



­

Assignment Document:

SQOOP

Sqoop is a connector between RDBMS to Hadoop (Import) or Hadoop to RDBMS(Export).

**2 options while importing data to Hadoop:**

1. Importing table data from RDBMS table to HDFS(file system)
2. Importing table data from RDBMS table to hive table

**2 options while exporting data to RDBMS:**

1. Exporting data from HDFS(file system) to RDBMS table
2. Exporting data from hive table to RDBMS table

**List all databases in MySQL**

sqoop-list-databases --connect jdbc:mysql://localhost --username hadoop --password hadoop

**List all tables in MySQL**

sqoop-list-tables --connect jdbc:mysql://localhost/hive --username hadoop --P

**Note :** If we pass -P as parameter, we can give the password in the run time so that the password is not hard-coded for security reasons.

**Pass parameter file to Sqoop**

sqoop-list-tables --options-file /root/cts/sqoop\_param

cat>sqoop\_param

--connect

jdbc:mysql://localhost/hive

--username

hadoop

--password

hadoop

**Import table data to HDFS (O/P file will be by default delimited text)**

sqoop-import --options-file /root/cts/sqoop\_param --table Employee -m 1

If there is no primary key in the table, then we need to specify no:mappers as 1 i.e., sequential import of data. we can explicitly specify no:of mappers for parallel import.

If there is no primary key in the table & if we need parallel import then, use split-by "some column name" & can specify any numbers of mappers. Those many part-m files will be generated.

**By default if there is primary key in a table or using split-by, it uses 4 mappers. We can see 4 mappers output in HDFS.**

sqoop-import --options-file /root/cts/sqoop\_param --table Employee --split-by EDept

**Import table data to HDFS (Import only specific columns)**

sqoop-import --options-file /root/cts/sqoop\_param --table Employee -m 1 --columns "ENo,ESal"

**Import table data to HDFS (Tab separated file format)**

sqoop-import --options-file /root/cts/sqoop\_param --fields-terminated-by '\t' --table Employee -m 1

**Import table data to HDFS (Save to target directory)**

sqoop-import --options-file /root/cts/sqoop\_param --table Employee -m 1 --target-dir /user/Employee

**Import table data to HDFS (Where condition)**

sqoop-import --options-file /root/cts/sqoop\_param --table Employee -m 1 --where "ESal > 50000"

**Import table data to HDFS (Where condition)**

sqoop-import --options-file /root/cts/sqoop\_param --table Employee --split-by EDept

**Import table data to Hive (Create table & load data)**

sqoop-import --options-file /root/cts/sqoop\_param --table Employee -m 1 --hive-import --create-hive-table

**Import table data to Hive (table already exists, Only load data)**

sqoop-import --options-file /root/cts/sqoop\_param --table Employee -m 1 --hive-import --hive-table emp\_hive

**Arguements:**

--create-hive-table Fail if the target hive table exists

--hive-import Import tables into Hive

--hive-overwrite Overwrite existing data in the Hive table.

--hive-table <table-name> To specify hive table name

--hive-database <database-name> To specify hive database name

--version To find the version

--hive-partition-key To specify partition column

**--import-all-tables:**

For the import-all-tables tool to be useful, the following conditions must be met:

1. Each table must have a single-column primary key.

2. You must intend to import all columns of each table.

3. You must not intend to use non-default splitting column, nor impose any conditions via a WHERE clause.

sqoop import-all-tables --connect jdbc:mysql://localhost/cts1 --username root --password cloudera --create-hive-table --hive-import --hive-database cts1 -m 1

This tool imports a set of tables from an RDBMS to HDFS. Data from each table is stored in a separate directory in HDFS.

**Incremental data load to HDFS with last last value**

sqoop-import --options-file /root/cts/conn\_file --table orders --check-column order\_id --incremental append --last-value 100;

**Incremental data load to hive table with last last value**

sqoop import --options-file /root/cts/sqoop\_param --table orders1 --incremental append --check-column order\_id --last-value 104 --hive-database cts\_db --hive-import --hive-table orders1;

**Incremental data load to hive table with last modified date**

sqoop-import --options-file /root/cts/conn\_file --table orders --check-column order\_date --incremental lastmodified --last-value '2014-01-25' --target-dir cts\_hdfs/abc

**Export data to MySQL table from hadoop(hdfs)**

sqoop-export --options-file conn\_file --table orders\_mysql --export-dir /user/root/orders

**Export data to MySQL table from hadoop(hive)**

sqoop-export --options-file conn\_file --table order\_mysql --hive-table orders\_hive --hive-database cts\_db;

**Export Failure Cases:**

1. Exports may fail for a number of reasons:
2. Loss of connectivity from the Hadoop cluster to the database (either due to hardware fault, or server software crashes)
3. Attempting to INSERT a row which violates a consistency constraint (for example, inserting a duplicate primary key value)
4. Attempting to parse records using incorrect delimiters
5. Capacity issues (such as insufficient RAM or disk space)

**codegen generates the .java, .class & .jar file for the sqoop job executed.**

sqoop codegen --connect jdbc:mysql://localhost/retail\_db --username root --password cloudera --table ordr;

**eval prints the output of the query on the screen**

sqoop eval --connect jdbc:mysql://localhost/retail\_db --username root --password cloudera --query "select \* from emp where deptno=20";

**Writing a sqoop Job**

sqoop job --create myjob -- import --connect jdbc:mysql://localhost/cts\_db --username root --password cloudera --table orders --hive-import --hive-database cts\_db;

**sqoop job -list;**

**sqoop job --exec myjob;**

**sqoop job -delete myjob;**