NAME:- DUBEY KARAN SANJEEV
CLASS:- B.E - 4
ROLL NO:- 04
BATCH:- A

# EXPERIMENT NO. 2

## AIM :- To transfer data between Hadoop and RDBMS using Sqoop tool.

**THEORY:**

Sqoop is a tool designed to transfer data between Hadoop and relational databases or mainframes. You can use Sqoop to import data from a relational database management system (RDBMS) such as MySQL or Oracle or a mainframe into the Hadoop Distributed File System (HDFS), transform the data in Hadoop MapReduce, and then export the data back into an RDBMS.

Sqoop automates most of this process, relying on the database to describe the schema for the data to be imported. Sqoop uses MapReduce to import and export the data, which provides parallel operation as well as fault tolerance.

With Sqoop, you can *import* data from a relational database system or a mainframe into HDFS. The input to the import process is either database table or mainframe datasets. For databases, Sqoop will read the table row-by-row into HDFS. For mainframe datasets, Sqoop will read records from each mainframe dataset into HDFS. The output of this import process is a set of files containing a copy of the imported table or datasets. The import process is performed in parallel. For this reason, the output will be in multiple files. These files may be delimited text files (for example, with commas or tabs separating each field), or binary Avro or SequenceFiles containing serialized record data.

A by-product of the import process is a generated Java class which can encapsulate one row of the imported table. This class is used during the import process by Sqoop itself. The Java source code for this class is also provided to you, for use in subsequent MapReduce processing of the data. This class can serialize and deserialize data to and from the SequenceFile format. It can also parse the delimited-text form of a record. These abilities allow you to quickly develop MapReduce applications that use the HDFS-stored records in your processing pipeline. You are also free to parse the delimiteds record data yourself, using any other tools you prefer.

After manipulating the imported records (for example, with MapReduce or Hive) you may have a result data set which you can then *export* back to the relational database. Sqoop's export process will read a set of delimited text files from HDFS in parallel, parse them into records, and insert them as new rows in a target database table, for consumption by external applications or users.

Sqoop includes some other commands which allow you to inspect the database you are working with. For example, you can list the available database schemas (with the sqoop-list-databases tool) and tables within a schema (with the sqoop-list-tables tool). Sqoop also includes a primitive SQL execution shell (the sqoop-eval tool).

Most aspects of the import, code generation, and export processes can be customized. For databases, you can control the specific row range or columns imported. You can specify particular delimiters and escape characters for the file-based representation of the data, as well as the file format used. You can also control the class or package names used in generated code. Subsequent sections of this document explain how to specify these and other arguments to Sqoop.

**CONCLUSION:**

Thus with sqoop commands one can transfer data in structured format to HDFS platform and vice-versa.

| Program formation/ Execution/ ethical practices (06) | Timely Submission and Documentation (02) | Viva Answer (02) | Experiment Marks (10) | Teacher Signature with date |
|---|---|---|---|---|
| | | | | |

**For Swithcing to mysql terminal from hadoop**
**mysql -u root -pcloudera;**

**For importing tables**
**sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --password cloudera --username root --table departments;**
**check the import table using hadoop fs -ls**
**For Relocation**
[training@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/db11 --username root --table student --m 1 --target-dir /user/training/sqoop1 (creates directly no need of mkdir)
hadoopfs -ls /user/training/studen4813
hadoop fs -cat /user/training/studen4813/part-m-00000

**For Copying only particular record**
sqoop import --connect jdbc:mysql://localhost/db11 --username root --table student --m 1 --where "name='aaaaa'" --target-dir /user/training/student4513
hadoop fs -ls /user/training/student4513
hadoop fs -cat /user/training/student4513/part-m-00000

**For copying all the tables from DB to HDFS**

**When we want to transfer, all the tables from local DB to the HDFS, it is necessary to have the primary key in each table.**
[training@localhost ~]$ sqoop import-all-tables --connect jdbc:mysql://localhost/dbname -username root

**For appending the data**
[training@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/bhavarthv --username root --table st44 --m 1 --incremental append --check-column id1 --last-value 3 --target-dir /user/training/st44
hadoop fs -ls /user/training/st44

**To copy content of hdfs into blank table in MySql**
**Now, We will create table schema in MySql for table st66**
create table st66 (id int primary key, name varchar(100), location varchar(50));

**content of table st44 from hdfs is copied into mysql table st66**
[training@localhost ~]$ sqoop export -connect jdbc:mysql://localhost/bhavarthv --username root --table st66 --m 1 --export-dir /user/training/st44

**Mysql to hive**
In hive,
first create database test2
then execute following command
[training@localhost ~]$ sqoop import --connect jdbc:mysql://localhost:3306/bhavarthv --table st44 --username root --hive-import --hive-table test_2.test22 --warehouse-dir /user/hive/warehouse
then go to hive and check

**to check the databases of mysql from sqoop**
[training@localhost ~]$ sqoop list-databases --connect jdbc:mysql://localhost/ --username root

**To check the tables of mysql from sqoop**
[training@localhost ~]$ sqoop list-tables --connect jdbc:mysql://localhost/bhavarthv --username root

**Sqoop -Eval**

[training@localhost ~]$ sqoop eval --connect jdbc:mysql://localhost/bhavarthv --username root --query "select * from st55"

**SQOOP Export**

Let us take an example of the employee data in file, in HDFS. The employee data is available in **emp_data** file in 'emp/' directory in HDFS. The **emp_data**is as follows.

```
1201, gopal,    manager, 50000, TP
1202, manisha,  preader, 50000, TP
1203, kalil,    php dev, 30000, AC
1204, prasanth, php dev, 30000, AC
1205, kranthi,  admin,   20000, TP
1206, satish p, grp des, 20000, GR
```

It is mandatory that the table to be exported is created manually and is present in the database from where it has to be exported.The following query is used to create the table 'employee' in mysql command line.

```
$ mysql
mysql> USE db;
mysql> CREATE TABLE employee (
   id INT NOT NULL PRIMARY KEY,
   name VARCHAR(20),
   deg VARCHAR(20),
   salary INT,
   dept VARCHAR(10));
```

The following command is used to export the table data (which is in **emp_data**file on HDFS) to the employee table in db database of Mysql database server.

```
$ sqoop export \
--connect jdbc:mysql://localhost/db \
--username root \
--table employee \
--export-dir /emp/emp_data
```

The following command is used to verify the table in mysql command line.

```
mysql>select * from employee;
```

If the given data is stored successfully, then you can find the following table of given employee data.

```
+------+--------------+------------+------------------+--------+
| Id   | Name         | Designation | Salary          | Dept   |
+------+--------------+------------+------------------+--------+
| 1201 | gopal        | manager    | 50000            | TP     |
| 1202 | manisha      | preader    | 50000            | TP     |
| 1203 | kalil        | php dev    | 30000            | AC     |
| 1204 | prasanth     | php dev    | 30000            | AC     |
| 1205 | kranthi      | admin      | 20000            | TP     |
| 1206 | satish p     | grp des    | 20000            | GR     |
+------+--------------+------------+------------------+--------+
```