**INTRODUCTION**1.

The traditional application management system, that is, the interaction of applications with relational database using RDBMS, is one of the sources that generate Big Data. Such Big Data, generated by RDBMS, is stored in **RelationalDatabase Servers** in the relational database structure.

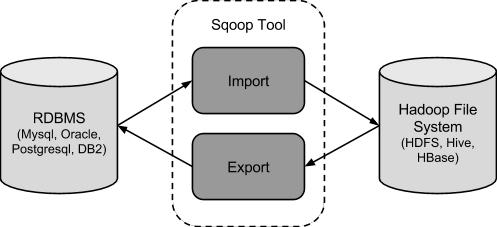
When Big Data storages and analyzers such as MapReduce, Hive, HBase, Cassandra, Pig, etc. of the Hadoop ecosystem came into picture, they required a tool to interact with the relational database servers for importing and exporting the Big Data residing in them. Here, Sqoop occupies a place in the Hadoop ecosystem to provide feasible interaction between relational database server and Hadoop’s HDFS.

**Sqoop:** “SQL to Hadoop and Hadoop to SQL”

Sqoop is a tool designed to transfer data between Hadoop and relational database servers. It is used to import data from relational databases such as MySQL, Oracle to Hadoop HDFS, and export from Hadoop file system to relational databases. It is provided by the Apache Software Foundation.

**How Sqoop Works?**

The following image describes the workflow of Sqoop.



**Sqoop Import**

The import tool imports individual tables from RDBMS to HDFS. Each row in a table is treated as a record in HDFS. All records are stored as text data in text files or as binary data in Avro and Sequence files.

**Sqoop Export**

The export tool exports a set of files from HDFS back to an RDBMS. The files given as input to Sqoop contain records, which are called as rows in table. Those are read and parsed into a set of records and delimited with user-specified delimiter.

**Syntax**

The following syntax is used to import data into HDFS

$ sqoop import (generic-args) (import-args)

Example in Mysql

Add a database called books, enter:  
mysql> **CREATE DATABASE books;**

Now, database is created. Use a database with use command, type:  
mysql> **USE books;**

Next, create a table called authors with name, email and id as fields:  
mysql> **CREATE TABLE authors (id INT, name VARCHAR(20), email VARCHAR(20));**

To display your tables in books database, enter:  
mysql> **SHOW TABLES;**

Finally, add a data i.e. row to table books using INSERT statement, run:  
mysql> **INSERT INTO authors (id,name,email) VALUES(1,"Vivek","xuz@abc.com");**Try to add few more rows to your table:  
mysql> **INSERT INTO authors (id,name,email) VALUES(2,"Priya","p@gmail.com");**  
mysql> **INSERT INTO authors (id,name,email) VALUES(3,"Tom","tom@yahoo.com");**

### **sqoop list commands**

Run the commands on the Unix prompt, on the node where you have sqoop installed.

#### List-databases

Lists databases in your mysql database.

$ sqoop list-databases --connect jdbc:mysql://192.168.80.134:3306/employees --username root

13/05/31 16:45:58 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.

information\_schema

employees

test

#### list-tables

Lists tables in your mysql database.

$ sqoop list-tables --connect jdbc:mysql://192.168.80.134:3306/employees --username root

13/05/31 16:45:58 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.

departments

dept\_emp

dept\_manager

employees

employees\_exp\_stg

employees\_export

salaries

titles

| **Argument** | **Description** |
| --- | --- |
| --append | Append data to an existing dataset in HDFS |
| --as-avrodatafile | Imports data to Avro Data Files |
| --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 |
| -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 |
| --where <where clause> | WHERE clause to use during import |
| -z,--compress | Enable compression |
| --compression-codec <c> | Use Hadoop codec (default gzip) |

**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.

--target-dir <new directory in HDFS>

The following command is used to import the **authors** table from MySQL database server to HDFS.

**sqoop** **import --connect jdbc:mysql://192.168.80.132:3306/books --username root --table authors --target-dir /sqoop/mysqlstage -m 1**

**sqoop import --connect jdbc:mysql://192.168.80.134:3306/books1 --username root --password hadoop --table authors --target-dir /mysqoop/mysqlstaging -m 1**

**sqoop import --connect jdbc:mysql://192.168.80.134:3306/books1 --username root --password hadoop --table authors --split-by id -target-dir /mysqoop/mysqlstaging1 -m 2**

**Example:**

**75 hadoop fs -copyToLocal /user/hue/empdata/sample.txt /DATA**

**76 hadoop fs -appendToFile /data/sample.txt /training/sample.txt**

**77 mysql**

**78 sqoop list-databases --connect jdbc:mysql://192.168.164.128:3306 --username root**

**80 sqoop list-tables --connect jdbc:mysql://192.168.164.128:3306/aprb2 --username root**

**81 sqoop import --connect jdbc:mysql://192.168.164.128:3306/aprb2 --username root --table authors --target-dir /sqoop/apr2b2/authors/ -m 1**

**82 hadoop fs -ls /sqoop/aprb2/authors**

**83 hadoop fs -ls /**

**84 hadoop fs -ls /sqoop**

**85 hadoop fs -ls /sqoop/apr2b2/authors**

**86 hadoop fs -cat /sqoop/apr2b2/authors/**

**87 hadoop fs -cat /sqoop/apr2b2/authors/part-m-00000**

**88 hisotry**

**89 history**

**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.

| **Argument** | **Description** |
| --- | --- |
| --check-column (col) | Specifies the column to be examined when determining which rows to import. |
| --incremental (mode) | Specifies how Sqoop determines which rows are new. Legal values for mode includeappend and lastmodified. |
| --last-value (value) | Specifies the maximum value of the check column from the previous import. |

Sqoop supports two types of incremental imports: append and lastmodified. You can use the --incrementalargument to specify the type of incremental import to perform.

You should specify append mode when importing a table where new rows are continually being added with increasing row id values. You specify the column containing the row's id with --check-column. Sqoop imports rows where the check column has a value greater than the one specified with --last-value.

An alternate table update strategy supported by Sqoop is called lastmodified mode. You should use this when rows of the source table may be updated, and each such update will set the value of a last-modified column to the current timestamp. Rows where the check column holds a timestamp more recent than the timestamp specified with --last-value are imported.

**Add one more record into authors table**

**INSERT INTO authors (id,name,email) VALUES(4,"tonia","toni@gmail.com")**

--incremental <mode> --check-column <column name>

--last-value <last check column value>

Run cat command before import:

hadoop fs -cat /sqoop/myswlstage/\*

result is :

1,Vivek,xuz@abc.com

2,Priya,p@gmail.com

3,Tom,tom@yahoo.com

sqoop import --connect jdbc:mysql://192.168.80.132:3306/books --username root --table authors --target-dir /sqoop/myswlstage -m 1 --incremental append --check-column id --last-value 3

Run cat command after import:

Result is:

1,Vivek,xuz@abc.com

2,Priya,p@gmail.com

3,Tom,tom@yahoo.com

4,tonia,toni@gmail.com

The above code will insert all the new rows based on the last value.

\*\*In hdfs in the **same old target directory ,a new file will be created**with all new records.

**Now we can think of second case where there are updates in rows**

+------+------------+----------+------+------------+

| sid | city | state | rank | rDate |

+------+------------+----------+------+------------+

| 101 | Chicago | Illinois | 1 | 2015-01-01 |

| 101 | Schaumburg | Illinois | 3 | 2014-01-25 |

| 101 | Columbus | Ohio | 7 | 2014-01-25 |

| 103 | Charlotte | NC | 9 | 2013-04-22 |

| 103 | Greenville | SC | 9 | 2013-05-12 |

| 103 | Atlanta | GA | 11 | 2013-08-21 |

| 104 | Dallas | Texas | 4 | 2015-02-02 |

| 105 | Phoenix | Arzona | 17 | 2015-02-24 |

+------+------------+----------+------+------------+

Here we use incremental lastmodified where we will fetch all the updated rows based on date.

sqoop import --connect jdbc:mysql://localhost:3306/ydb --table yloc --username root -P --check-column rDate --incremental lastmodified --last-value 2014-01-25 --target-dir yloc/loc

**\*\* we have to specify a new target directory**

**Sqoop Job**

Sqoop job creates and saves the import and export commands. It specifies parameters to identify and recall the saved job. This re-calling or re-executing is used in the incremental import, which can import the updated rows from RDBMS table to HDFS

**Syntax**

The following is the syntax for creating a Sqoop job

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

**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 **authors** table in the **books** database to the HDFS file.

sqoop job --create myjob -- import --connect jdbc:mysql://192.168.80.132:3306/books --username root --table authors --target-dir /sqoop/mysqlstage3 -m 1

Note: "--" and a space before the import.

**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.

$ sqoop job --list

Available jobs:

Myjob

**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**.

sqoop job --show myjob

It shows the tools and their options, which are used in **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**.

$ sqoop job --exec myjob