CSCE 5300 Introduction to Big Data and Data Science

Lesson 5

Apache Sqoop

Dr. Zeenat Tariq

Overview

- Installation
- What is Sqoop
- Why Sqoop
- Relation with Hadoop
- How it works
- Who uses
- Sqoop import use case
- Sqoop export use case

Sqoop Installation (Linux)
(Your own Practice)

Prerequisites

- verify Java installation
  - $ java –version
- If not found, install java
- Verifying Hadoop Installation
  - $ hadoop version
- If not found, install Hadoop

Installation

- Download
http://mirrors.sonic.net/apache/sqoop/1.4.7/

- Installation
https://www.tutorialspoint.com/sqoop/sqoop_installation.htm

Sqoop Installation on Linux

- Download

http://mirrors.sonic.net/apache/sqoop/1.4.7/

- Extract downloaded tar file

$tar -xzvf sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz

- move it to “/usr/lib/sqoop” directory.

$mv sqoop-1.4.7.bin__hadoop-2.6.0 /usr/lib/sqoop

- Configuring bashrc - apped the following lines to ~/**.bashrc** file

export SQOOP_HOME=/usr/lib/sqoop export PATH=$PATH:$SQOOP_HOME/bin

- *$ source ~/.bashrc*

- To configure Sqoop with Hadoop, you need to edit the **sqoop-env.sh** file, which is placed in the **$SQOOP_HOME/conf** directory. First of all, Redirect to Sqoop config directory and copy the template file using the following command

```
$ cd $SQOOP_HOME/conf
$ mv sqoop-env-template.sh sqoop-env.sh
```

- Open sqoop-env.sh and edit the following lines

```
export HADOOP_COMMON_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=/usr/local/hadoop
```

- download **mysql-connector-java-5.1.30.tar.gz**

http://ftp.ntu.edu.tw/MySQL/Downloads/Connector-J/

- Extract tar file

tar -zxf mysql-connector-java-5.1.30.tar.gz

- Move to /usr/lib/sqoop/lib

mv mysql-connector-java-5.1.30-bin.jar /usr/lib/sqoop/lib

- Verifying Sqoop

$ cd $SQOOP_HOME/bin

$ sqoop-version

Reference

https://www.tutorialspoint.com/sqoop/sqoop_installation.htm

What is Sqoop

•Designed for efficiently transferring bulk data between Hadoop and structured data stores such as relational databases

•Sqoop imports data from external structured databases into HDFS or related data stores like Hive

•Sqoop can also used to export from Hadoop to external structured databases

•Sqoop works with relational databases such as : Teradata, Oracle, MySQL, PostgreSQL and etc..

Why Sqoop

- Many organizations deploy Hadoop to analyze vast stream of information
- When Big Data storages and analyzers of the Hadoop ecosystem came into picture
- We need some tool to import and export from big data storages to external world
- Here, Sqoop occupies a place in the Hadoop ecosystem to provide feasible interaction between relational database server and Hadoop’s HDFS

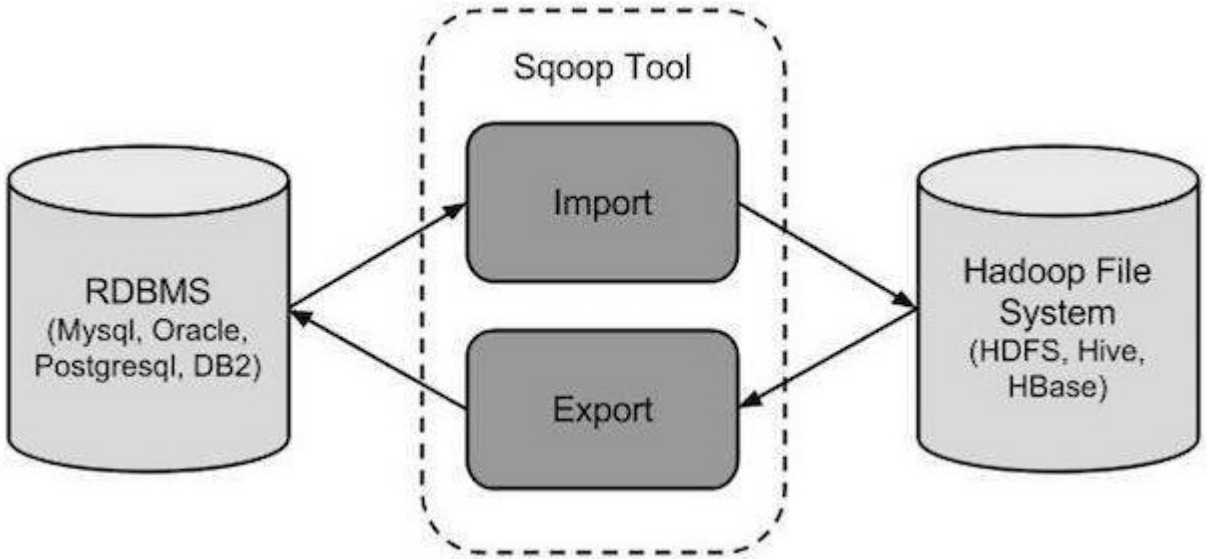
Hadoop- Sqoop?

- Hadoop is great for storing massive data in terms of volume using HDFS
- It provides scalable processing environment for structured and unstructured data
- But's it is batch-oriented processing thus not suitable for interactive query applications
- Sqoop act like ETL tool used to copy data between HDFS and SQL databases

What Sqoop does

- Allows data imports/ exports
- Parallelize data transfer
- Copies data quickly
- Makes data analysis more efficient
- Mitigates excessive loads

How Sqoop works?



Who uses Sqoop

- Online Marketer Coupons.com uses Sqoop to exchange data between Hadoop and IBM Netezza data warehouse
- They are querying the structure data and pipes the results to Hadoop
- Education company The Apollo group also uses the software to inject the results from Hadoop jobs back to relational databases

MySQL to HDFS

✓ Hadoop allows data transfer from and to MySQL to HDFS.

Example:

```
sqoop import \
--connect jdbc:mysql://<ip address>\<database name>
--table <mysql_table name>
--username <username_for_mysql_user> --password <Password>
-m <number of mappers to run>
--target-dir <target directory where we need copied data>
```

MySQL to HDFS

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

Selective Imports

✓ We can import data from the table to Hadoop according to the query that we want to run on MySQL.

Example:

```
sqoop import
--connect jdbc:mysql://<ip address>\<database name>
--query 'SELECT <table_name>.* FROM <table_name> where $CONDITIONS '
----username <username_for_mysql_user> --password <Password>
 -m <number of mappers to run>
--target-dir <target directory where we need copied data>
```

Selective Imports

✓ Other points to note:

- ✓ We can do joins on the Database table preloading into the Database.
- ✓ We can have multiple mappers only when the primary key is defined.
- ✓ If primary key is not specified, most of the processing take place in reducers.
- ✓ If primary key is defined, most of the processing happens in mappers.
- ✓ We can specify the columns according to which splits have to be made.

Joining Table

Example:

```
sqoop import
--connect jdbc:mysql://<ip address>\<database name>
--username <username_for_mysql_user> --password <Password>
--target-dir <target directory where we need copied data>
--query 'select a.*,b.* from <table1> a, <table2> b where a.[cloumn1] > 20,b.[column2]<10 and
$CONDITIONS'
-m <number of mappers to run>
```

✓ **Where Clause**

Example:

```
sqoop import \
--connect jdbc:mysql://<ip address>\<database name>
--table <mysql_table name>
--username <username_for_mysql_user> --password <Password>
 -m <number of mappers to run>
--target-dir <target directory where we need copied data>
--where Column_name >20
```

Option File

We can give above option in the configuration file also so that there will be no change in the code once deployed and we can do changes only in the configuration files.

Sqoop
--options-file <file_path>
--table <table_name>

The file will look like this:

Controlling Import Process

By default,

✓ Sqoop will load data to a directory whose name is same as the parent table name.
✓ The target directories are created in the parent directory of the user initiating Sqoop command.

We can control the directory where the target directory is created by:

```
--warehouse-dir <the directory name to store all table directory>
```

We can give names to directory corresponding to each table, instead of making it same as the table name:

```
-- target-dir <name of directory instead of giving table name>
```

We can give our own output file data's delimiters:

```
--fields-terminated-by '\t'
--lines-terminated-by '\n'
```

Controlling Import Process

Sqoop is preconfigured to map most of the SQL types to appropriate Java or Hive representatives. However, we can override this by:

```
--map-column-java id=String,value=Integer
```

We can append data to the output table directory with:

```
--append
```

We can enable compression in the output file by giving –z option with the command:

Import to Hive

✓ **Sqoop can be used to directly import data from RDBMS to HIVE format:**

Example:

```
sqoop import
--connect jdbc:mysql://<ip address>\<database name>
--username <username_for_mysql_user> --password <Password>
--table <mysql_table name>
--hive-import
-m 1
```

All previously defined options can be used with this command. The data will be saved by defa
warehouse folder of HIVE.

Import to Hive

HIVE Argument:

Argument	Description
--hive-home <dir>	Override $HIVE_HOME
--hive-import	Import tables into Hive (Uses Hive's default delimiters if none are set)
--hive-overwrite	Overwrite existing data in the Hive table.
--create-hive-table	If set, then the job will fail if the target Hive table exits. By default this property is false.
--hive-table <table-name>	Sets the table name to use when importing to Hive.
--hive-drop-import-delims	Drops *\n*, *\r*, and *\01* from string fields when importing to Hive.
--hive-partition-key	Name of a Hive field to partition are sharded on
--hive-partition-value <v>	String-value that serves as partition key for this imported into Hive in this job.

Export from HDFS to Mysql

✓ Sqoop also provides the feature to export data from Hadoop based systems to RDMS. The target table must already exist in the database.

Example:

```
sqoop export
--connect jdbc:mysql://<ip address>\<database name>
--username <username_for_mysql_user> --password <Password>
--table <mysql_table name>
--export-dir <the directory name from where data has to be exported>
-m <number of mappers to run>
```

Export from HDFS to MySql

✓ Export Control Argument:

Argument	Description
--direct	Use direct export fast path.
--export-dir <dir>	HDFS source path for the export.
-m,--num-mappers <n>	Use *n* map tasks to export in parallel.
--table <table-name>	Table to populate.
--update-key <col-name>	Anchor column to use for updates.
--input-null-string <null-string>	The string to be interpreted as null for string columns.
--input-null-non-string <null-string>	The string to be interpreted as null for non-string columns.
--staging-table <staging-table-name>	The table in which data will be staged before being inserted into the destination table.
--clear-staging-table	Indicates that any data present in the staging table can be deleted.

Export to Hive

- ✓ For Exporting data from Sqoop to any HIVE table, we have to export the data to the warehouse directory of HIVE. We can export data directly into pre-created table.

Example:

```
sqoop export
--connect jdbc:mysql://<ip address>\<database name>
--username <username_for_mysql_user> --password <Password>
--table <mysql_table name>
--export-dir /user/hive/warehouse/<table name>
--input-fields-terminated-by '\001'
-m <number of mappers>
```

- ✓ Export to Hive (Update)
  - If you specify the --update-key argument, Sqoop will instead modify an existing dataset in the database.
  - Each input record is treated as an UPDATE statement that modifies an existing row.
  - The row that a statement modifies is determined by the column name specified with --update-key.

Performance

✓ Data Transferred:
Number of records : 136,343,533
Size of Records : 12.770 Gb

✓ RDBMS:
MySQL
Installed on a single node machine.

✓ Total time taken:
Import data from MySQL to HDFS: 16 min
Import data from MySQL to Hive: 16 min
Export data from HDFS to MySQL : 20 min
Export data from Hive to MySQL : 20 min

Also note that performance depends on the read speed of the underlying RDMS also.

In Class Exercise

Part - 1

Part 1

1. Create table in MySQL and Import into HDFS through Sqoop
2. Export table from HDFS to MySQL

MySQL

Step:1

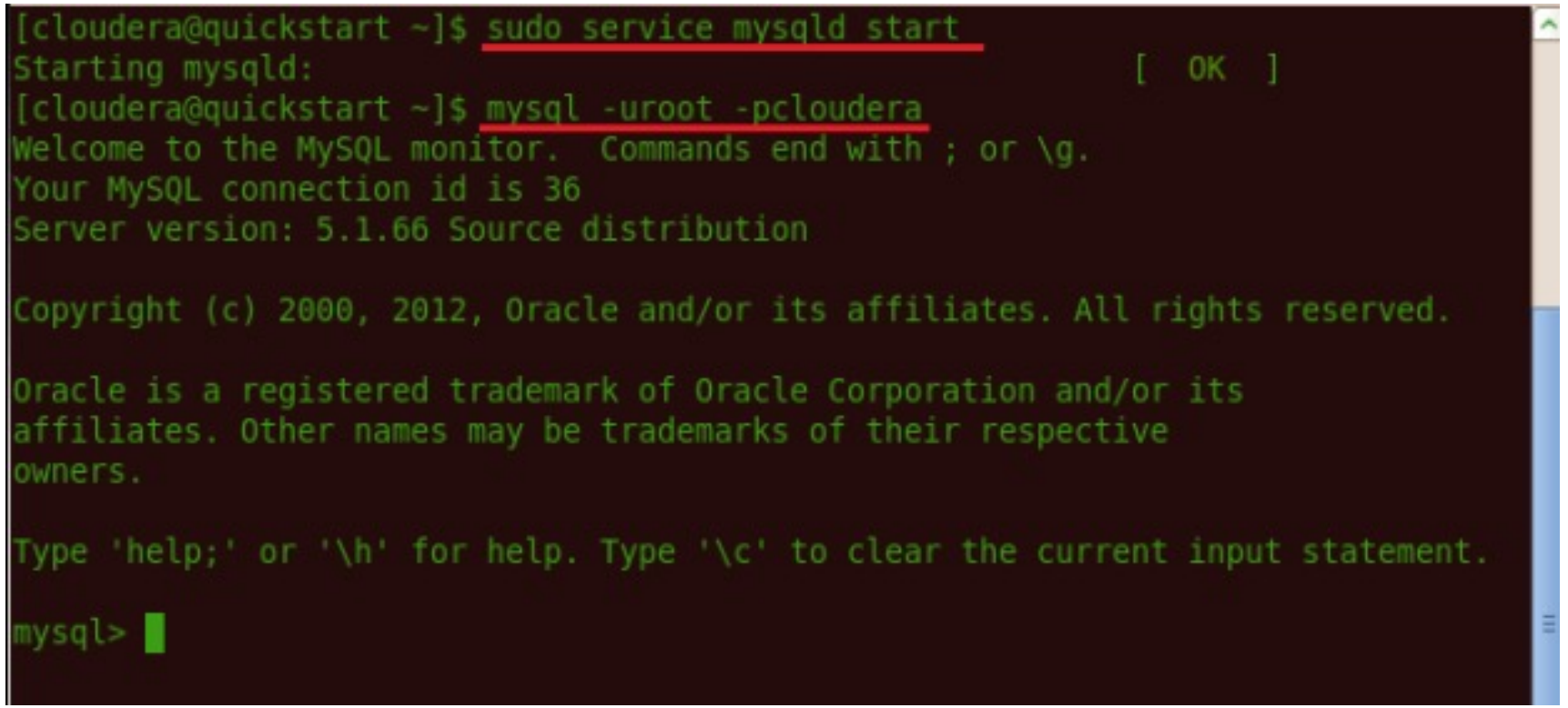
Start the MySQL service with the below command:

sudo service mysqld start

And enter MySQL shell using the below command:

mysql -u root -pcloudera

MySQL


[cloudera@quickstart ~]$ sudo service mysqld start
Starting mysqld: [OK]
[cloudera@quickstart ~]$ mysql -uroot -pcloudera
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 36
Server version: 5.1.66 Source distribution
Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>

MySQL

Step:2

Command to list database if already existing:

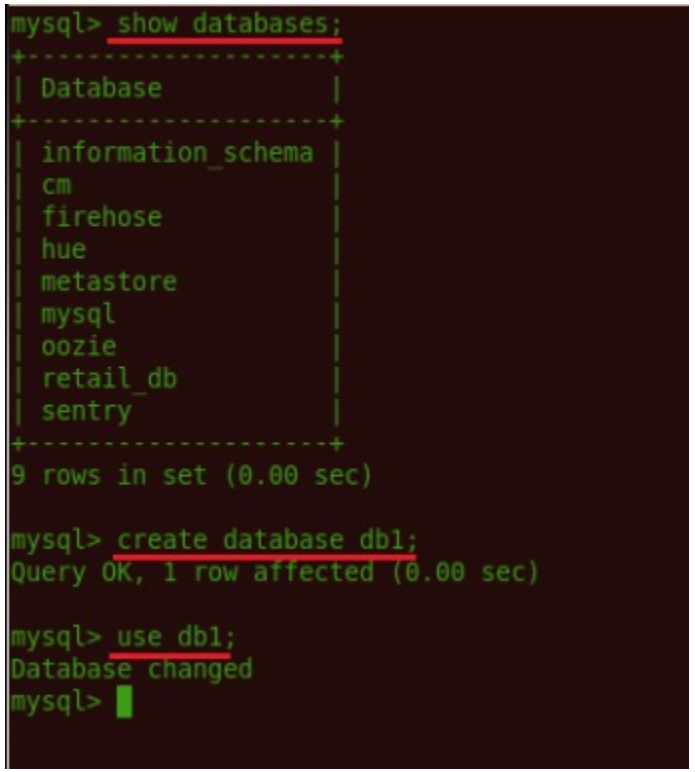
show databases;

Command to create a new database:

create database db1;

Command for using the database:

use db1;



Step:3

Also creating table, inserting values inside table is done using the following syntax.

create table <table name>(column name1, column name 2);

insert into <table name> values(column1 value1, column2 value1);

```
mysql> create table acad(emp_id INT NOT NULL AUTO_INCREMENT,emp_name VARCHAR(100
),emp_sal INT,PRIMARY KEY(emp_id));
Query OK, 0 rows affected (0.00 sec)

mysql> insert into acad values(5,"sanam",50000),(6,"opra",600000),(7,"yella",700
000);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from acad;
+--------+----------+---------+
| emp_id | emp_name | emp_sal |
+--------+----------+---------+
|      5 | sanam    |   50000 |
|      6 | opra     |  600000 |
|      7 | yella    |  700000 |
+--------+----------+---------+
3 rows in set (0.00 sec)

mysql>
```

- The import tool imports individual tables from RDBMS to HDFS. Each row in a table is treated as a record in HDFS
- **Sql table**

```
mysql> select * from acad;
+-------+----------+
| empid | emp_name |
+-------+----------+
|     5 | maya     |
|     6 | vinu     |
+-------+----------+
2 rows in set (0.00 sec)
```

Sqoop - import

Importing a Table

sqoop import --connect jdbc:mysql://localhost/db1 --username root --password cloudera --table acad --m 1

```
[cloudera@quickstart Downloads]$ sqoop import --connect jdbc:mysql://localhost/db1 --username root --password cloudera --table acad --m 1
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/06/20 09:11:24 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
18/06/20 09:11:24 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
18/06/20 09:11:25 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
18/06/20 09:11:25 INFO tool.CodeGenTool: Beginning code generation
18/06/20 09:11:26 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `acad` AS t LIMIT 1
18/06/20 09:11:26 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `acad` AS t LIMIT 1
18/06/20 09:11:26 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-mapreduce
Note: /tmp/sqoop-cloudera/compile/a498b8e8645dcb0f21ab9394814ef5f2/acad.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/06/20 09:11:30 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-cloudera/compile/a498b8e8645dcb0f21ab9394814ef5f2/acad.jar
18/06/20 09:11:30 WARN manager.MySQLManager: It looks like you are importing from mysql.
18/06/20 09:11:30 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
18/06/20 09:11:30 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
18/06/20 09:11:30 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
18/06/20 09:11:30 INFO mapreduce.ImportJobBase: Beginning import of acad
18/06/20 09:11:30 INFO Configuration.deprecation: mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
18/06/20 09:11:31 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar
18/06/20 09:11:33 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps
18/06/20 09:11:33 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
18/06/20 09:11:34 WARN hdfs.DFSClient: Caught exception
```

MAC Note: sqoop import *–direct* --connect “jdbc:mysql://localhost/db1?useSSL=false” --username root --password cloudera --table acad --m 1

Sqoop - import

hadoop fs -ls

hadoop fs –ls acad/

hadoop fs -cat acad/*

```
18/06/20 09:12:05 INFO mapreduce.Job:  map 100% reduce 0%
18/06/20 09:12:05 INFO mapreduce.Job: Job job_1529469523822_0002 completed successfully
18/06/20 09:12:06 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=170929
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=87
                HDFS: Number of bytes written=14
                HDFS: Number of read operations=4
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Other local map tasks=1
                Total time spent by all maps in occupied slots (ms)=10328
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=10328
                Total vcore-milliseconds taken by all map tasks=10328
                Total megabyte-milliseconds taken by all map tasks=10575872
        Map-Reduce Framework
                Map input records=2
                Map output records=2
                Input split bytes=87
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=253
                CPU time spent (ms)=1470
                Physical memory (bytes) snapshot=113754112
                Virtual memory (bytes) snapshot=1510182912
                Total committed heap usage (bytes)=60751872
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=14
18/06/20 09:12:06 INFO mapreduce.ImportJobBase: Transferred 14 bytes in 33.2106 seconds (0.4216 bytes/sec)
18/06/20 09:12:06 INFO mapreduce.ImportJobBase: Retrieved 2 records.
```

```
[cloudera@quickstart Downloads]$ hadoop fs -ls
Found 3 items
drwxr-xr-x   - cloudera cloudera          0 2018-06-20 09:12 acad
drwxr-xr-x   - cloudera cloudera          0 2018-06-20 09:04 emp
-rw-r--r--   1 cloudera cloudera  456449889 2018-06-19 16:25 incidents.csv
[cloudera@quickstart Downloads]$ hadoop fs -ls acad/
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2018-06-20 09:12 acad/_SUCCESS
-rw-r--r--   1 cloudera cloudera         14 2018-06-20 09:12 acad/part-m-00000
[cloudera@quickstart Downloads]$ hadoop fs -cat acad/*
5,maya
6,vinu
```

Cont..

- **Importing into Target Directory**

```
bigdata@BIGDATA:~$ sqoop import \
> --connect jdbc:mysql://localhost/mysql \
> --username root \
> --password root \
> --table Persons \
> --m 1 \
> --target-dir queryresult
```

- **verify the imported data in /queryresult directory form Persons table.**

```
bigdata@BIGDATA:~$ $HADOOP_HOME/bin/hadoop fs -cat queryresult/part-m-*
89,nandigam,nag,bluffs,kansas
bigdata@BIGDATA:~$
```

Cont..

Import All Tables

Each table data is stored in a separate directory and the directory name is same as the table name

- **Syntax**

```
$ sqoop import-all-tables (generic-args) (import-args)
$ sqoop-import-all-tables (generic-args) (import-args)
```

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 rows in table.

- **Syntax**

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

- 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.

```
mysql> CREATE TABLE employee (     id INT,    lastname VARCHAR(20),     First
e VARCHAR(20),    address VARCHAR(10),    cityname VARCHAR(10));
Query OK, 0 rows affected (0.34 sec)
```

Cont..

- Command is used to export the table data from hadoop to Sql

```
bigdata@BIGDATA:~$ sqoop export --connect jdbc:mysql://localhost/mysql --username root --
password root --table employee --export-dir queryresult/part-m-00000
```

```
mysql> select * from employee;
+------+----------+-----------+---------+----------+
| id   | lastname | Firstname | address | cityname |
+------+----------+-----------+---------+----------+
|   89 | nandigam | nag       | bluffs  | kansas   |
+------+----------+-----------+---------+----------+
1 row in set (0.00 sec)
```

In Class Exercise

Part - 2

Create Hive Tables Using HQL

Table and Schema Creation through HQL Script

hive –f tables-schema.hql

Note:

1. Download the data (data/employees from SourceCode Link)
2. Change the location of data in file based on your data.
3. Comment lines (--) with DB in HQL File (3 lines)

Create Hive Tables Using HQL

```
hive> show tables;
OK
employees
sales
Time taken: 0.516 seconds, Fetched: 2 row(s)
hive> describe employees
    > ;
OK
name                    string
salary                  float
subordinates            array<string>
deductions              map<string,float>
address                 struct<street:string,city:string,state:string,zip:int>
Time taken: 0.23 seconds, Fetched: 5 row(s)
hive>
```

Target SQL Construction

```
hive> show tables;
OK
employees
sales
Time taken: 0.516 seconds, Fetched: 2 row(s)
hive> describe employees
    > ;
OK
name                    string
salary                  float
subordinates            array<string>
deductions              map<string,float>
address                 struct<street:string,city:string,state:string,zip:int>
Time taken: 0.23 seconds, Fetched: 5 row(s)
hive>
```

Can Complex Datatypes to represented in SQL ?

Hive Complex Data Types: http://hadooptutorial.info/hive-data-types-examples/

Easier Hive Table to mySQL

```
hive> CREATE TABLE emp ( empid INT, emp_name STRING) ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LINES TERMINATED BY '\n'
    > STORED AS TEXTFILE;
OK
Time taken: 0.186 seconds
hive> Show tables;
OK
emp
employees
sales
Time taken: 0.024 seconds, Fetched: 3 row(s)

hive> LOAD DATA INPATH 'acad/'
    > INTO TABLE emp;
Loading data to table default.emp
Table default.emp stats: [numFiles=1, totalSize=14]
OK
Time taken: 0.897 seconds

hive> select * from emp;
OK
5       maya
6       vinu
Time taken: 0.362 seconds, Fetched: 2 row(s)
```

Easier Hive Table to mySQL

Location of hive tables: (our existing *emp* is our interest)

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/
Found 3 items
drwxrwxrwx   - cloudera supergroup          0 2018-06-20 09:47 /user/hive/warehouse/emp
drwxrwxrwx   - cloudera supergroup          0 2018-06-20 07:59 /user/hive/warehouse/employees
drwxrwxrwx   - cloudera supergroup          0 2018-06-15 11:45 /user/hive/warehouse/sales
[cloudera@quickstart ~]$
```

Creating mysql target Table *empNew*

```
mysql> create table empNew(empid INT,emp_name VARCHAR(100));
Query OK, 0 rows affected (0.01 sec)
```

Exporting Table to MySQL empNew through sqoop

sqoop export --connect jdbc:mysql://localhost/db1 --username root --password cloudera --table empNew --export-dir /user/hive/warehouse/emp -m 1

```
[cloudera@quickstart ~]$ sqoop export --connect jdbc:mysql://localhost/db1 --username root --password cloudera --table e
mpNew --export-dir /user/hive/warehouse/emp -m 1
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/06/20 09:57:05 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
18/06/20 09:57:05 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P inst
ead.
18/06/20 09:57:06 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
18/06/20 09:57:06 INFO tool.CodeGenTool: Beginning code generation
18/06/20 09:57:07 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `empNew` AS t LIMIT 1
18/06/20 09:57:07 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `empNew` AS t LIMIT 1
18/06/20 09:57:07 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-mapreduce
Note: /tmp/sqoop-cloudera/compile/d44eda0f36d64e776a9f1576194bc4ca/empNew.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
```

```
18/06/20 09:57:39 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=170884
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=166
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=4
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
        Job Counters
                Launched map tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=7142
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=7142
                Total vcore-milliseconds taken by all map tasks=7142
                Total megabyte-milliseconds taken by all map tasks=7313408
        Map-Reduce Framework
                Map input records=2
                Map output records=2
                Input split bytes=149
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=105
                CPU time spent (ms)=940
                Physical memory (bytes) snapshot=129552384
                Virtual memory (bytes) snapshot=1508089856
                Total committed heap usage (bytes)=60751872
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
18/06/20 09:57:39 INFO mapreduce.ExportJobBase: Transferred 166 bytes in 25.4916 seconds (6.512 bytes/sec)
18/06/20 09:57:39 INFO mapreduce.ExportJobBase: Exported 2 records.
[cloudera@quickstart ~]$
```

Target mySQL Table

The Data from Hive Table has been transferred to the MySQL Table

```
mysql> select * from empNew;
+-------+----------+
| empid | emp_name |
+-------+----------+
|     5 | maya     |
|     6 | vinu     |
+-------+----------+
2 rows in set (0.00 sec)
```

In Class Exercise

Part - 3

Choose one of following datasets

1. Dividends
2. Employees
3. Shakespeare
4. Stocks
5. Twitter

Choose one of the tasks

1. Create Hive Table and export to MySQL
2. Create SQL Table and import as Hive

Form 3 intuitive questions from your data

1. Statistics
2. WordCount
3. Identifying pattern

Convert these questions to queries

References

- http://sqoop.apache.org
- https://www.tutorialspoint.com/sqoop/index.htm
- https://hortonworks.com/apache/sqoop/#section_1