop a dividir la consulta en múltiples trozos que puedan ser transferidos en paralelo, se necesita incluir el marcador de posición `$CONDITIONS` en la cláusula *`where`* de la consulta. Sqoop sustituirá automáticamente este marcador de posición por las condiciones generadas que especifican qué trozo de datos debe transferir cada tarea individual. Aunque se podría omitir `$CONDITIONS` forzando a Sqoop a ejecutar sólo un trabajo utilizando el parámetro `--num-mappers 1`, tal limitación tendría un grave impacto en el rendimiento.

Si se quiere importar los resultados de una consulta en paralelo, entonces cada tarea `map` necesitará ejecutar una copia de la consulta, con los resultados particionados por condiciones límite inferidas por Sqoop. La consulta debe incluir el token `$CONDITIONS` que cada proceso Sqoop sustituirá por una única expresión de condición. También se debe seleccionar una columna de división con `--split-by`.

3. Exportación

Transferencia de datos desde Hadoop

La función de exportación de Sqoop que permite transferir datos desde el ecosistema Hadoop a bases de datos relacionales. Por ejemplo, para exportar datos desde el directorio *`export-dir`*

cities (el directorio en HDFS que contiene los datos de origen) a la tabla *cities* (la tabla a rellenar en la base de datos), utilizaríamos el siguiente comando de Sqoop:

```
sqoop export \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --table cities \
  --export-dir cities
```

Exportar funciona de forma similar a importar, excepto que exportar transfiere los datos en la otra dirección. En lugar de transferir datos desde la base de datos relacional mediante consultas `SELECT`, Sqoop transferirá los datos a la base de datos relacional mediante sentencias `INSERT`. El flujo de trabajo de exportación de Sqoop coincide con el de importación con ligeras diferencias. Después de ejecutar el comando Sqoop, Sqoop se conectará a la base de datos para obtener varios metadatos sobre la tabla, incluyendo la lista de todas las columnas con sus tipos apropiados. Utilizando estos metadatos, Sqoop generará y compilará la clase Java. La clase generada se utilizará en el trabajo MapReduce enviado que exportará sus datos. Al igual que en el modo de importación, no se transfieren datos a través del propio cliente Sqoop. Todas las transferencias se realizan en el trabajo MapReduce, con Sqoop supervisando el proceso.

Sqoop obtiene los metadatos de la tabla en la exportación: la tabla de destino (especificada con el parámetro `--table`) debe existir antes de ejecutar Sqoop. La tabla no tiene por qué estar vacía, e incluso se pueden exportar nuevos datos de Hadoop a la base de datos de forma iterativa. El único requisito es que no se infrinja ninguna restricción al realizar las sentencias `INSERT` (por ejemplo, no se puede exportar dos veces el mismo valor para cualquier clave primaria o única).

Inserción de datos por lotes

Aunque la función de exportación de Sqoop se ajusta a muchas necesidades, es demasiado lenta. Parece que cada fila se inserta en una sentencia de inserción independiente. ¿Hay alguna forma de insertar varias sentencias por lotes?

Adaptado a varias bases de datos y casos de uso, Sqoop ofrece múltiples opciones para insertar más de una fila a la vez.

En primer lugar, puedes activar la inserción por lotes JDBC mediante el parámetro `--batch`:

```
sqoop export \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --table cities \
  --export-dir cities \
  --batch
```

La segunda opción es utilizar la propiedad `sqoop.export.records.per.statement` para especificar el número de registros que se utilizarán en cada sentencia de inserción:

```
sqoop export \
  -D sqoop.export.records.per.statement=10 \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --table cities \
  --export-dir cities
```

Por último, se pueden establecer cuántas filas se insertarán por transacción con la propiedad `sqoop.export.statements.per.transaction`:

```
sqoop export \
  -D sqoop.export.statements.per.transaction=10 \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --table cities \
  --export-dir cities
```

Los valores por defecto pueden variar de un conector a otro. Por defecto, Sqoop deshabilita el procesamiento por lotes y establece un valor de 100 para las propiedades `sqoop.export.records.per.statement` y `sqoop.export.statements.per.transaction`.

Actualización de un conjunto de datos existente

Anteriormente se exportaron datos desde Hadoop, tras lo cual se ejecutó un procesamiento adicional que los modificó. Si en lugar de borrar los datos existentes de la base de datos, se prefiere actualizar las filas modificadas se puede aprovechar la función de actualización que emitirá sentencias UPDATE en lugar de INSERT. El modo de actualización se activa utilizando el parámetro `--update-key` que contiene el nombre de una columna que puede identificar una fila modificada, normalmente la clave primaria de una tabla. Por ejemplo, el siguiente comando permite utilizar el `id` de columna de la tabla *cities*:

```
sqoop export \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --table cities \
  --update-key id
```

El parámetro --update-key se utiliza para indicar a Sqoop que actualice las filas existentes en lugar de insertar filas nuevas. Este parámetro requiere una lista de columnas separadas por comas que deben utilizarse para identificar de forma exclusiva una fila. Todas esas columnas se utilizarán en la cláusula `WHERE` de la consulta `UPDATE` generada. Todas las demás columnas de la tabla se utilizarán en la parte `SET` de la consulta. Por ejemplo, para una tabla que contiene

cuatro columnas (c1, c2, c3 y c4), si se llama a Sqoop con `--update-key c2, c4` se generará la siguiente consulta de actualización:

```
UPDATE table SET c1 = ?, c3 = ? WHERE c2 = ? and c4 = ?
```

Es muy importante entender la estructura de la consulta para ver cómo el modo de actualización exportará datos desde Hadoop. En primer lugar, las columnas utilizadas para identificar la fila nunca se actualizarán porque no forman parte de la cláusula `SET`. Además, si los datos en Hadoop contienen algunas filas completamente nuevas, la cláusula `WHERE` no coincidirá con ninguna fila en el lado de la base de datos. Una operación de este tipo en el lado de la base de datos es totalmente válida, pero no da lugar a filas actualizadas.