Sqoop

Motivación

Es una herramienta para transferir datos entre **RDBMs** y **Hadoop**

Escenario típico: datos almacenados en Bases de Datos Relacionales que queremos operar aprovechando la potencia de Hadoop

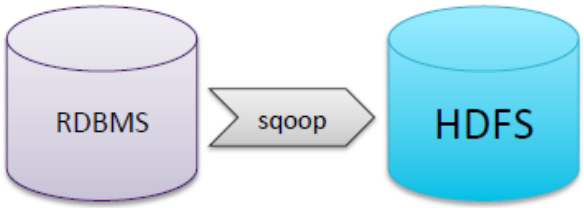
- Datos de uso general
- Datos Legacy

Es posible acceder a los datos de una BBDD como si fuera un origen más especificándolo en nuestro Mapper

- Esto llevaría a un Distributed Denial of Service (DDoS), es decir, estaríamos Hackeando nuestra BBDD.
- No parece una buena solución.

Solución

- **Utilizar** la herramienta **Sqoop para importar datos** desde las RDBMS a HDFS



Sqoop: de SQL a Hadoop

Sqoop es una herramienta **Open Source** creada originalmente por Cloudera
- Ahora es un proyecto perteneciente al Apache Software Foundation

Su nombre viene de SQL to Hadoop → Sqoop

Su **objetivo fundamental es transferir datos entre RDBMS y Hadoop** (HDFS). Hay varias opciones:
- Transferir solo una tabla
- Transferir todas las tablas en una BBDD
- Transferir partes de una tabla. Sqoop soporta la cláusula WHERE de SQL

Para **importar** datos utiliza **MapReduce**
- Pero permite determinar cuántos maps se pueden ejecutar a la vez.
- Por defecto usa cuatro Maps
- Aunque este valor es configurable
- Esto provee además la posibilidad de paralelización y tolerancia a fallos
- Los datos son importados registro a registro

Utiliza una inferfaz JDBC
- Que en principio es compatible con casi todas las BBDD compatibles con esta interfaz

Sqoop: algunas características

Los **datos son importados** a HDFS **como text files** delimitados o SequenceFiles

- Por defecto se importan como text files **separados por comas**

Los ficheros son importados mediante MapReduce, como hemos comentado, por lo tanto se almacenan en formato part-*.0*

Es posible usarlo para importaciones de datos incrementales

- El primer import importa todas las filas en una tabla
- El resto de imports solo las filas creadas desde la última importación (argumentos)
  - **check-column (col)**: Especifica la columna que debe ser examinada cuando se determina qué columas hay que importer.
  - **incremental (mode):** Especifica como Sqoop determina qué columas son nuevas. Los valores de mode pueden ser append y lastmodified, que son las dos formas que hay de hacer imports incrementales
  - **last-value (value)**: Especifica el máximo valor de la última columna leída desde la última importación. El valor se imprime por pantalla.

El resultado de la importación genera un fichero/clase java class

- Es usada durante la importación por Sqoop
- Serializa y deserializa datos en formato SequenceFile.
- Permite reutilizar el código en subsecuentes MapReduce Jobs
- Permite parsear registros con formato de texto delimitado

Sqoop: conectores genéricos

Conectores genéricos de Sqoop

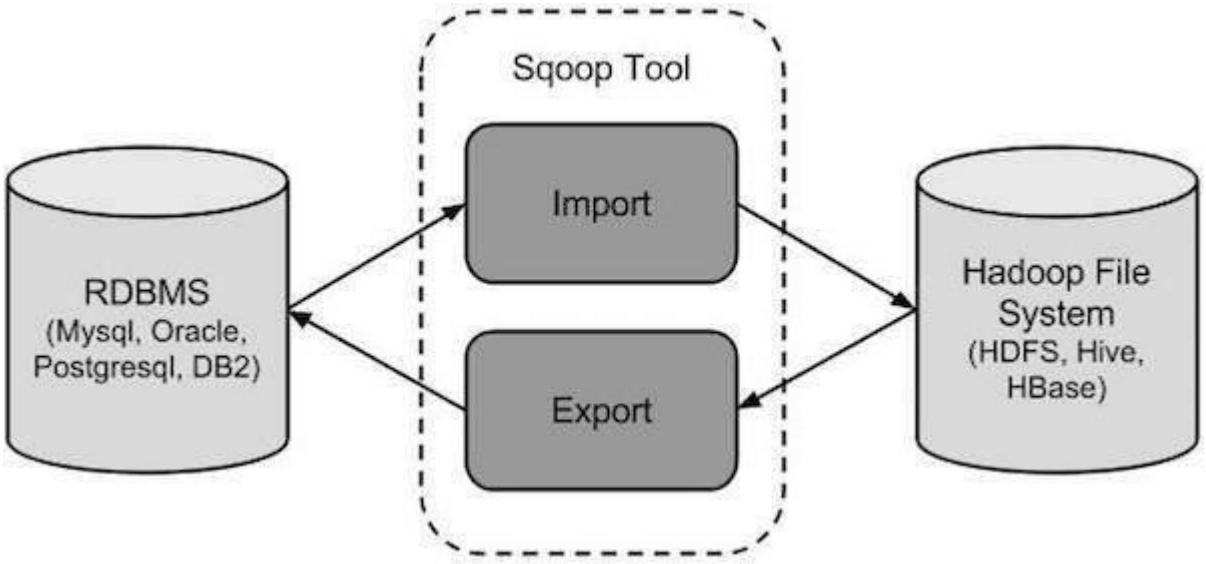
Se utilizan para configurar los parámetros básicos de sqoop

```
$ sqoop help import
usage: sqoop import [GENERIC-ARGS] [TOOL-ARGS]

Common arguments:
   --connect <jdbc-uri>     Specify JDBC connect string
   --connect-manager <class-name>     Specify connection manager class to use
   --driver <class-name>    Manually specify JDBC driver class to use
   --hadoop-mapred-home <dir>      Override $HADOOP_MAPRED_HOME
   --help                   Print usage instructions
   --password-file          Set path for file containing authentication password
   -P                       Read password from console
   --password <password>    Set authentication password
   --username <username>    Set authentication username
   --verbose                Print more information while working
   --hadoop-home <dir>     Deprecated. Override $HADOOP_HOME

[...]

Generic Hadoop command-line arguments:
(must preceed any tool-specific arguments)
Generic options supported are
-conf <configuration file>     specify an application configuration file
-D <property=value>            use value for given property
-fs <local|namenode:port>      specify a namenode
-jt <local|jobtracker:port>    specify a job tracker
-files <comma separated list of files>    specify comma separated files to be copied to the map reduce cluster
-libjars <comma separated list of jars>    specify comma separated jar files to include in the classpath.
-archives <comma separated list of archives>    specify comma separated archives to be unarchived on the compute machines.

The general command line syntax is
bin/hadoop command [genericOptions] [commandOptions]
```

Sqoop: conectores a medida

Existen conectores específicos para diferentes BBDD

- Su objetivo es proveer de una interfaz más rápida de importación
- Habitualmente no son opensource pero si son gratis



Sqoop: compatibilidad con BBDDs

Database	version	--direct support?	connect string matches
HSQLDB	1.8.0+	No	jdbc:hsqldb:*//
MySQL	5.0+	Yes	jdbc:mysql://
Oracle	10.2.0+	No	jdbc:oracle:*//
PostgreSQL	8.3+	Yes (import only)	jdbc:postgresql://
CUBRID	9.2+	NO	jdbc:cubrid:*

El modo **direct** permite conectar a la base de datos sin utilizar JDBC, lo cual permite mayor rapidez.

Como hemos comentado, existen otros conectores no OpenSource pero sí gratuitos.

Sqoop: Sintaxis Básica

La sintaxis básica corresponde a lo siguiente

- sqoop tool-name [tool-options]

Ejemplos de tool-name

- import
- import-all-tables
- list-tables

Ejemplos de tool-options

- --connect
- --username
- --password

Sqoop: Ejemplo

Ejemplo: importar una tabla llamada *empleados* de una bbdd llamada *personal*

```
sqoop import --username user --password pass \
--connect jdbc:mysql://database.example.com/personal  --table empleados
--tarjet-dir XXX
```

Otras formas de autenticación

- **-p** recoge la password desde el prompt
- **--pasword-alias**, indica el fichero donde está almacenado la contraseña. Similar a--pasword-file
- **--password**, método inseguro donde se pasa la password en claro.

Ejemplo: igual que la anterior pero solamente aquellos empleados con más de 35 años

```
sqoop import --username user --password pass \
--connect jdbc:mysql://database.example.com/personal \
--table empleados --where “edad>35” --tarjet-dir XXX
```

Sqoop: Ejemplo

Queries específicas con **--query**

```
sqoop import [% argumentos %] --query 'SELECT a.*, b.* FROM a JOIN b on
(a.id == b.id) WHERE $CONDITIONS'  --target-dir /user/foo/joinresults
```

Para controlar el número de Mappers utilizar la concición **-m** o **--num-mappers**

```
sqoop import \ --query 'SELECT a.*, b.* FROM a JOIN b on (a.id == b.id)
 WHERE $CONDITIONS'  -m 1 --target-dir /user/foo/joinresults
```

Los **jars** necesarios por Sqoop se copian en la **distributed cache** en Hadoop

Sqoop: Ejemplo

Opciones Import

Argument	Description
`--append`	Append data to an existing dataset in HDFS
`--as-avrodatafile`	Imports data to Avro Data Files
`--as-sequencefile`	Imports data to SequenceFiles
`--as-textfile`	Imports data as plain text (default)
`--boundary-query <statement>`	Boundary query to use for creating splits
`--columns <col,col,col…>`	Columns to import from table
`--direct`	Use direct import fast path
`--direct-split-size <n>`	Split the input stream every *n* bytes when importing in direct mode
`--inline-lob-limit <n>`	Set the maximum size for an inline LOB
`-m,--num-mappers <n>`	Use *n* map tasks to import in parallel
`-e,--query <statement>`	Import the results of *`statement`*.
`--split-by <column-name>`	Column of the table used to split work units
`--table <table-name>`	Table to read
`--target-dir <dir>`	HDFS destination dir
`--warehouse-dir <dir>`	HDFS parent for table destination
`--where <where clause>`	WHERE clause to use during import
`-z,--compress`	Enable compression
`--compression-codec <c>`	Use Hadoop codec (default gzip)
`--null-string <null-string>`	The string to be written for a null value for string columns
`--null-non-string <null-string>`	The string to be written for a null value for non-string columns

Sqoop: otras opciones

Con Sqoop es posible **exportar** datos desde HDFS e insertarlos en una tabla existente en una RDBMS

```
sqoop export [options]
```

La ayuda en Sqoop se obtiene así:

```
sqoop help
```

Y para obtener ayuda de un comando en particular

```
sqoop help command
```

Sqoop: Hive

Es posible importar datos directamente a Hive en lugar de hacerlo a HDFS

Para ello se utiliza el comando **--hive-import**

En este caso Sqoop genera las siguientes acciones

- Si la tabla hive ya existe, se puede indicar en las opciones la condición **overwrite** para que sobreescriba los datos en la tabla
- Sqoop generará un script hive con la creación de la tabla si esta no está creada
- Sqoop generará una instrucción de carga para mover los datos importados al warehouse de hive

```
sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES --hive-import
```

Sqoop: Hive

Argumentos Hive

Argument	Description
`--hive-home <dir>`	Override `$HIVE_HOME`
`--hive-import`	Import tables into Hive (Uses Hive's default delimiters if none are set.)
`--hive-overwrite`	Overwrite existing data in the Hive table.
`--create-hive-table`	If set, then the job will fail if the target hive
	table exits. By default this property is false.
`--hive-table <table-name>`	Sets the table name to use when importing to Hive.
`--hive-drop-import-delims`	Drops *\n*, *\r*, and *\01* from string fields when importing to Hive.
`--hive-delims-replacement`	Replace *\n*, *\r*, and *\01* from string fields with user defined string when importing to Hive.
`--hive-partition-key`	Name of a hive field to partition are sharded on
`--hive-partition-value <v>`	String-value that serves as partition key for this imported into hive in this job.
`--map-column-hive <map>`	Override default mapping from SQL type to Hive type for configured columns.

Sqoop: Hive

La API completa se encuentra en: https://sqoop.apache.org/

Sqoop: Hive

Ejercicios