gopikrishna.pujari@gmail.com

Sqoop Practice

Why we need Sqoop

- RDBMS is for storing and processing structured data. Some times if our RDBMS can't store the huge data and can't process huge amount of data better we can keep that data into HDFS and process that data.
- sqoop working like bridge in between RDBMS and HDFS.
  RDBMS => HDFS (Import)
  HDFS => RDBMS(Export)
- Automate the process
- Sqoop is built on top of MapReducce
- Sqoop is bridge between RDBMS and HDFS
- Sqoop is running with Sqoop interpreter.
- By default sqoop is running with 4 mappers and no reducers

1) Installation

2) import table from mysql(RDBMS) to HDFS
3) import part table from mysql(RDBMS) to HDFS
4) import all tables from a specific database from mysql(RDBMS) to HDFS
5) export a table from HDFS to specific database to mysql(RDBMS)
6) Display databases from mysql(RDBMS)
7) Display tables from mysql(RDBMS)
8) Importing RDBMS data to the HIVE table
9) Exporting data from HIVE table to RDBMS (mysql)table
10) sqoop-eval
11) sqoop-job
12) Using Options files to pass Arguments
13) sqoop-codegen

gopikrishna.pujari@gmail.com

1) Installation

- ✓ you can download the hadoop related softwares from the site **https://archive.apache.org/dist/**
- ✓ sqoop can be downloaded from **https://archive.apache.org/dist/sqoop/1.4.3/**
- ✓ extract the sqoop-1.4.3.tar.gz and place it in /opt/sqoop
- ✓ include the following commands in .bashrc file

```
export SQOOP_HOME/opt/sqoop
export PATH=$PATH:$SQOOP_HOME/bin
```

- ✓ you can work with sqoop from the terminal

```
$ sqoop help import
```

****

Note : to work with sqoop we must have database and tables in mysql(RDBMS)
first we need to place mysql-connector-java-5.1.24-bin.jar file in $SQOOP_HOME/lib folder

Create a database in mysql with the name sqoopdb and create files in that.

```
$ mysql -utraining –ptraining
```

Note: if you know the database and to enter into the database directly while login into mysql use the following command
$mysql utraining –ptraining <database-name>

➔ create a database
```
mysql> create database sqoopdb;
```

➔ grant all permission for 'training' user on 'sqoopdb'
```
mysql>GRANT ALL ON sqoopdb.* TO 'training'@'localhost';
```

➔ to check the previleages
```
mysql>select Host,User,Show_db_priv from mysql.user;
```
➔ change the current database to sqoopdb.

```
mysql> use sqoopdb;
```

➔ create a table

```
mysql>  create table emp(id int,name varchar(20),sal float);
mysql> show tables;
mysql> select * from emp;
```

➔ insert values into emp table

```
mysql>INSERT INTO emp VALUES(100,"ramesh",5000),(101,"kranthi",6000), (103,"verma",4000), (104,"kiran",6000),
(105,"suri",4000), (106,"murthi",9000), (107,"raju",4500), (108,"vamsi",6500);
```

➔ create a table dept

```
mysql>  create table dept(did int,dname varchar(20),ename varchar(20));
```

➔ insert the values into dept

```
mysql>INSERT INTO dept VALUES(1,"IT","ramesh"),(2,"HR"," verma "),(3," Sales "," kiran "),(4," Mkt "," murthi ");


mysql> select * from emp;
```

➔ If we want to add primary key to a table

```
mysql>ALTER TABLE emp ADD PRIMARY KEY(id);
```

➔ If a primary key already exists then you want to do this

```
mysql>ALTER TABLE emp DROP PRIMARY KEY, ADD PRIMARY KEY(id,name);
```

****

2) import table from mysql(RDBMS) to HDFS

Syntax:

```
$ sqoop import \
--connect jdbc:mysql://<host_name>:<port>/<database_name> \
--username <user_name> --password <password> \
--table <mysql_table_name> --fields-terminated-by '<delimiter>'
```

```
$ sqoop import \
--connect jdbc:mysql://localhost/sqoopdb \
--username training --password training \
 --table emp --fields-terminated-by '\t'
Error during import: No primary key could be found for table emp. Please specify one with --split-by or perform a sequential import with '-m 1'.
```

****

Note: If primary key is defined for a table in mysql then we don't need to define mappers during sqoop import by -m and it will run 4 mappers automatically.
If there is no primary key is defined for a table in mysql we need to specify the no. of mapper through -m in sqoop import

****

```
$ sqoop import \
--connect jdbc:mysql://localhost/sqoopdb \
--username training --password training \
--table emp --fields-terminated-by '\t'  -m 1
```

- ➔ By running the above command under "/tmp/sqoop-training/compile/folder" it will create <table-name>.java,<table-name>.class, <table-name>.jar file. Here emp.java,emp.class,emp.jar are created.
- ➔ if we are not specify the "--fields-terminated-by" clause it will store as comma separated values by default
- ➔ only one mapper will be run because we have specified that –m 1

```
$ hadoop fs -ls /user/training/emp/part*
-rw-r--r--   1 training supergroup        155 2016-08-10 14:39 /user/training/emp/part-m-00000
```

```
mysql>ALTER TABLE emp ADD PRIMARY KEY(id);
```

```
$ sqoop import \
--connect jdbc:mysql://localhost/sqoopdb \
--username training --password training \
--table emp --fields-terminated-by '\t'
ERROR tool.ImportTool: Encountered IOException running import job:
org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory emp already exists
```

Remove the directory and rerun the above command

```
$ hadoop fs -rmr /user/training/emp
```

We can specity the target-dir by the following command

```
$ sqoop import \
--connect jdbc:mysql://localhost/sqoopdb \
--username training --password training \
--table emp --fields-terminated-by '\t' --target-dir /user/nandu
```

- ➔ there should not be the "/user/nandu" folder before running the above command. If present we will get the following error
  ```
  ERROR security.UserGroupInformation: PriviledgedActionException as:training
  cause:org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory /user/nandu already exists
  ```

➔ this will create a directory /user/nandu
➔ we are not specified the mapper. so by default 4 mappers will be run.

```
$ hadoop fs -ls /user/nandu/part*

-rw-r--r--   1 training supergroup        37 2016-08-10 15:01 /user/nandu/part-m-00000
-rw-r--r--   1 training supergroup        34 2016-08-10 15:01 /user/nandu/part-m-00001
-rw-r--r--   1 training supergroup        33 2016-08-10 15:01 /user/nandu/part-m-00002
-rw-r--r--   1 training supergroup        51 2016-08-10 15:02 /user/nandu/part-m-00003
```

****

Note :

i) By default, Sqoop will create a directory with the same name as the imported table inside your home directory on HDFS and import all data there.
HDFS home directory will be "/user/<logged_in_user>".

➔ so by the above statements the directory table will be created under /user/training
➔ The above command will copy external rdbms table data(here emp) into "/user/training/emp"

ii) We can specify
"**--target-dir /user/nandu** " in above statement then it will copy the external rdbms table data(here emp) into "/user/nandu/emp"

➔ The only requirement is that this directory "/user/nandu" must not exist prior to running the Sqoop command.

iii) We can specify
"**--warehouse-dir /user/moneesh** " in above statement then

➔ If you want to run multiple Sqoop jobs for multiple tables, you will need to change the --target-dir parameter with every invocation.
➔ As an alternative, Sqoop offers another parameter by which to select the output directory. Instead of directly specifying the final directory, the parameter **"--warehouse-dir"** allows you to specify only the parent directory.

➔ Rather than writing data into the warehouse directory, Sqoop will create a directory with the same name as the table inside the warehouse directory and import data there. This is similar to the default case where Sqoop imports data to your home directory on HDFS, with the notable exception that the --warehouse-dir parameter allows you to use a directory other than the home directory. Note that this parameter does not need to change with every table import unless you are importing tables with the same name.

The import is done in two steps

- The first step Sqoop introspects the database to gather the necessary metadata for the data being imported
- The second step is a map-only Hadoop job that sqoop submits to the cluster. It is the job that does the actual data transfer using the metadata captured in the previous step.

****

Incremental import

We have imported mysql "emp" table data into HDFS (/user/nandu/emp). Suppose the last record id will be "300".

After importing the table data into HDFS the mysql table is get updated with another 200 records and now total 500 records are present in mysql table.

But in HDFS we have only 300 records present. So if you want to update the HDFS data also we can use incremental import. By this only newly updated records in mysql are going to be inserted in the HDFS.

Syntax

```
$ sqoop import \
--connect jdbc:mysql://<host_name>:<port>/<database_name> \
--username <user_name> --password <password> \
--table <mysql_table_name> --fields-terminated-by '<delimiter>'
--target-dir <HDFS Path>
--incremental <mode> \
--check-column <column_name> \
--last-value <last_check_column_value> -m 1
```

```
$ sqoop import \
--connect jdbc:mysql://localhost/sqoopdb \
--username training --password training \
--table emp --fields-terminated-by '\t' --target-dir /user/nandu \
--incremental append \
--check-column id \
--last-value 300 -m 1
```

- ➔ By “append” instead of updating all the records only new records will be updated in HDFS
- ➔ check-column will be mostly primary key of the table, here in emp table “id” is the primary key
- ➔ last-value value is 300 .

---

3) import part table from mysql(RDBMS) to HDFS

Syntax:

```
$ sqoop import \
--connect jdbc:mysql://<host_name>:<port>/<database_name> \
--username <user_name> --password <password> \
--table <table_name> --fields-terminated-by '<delimiter>'
--columns "<field_name1>,<field_name2>"
```

or

```
--where " <condition>" -m <no_of_mappers>
```

```
$ sqoop import \
--connect jdbc:mysql://localhost/sqoopdb \
--username training --password training \
--table emp --fields-terminated-by '\t'
--columns "name,sal"  -m 1 --target-dir /user/nandu/emp1
```

- ➔ all employee details with name,sal fields only will be stored in /user/nandu/emp1.

```
$hadoop fs -ls /user/nandu/emp1/part*
```

```
$ sqoop import  \
--connect jdbc:mysql://localhost/sqoopdb \
--username training --password training \
--table emp --fields-terminated-by '\t'  \
--where "sal>8000.0"  -m 1 --target-dir /user/nandu/emp2
```

- ➔ the employees whose sal is greater than 8000.0 only be stored in the emp2.

```
$ hadoop fs -ls /user/nandu/emp2/part*
```

Note: **--columns** option will filter the required columns
--where option will filter the required rows

4) import all tables from a specific database from mysql(RDBMS) to HDFS

Syntax:

```
$ sqoop import-all-tables \
--connect jdbc:mysql://<host_name>:<port>/<database_name> \
--username <user_name> --password <password> \
--field-terminated-by '<delimiter>'
```

```
$ sqoop import-all-tables \
--connect jdbc:mysql://localhost/sqoopdb \
--username training --password training \
--fields-terminated-by '\t' -m 1
```

- ➔ two directories will be created : /user/training/emp and /user/training/dept
- ➔ in "/tmp/sqoop-training/compile/folder" emp.java, emp.class, emp.jar, dept.java, dept.class, dept.jar files will be created.

---

5) export a table from HDFS to specific database to mysql(RDBMS)

Note: the database and tables should be created in mysql with specific field names .

Syntax:

```
$ sqoop export \
--connect jdbc:mysql://<host_name>:<port>/<database_name> \
--username <user_name> --password <password> \
--table <newly_created_mysql_table_name> --input-fields-terminated-by '<delimiter>' \
--export-dir <hdfs-directory-path>
```

Prerequisite: before exporting make sure that the table must be created in mysql

i) first create a table in mysql with specified field names according to the input data.

```
mysql>use sqoopdb;
mysql> create table gopie_test(id int, name varchar(20),primary key(id));
```

ii) load the data from LFS to HDFS

/user/training/gopietest.txt(LFS)

```
100    gopie
101    moneesh
102    nandu
103    chanti
104    ramesh
105    narayana
106    kiran
```

```
$hadoop fs -put /home/training/gopietest.txt /user/training/moneesh/
```

iii) run the sqoop export command

```
$ sqoop export \
 --connect jdbc:mysql://localhost:3306/ sqoopdb \
--username training --password training \
--table gopie_test --input-fields-terminated-by '\t' \
--export-dir '/user/training/moneesh/gopietest.txt'
```

---

6) Display databases from mysql(RDBMS)

Syntax

```
 $ sqoop list-databases \
 --connect jdbc:mysql://<host_name>:<port> \
 --username <user_name> --password <password>
```

```
$ sqoop list-databases --connect jdbc:mysql://localhost --username training --password training
$ sqoop-list-databases --connect jdbc:mysql://localhost --username training --password training
```

---

7) Display tables from mysql(RDBMS)

Syntax

```
 $ sqoop list-tables \
 --connect jdbc:mysql://<host_name>:<port>/<database_name> \
 --username <user_name> --password <password>
```

```
$ sqoop list-tables --connect jdbc:mysql://localhost/sqoopdb --username training --password training
```

8) Importing RDBMS data to the HIVE table

Run the hiveserver2 before executing the command

syntax

```
$ sqoop import \
--connect jdbc:mysql://<host_name>:<port>/<database_name> \
--username <user_name> --password <password> \
--table <mysql-table-name> \
--fields-terminated-by '<delimiter>' --target-dir <hdfs path>
--hive-table <table-to-create-hive> --create-hive-table
--hive-import --hive-home <hive-warehouse> -m 1
```

```
$ sqoop import --connect jdbc:mysql://localhost/ sqoopdb --username training --password training  --table emp --
fields-terminated-by '\t'  --target-dir /user/nandu/krish --hive-table mysql_emp --create-hive-table --hive-import --
hive-home /user/hive/warehouse -m 1
```

Process of execution

- ➢ first the mysql data(emp table in sqoopdb) is being imported into HDFS home location if we are not using "--target-dir".
- ➢ If we are using "--target-dir" option then the mysql table data (emp table in sqoopdb) is imported into "/user/nandu/krish " location.
- ➢ The directory "/user/nandu/krish " should not be exist before running the above command
- ➢ If we use "--create-hive-table" then sqoop interpreter will automatically creates the hive table " mysql_emp " based on the schema defined for mysql table emp.
- ➢ If we are not using "--create-hive-table" clause then we need to explicitly create the table " hive_emp " in hive.
- ➢ with the help of "--hive-import --hive-home /user/hive/warehouse" the data from HDFS location "/user/nandu/krish " is imported into the hive warehouse location
- ➢ we can fire the query for the table data of mysql_emp through hive shell

```
hive> select  * from mysql_emp;
```

➔ here "user/hive/warehouse" means that it is the "default" dababase location of Hive.

```
mysql data ----> HDFS location ---> Hive warehouse location
emp table -----> /user/nandu/krish --> /user/hive/warehouse
```

****

➔ If HBase and Hive are installaed on VM and they are using different versions of thrift jars. So Sqoop will not be able to upload the data into hive table because of the different version of thrift jars.
➔ change HBASE_HOME to some other non-existent path in your .bashrc file and do the import.
➔ Revert back HBASE_HOME once you have performed the import to hive table.
➔ close the hive shell if you have already opened it.

****

9) Exporting data from HIVE table to RDBMS (mysql)table

Syntax:

```
sqoop export \
-- connect jdbc:mysql://<host_name>:<port>/<database_name> \
--username <user-name> --password <password>  \
--table <mysql-table-name> \
--export-dir <hive-warehouse-directory> -m 1 --input-fields-terminated-by '<delimiter>'
```

a) If you are exporting the data from the hive table to RDBMS table, you need to provide how the input fields are terminated in your hive table. By default they are terminated by '\001'('\001' is octal representation of ^A). By default, Hive will stored data using ^A as a field delimiter and \n as a row delimiter.

b) Select the hive table which is going to be exported to mysql. Here we are taking 'mysql_emp'(which is created in previous command). hive_emp table data will be present in the hdfs location "/user/hive/warehouse/mysql_emp", which will be our --export-dir.

c) Then create a mysql table "hive_emp" in sqoopdb with the same schema of "hive_emp".
```
mysql> use sqoopdb;
mysql> create table hive_emp(id int,name varchar(20),sal float, PRIMARY KEY(id));
```

d) fire the following command

```
$ sqoop export --connect jdbc:mysql://localhost/sqoopdb --username training --password training  --table hive_emp --export-dir /user/hive/warehouse/mysql_emp -m 1 --input-fields-terminated-by '\t'
```

We can import data from mysql to HDFS, Hive, HBase.
We can export data from HDFS, Hive to mysql. And we can't export data from HBase to mysql.

****
http://sqoop.apache.org/docs/1.4.2/SqoopUserGuide.html
http://www.geoinsyssoft.com/avro-file-format-using-sqoop/
http://wpcertification.blogspot.in/2015/05/importing-data-from-sqoop-into-hive.html
****

10) sqoop-eval

➔ allows users to quickly run simple SQL queries against a database.
➔ results are printed to the console.
➔ this allows users to preview their import queries to ensure they import the data they expect.

syntax

```
$ sqoop eval (generic-args) (eval-args)
$ sqoop-eval (generic-args) (eval-args)
```

Common Arguments (generic-args)

Argument	Description
--connect <jdbc-uri>	Specify JDBC connect string
--connection-manager	<class-name> Specify connection manager class to use
--driver <class-name>	Manually specify JDBC driver class to use
--hadoop-home <dir>	Override $HADOOP_HOME
--help	Print usage instructions
-P	Read password from console

--password <password>	Set authentication password
--username <username>	Set authentication username
--verbose	Print more information while working
--connection-param-file <filename>	Optional properties file that provides connection parameters

SQL evaluation arguments(eval-args)

Argument	Description
-e,--query <statement>	Execute statement in SQL.

Select ten records from the emp table in sqoopdb:
$ sqoop eval --connect jdbc:mysql://localhost/sqoopdb --username training --password training --query "SELECT * FROM emp LIMIT 10"

Insert a row into the emp table:
$ sqoop eval --connect jdbc:mysql://localhost/sqoopdb --username training --password training -e "INSERT INTO emp VALUES(109, 'Moneesh',15000)"

11) sqoop-job

- ➔ this job tool allows you to create and work with saved jobs. Saved jobs remember the parameters used to specify a job, so they can be re-executed by invoking the job by its handle.
- ➔ If a saved job is configured to perform an incremental import, state regarding the most recently imported rows is updated in the saved job to allow the job to continually import only the newest rows.

Syntax

```
$ sqoop job (generic-args) (job-args) [-- [subtool-name] (subtool-args)]
$ sqoop-job (generic-args) (job-args) [-- [subtool-name] (subtool-args)]
```

Job management options:

Argument	Description
--create <job-id>	Define a new saved job with the specified job-id (name). A second Sqoop command-line, separated by a -- should be specified; this defines the saved job.
--delete <job-id>	Delete a saved job.
--exec <job-id>	Given a job defined with --create, run the saved job.
--show <job-id>	Show the parameters for a saved job.
--list	List all saved jobs

Savede job
----
i) Sqoop does not store passwords in the metastore
ii) it prompts each time at job invocation
ii) But it can be overriden by setting
sqoop.metastore.client.record.password to true

a) Creation of saved jobs is done with the "--create" action. The following is a saved job which will import the dept table into HDFS (/user/training/dept) when it is going to execute.

   ```
   $ sqoop job --create gkjob -- import --connect jdbc:mysql://localhost/sqoopdb --username training --password training --table dept -m 1
   ```

   Note: in HDFS the data will be stored as comma separated values by default

b) to list the saved jobs
$ sqoop job --list

c) We can inspect the configuration of a job with the show action:
$ sqoop job --show gkjob

d) And if we are satisfied with it, we can run the job with exec:
$ sqoop job --exec gkjob

12) Using Options files to pass Arguments

- When using Sqoop, the command line options that do not change from invocation to invocation can be put in an options file for convenience.
- An options file is a text file where each line identifies an option in the order that it appears otherwise on the command line.
- Option files allow specifying a single option on multiple lines by using the back-slash character at the end of intermediate lines.
- Also supported are comments within option files that begin with the hash character.
- Comments must be specified on a new line and may not be mixed with option text.
- All comments and empty lines are ignored when option files are expanded.
- Unless options appear as quoted strings, any leading or trailing spaces are ignored.

- Quoted strings if used must not extend beyond the line on which they are specified.

****

i) create an options file in /home/training/sqoopimport.txt

```
#
# Options file for Sqoop import
#

# Specifies the tool being invoked
import

# Connect parameter and value
--connect
jdbc:mysql://localhost

# Username parameter and value
--username
training

--password
training

#
# Remaining options should be specified in the command line.
#
```

ii) run the following command

$ sqoop list-databases --options-file /home/training/sqoopimport.txt

➔ it will list the databases.

13) sqoop-codegen

- The codegen tool generates Java classes which encapsulate and interpret imported records.
- The Java definition of a record is instantiated as part of the import process, but can also be performed separately.
- For example, if Java source is lost, it can be recreated. New versions of a class can be created which use different delimiters between fields, and so on.

$ sqoop codegen --connect jdbc:mysql://localhost/sqoopdb --username training --password training --table emp
➔ this will creates a folder under "/tmp/sqoop-training/compile" with emp.java, emp.class and emp.jar

$ sqoop codegen --connect jdbc:mysql://localhost/sqoopdb --username training --password training --table emp --outdir /home/training/gopiedata/sqoop --class-name com.sqoopexample.SqoopImportEmp

➔ this will creates "SqoopImportEmp.java" file under "/home/training/gopiedata/sqoop/com/sqoopexample"
➔ also creates "SqoopImportEmp.class" file under "/tmp/sqoop-training/compile/<randomFolder>/com/sqoopexample"
➔ a jar file "com.sqoopexample.SqoopImportEmp.jar" wiil be created under "/tmp/sqoop-training/compile/<randomFolder>

Sqoop Connectors

By default sqoop includes connectors for various popular databases

MySQL
PostgreSQL
SQL Server
Oracle
DB2