@ mysql :

\*copy data from one table to another ;

mysql > insert into emp2 select \* from emp;

\*union tables

mysql> select \* from emp

-> union all

-> select \* from emp3;

\*union tables but schema is changed

mysql> select name,city from tab union all select name,city from tab2;

*@ sqoop : import export queries*

**1. import without primary key- mappers - 1**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --table emp2 -m 1 --target-dir '/user/cloudera/sqoop\_data/without\_pk'

**2. import with fields terminated by tab; \t**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --table emp2 -m 1 --target-dir '/user/cloudera/sqoop\_data/with\_tab1' --fields-terminated-by '\t'

**3. import with where then spically 'where' condition, other wise all rows data will be fetched.**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --table emp2 -m 1 --where 'sal>=55000' --target-dir '/user/cloudera/sqoop\_data/where1'

**4. import with where- use of and{both condition should be true} with 2 cloumns**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --table emp3 -m 1 --where 'sal>=55000 and gen="f"' --target-dir '/user/cloudera/sqoop\_data/where2'

**5. if use of only few columns then**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --table emp3 -m 1 --columns name,sal,dno --target-dir '/user/cloudera/sqoop\_data/column'

**6. use of cloumns with where**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --table emp3 -m 1 --columns name,sal,dno --where 'sal>=55000' --target-dir '/user/cloudera/sqoop\_data/column2'

**7. Imported Failed: column not found: sal\*0.1, if we want to done some maths expression that time it's consider as the coloumn.**

**error - command.**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --table emp3 -m 1 --columns name,sal,sal\*0.1 --target-dir '/user/cloudera/sqoop\_data/column3'

**8. for maths experssion - query**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --query 'select id,name,sal,sal\*0.1,sal\*0.2,sal+(sal\*0.2)-(sal\*0.1),gen,dno from emp3 where $CONDITIONS' -m 1 --target-dir '/user/cloudera/sqoop\_data/query'

output:

1,manish,50000,5000.0,10000.0,55000.0,m,11

2,niki,55000,5500.0,11000.0,60500.0,m,13

**9. if use, 'where' in query**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --query 'select \* from emp3 where dno in (11,13) and gen="m" and $CONDITIONS' -m 1 --target-dir '/user/cloudera/sqoop\_data/query1'

**10. if union's the 2 tables -schema should be same**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --query 'select \* from emp where $CONDITIONS union all select \* from emp3 where $CONDITIONS' -m 1 --target-dir '/user/cloudera/sqoop\_data/query2'

**11. union tables but schema is changed**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --query 'select name,city from tab where $CONDITIONS union all select name,city from tab2 where $CONDITIONS' -m 1 --target-dir '/user/cloudera/sqoop\_data/union'

**12. if we wants to union two tables but schema is different**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --query 'select id,name,age,"NOCITY" as city from samp1 where $CONDITIONS union all select id,name,0 as age,city from samp2 where $CONDITIONS' -m 1 --target-dir '/user/cloudera/sqoop\_data/union1'

**13.if we wants to join two tables column**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --query 'select id,name,sal,gen,dname,loc from emp3 e join dept d where $CONDITIONS and e.dno=d.dno' -m 1 --target-dir '/user/cloudera/sqoop\_data/union3'

**14. group by - ETL bcoz queries done by rdbms.**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --query 'select gen, sum(sal) from emp3 where $CONDITIONS group by gen' -m 1 --target-dir '/user/cloudera/sqoop\_data/group'

output:

f,269000

m,271000

**15. query start with "\_\_"-it's takes as java code so we have to use '\' for the $**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --query "select \* from emp where gen='m' and \$CONDITIONS" -m 1 --target-dir '/user/cloudera/sqoop\_data/mask'

**16. --append**

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --query "select \* from emp where gen='m' and \$CONDITIONS" -m 1 --target-dir '/user/cloudera/sqoop\_data/mask' --append

\* --bounary-query

\* if primary id present then, no need to set --split-by and also mapper is not compulsary.

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --table emp --boundary-query "select 2,4 from emp" --target-dir '/user/cloudera/sql\_data1/boundary3'

\*compulsary to add --spilt-by with --boundary-query if the primary key not declaired & mapper is not compulsary.

$ sqoop import --connect 'jdbc:mysql://localhost:3306/sql\_data' --username root --password cloudera --table emp3 --split-by id --boundary-query "select 2,4 from emp3" --target-dir '/user/cloudera/sql\_data1/boundary2'

**17.** $ sqoop import --connect 'jdbc:mysql://localhost:3306/db' --username root --passwrod cloudera --table tbname -m 5 --split-by id --target-dir 'path'

Importing a Table

Sqoop tool ‘import’ is used to import table data from the table to the Hadoop file system as a text file or a binary file.

## Importing into Target Directory

We can specify the target directory while importing table data into HDFS using the Sqoop import tool.

## Import Subset of Table Data

We can import a subset of a table using the ‘where’ clause in Sqoop import tool. It executes the corresponding SQL query in the respective database server and stores the result in a target directory in HDFS.

## Incremental Import

Incremental import is a technique that imports only the newly added rows in a table. It is required to add ‘incremental’, ‘check-column’, and ‘last-value’ options to perform the incremental import.

Import all tables

**Note** − If you are using the import-all-tables, it is mandatory that every table in that database must have a primary key field.

## Create Job (--create)

Here we are creating a job with the name **myjob**, which can import the table data from RDBMS table to HDFS. The following command is used to create a job that is importing data from the **employee** table in the **db** database to the HDFS file.

## Verify Job (--list)

**‘--list’** argument is used to verify the saved jobs. The following command is used to verify the list of saved Sqoop jobs.

## Inspect Job (--show)

**‘--show’** argument is used to inspect or verify particular jobs and their details. The following command and sample output is used to verify a job called **myjob**.

## Execute Job (--exec)

**‘--exec’** option is used to execute a saved job. The following command is used to execute a saved job called **myjob**.

## Select Query Evaluation

Using eval tool, we can evaluate any type of SQL query. Let us take an example of selecting limited rows in the **employee** table of **db** database. The following command is used to evaluate the given example using SQL query.

## Insert Query Evaluation

Sqoop eval tool can be applicable for both modeling and defining the SQL statements. That means, we can use eval for insert statements too. The following command is used to insert a new row in the **employee** table of **db**database.

$ sqoop eval \

--connect jdbc:mysql://localhost/db \

--username root \

-e “INSERT INTO employee VALUES(1207,‘Raju’,‘UI dev’,15000,‘TP’)”

Codegen

It generates DAO class in Java, based on the Table Schema structure. The Java definition is instantiated as a part of the import process. The main usage of this tool is to check if Java lost the Java code. If so, it will create a new version of Java with the default delimiter between fields.

Creates the jar file & shows us.

$ cd /tmp/sqoop-hadoop/compile/9a300a1f94899df4a9b10f9935ed9f91/

$ ls

emp.class

emp.jar

emp.java