**Apache Sqoop Lab**

**RDBMS Installation**

First, we need to install MySQL in our CentOS distribution (this works in Cloudera, however, cloudera has MySQL installed, so you can skip the installation steps if needed). *This changes between Unix distributions.*

1. Download RPM.

wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm

1. Install RPM.

sudo rpm -ivh mysql-community-release-el7-5.noarch.rpm

1. Install MySQL with yum through installed RPM.

sudo yum install mysql-server

1. Start the service.

sudo service mysqld start

1. Test the connection

mysql -u root –p

1. Your distribution my have a password for root on MySQL, you might need to reset it.

sudo service mysqld stop

sudo mysqld\_safe --skip-grant-tables >> safe\_mod.log &

mysql –u root

use mysql;

update user SET PASSWORD=PASSWORD("p4ssw0rd") WHERE USER='root';

flush privileges;

exit

1. Start the service again and login with new password.

sudo service mysqld start

mysql –u root –p

--Enter password set up in previous step.

\h

--Should prompt all available commands

**Create a new user and database**

1. Setup database and admin user.

CREATE DATABASE STUDENT\_DB;

USE STUDENT\_DB;

GRANT ALL ON STUDENT\_DB.\* TO STUDENT\_ADMIN@localhost IDENTIFIED BY'p4ssw0rd';

--Be careful, username is case sensitive AND database name also.

1. Connect to the new database/user.

EXIT;

mysql –u STUDENT\_ADMIN –p

USE STUDENT\_DB;

**Sqoop Export (HDFS -> MySQL)**

1. Move student.csv file into hdfs (get this dataset from hive or pig examples location)

hdfs dfs –copyFromLocal ~/LData/sqoop

hdfs dfs –copyFromLocal ~/LData/sqoop/student.csv HData/

--Make sure this directory only has this csv file

1. Create student table in MySQL.

CREATE TABLE STUDENT (

SSN VARCHAR(50),

FIRST\_NAME VARCHAR(50),

LAST\_NAME VARCHAR(50),

RACE VARCHAR(50),

AGE INT,

STATE VARCHAR(50));

DESCRIBE STUDENT;

1. Perform sqoop export

sqoop export \

--connect jdbc:mysql://localhost/STUDENT\_DB \

--username STUDENT\_ADMIN \

--password p4ssw0rd \

--table STUDENT \

--input-fields-terminated-by ‘,’ \

--export-dir HData/sqoop/ \

-m 1

**Note how there is an error with no visible “why”.** This is a good chance to look into Hadoop logs and learn how to do it.

* Go to the link provided in the output for the Hadoop Job.
* Got to the Mapper task, and check the output:

…

org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:793) at org.apache.hadoop.mapred.MapTask.run(MapTask.java:341) at org.apache.hadoop.mapred.YarnChild$2.run(YarnChild.java:164) at java.security.AccessController.doPrivileged(Native Method) at javax.security.auth.Subject.doAs(Subject.java:415) at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1917) at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:158) Caused by: java.lang.RuntimeException: **Can't parse input data: 'age' at STUDENT.\_\_loadFromFields(STUDENT.java:463) at STUDENT.parse(STUDENT.java:386)** at org.apache.sqoop.mapreduce.TextExportMapper.map(TextExportMapper.java:89) ... 10 more Caused by: java.lang.NumberFormatException: For input string: "age" at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65) at java.lang.Integer.parseInt(Integer.java:492) at java.lang.Integer.valueOf(Integer.java:582) at STUDENT.\_\_loadFromFields(STUDENT.java:455) ... 12 more

There is an issue with ‘age’ going into a numeric field of the database. Sqoop has no available option to skip headers, so we will have to cleanse the file in HDFS.

--Use the sed transformation command to remove first line

sed -i 1d LData/student.csv

hdfs dfs -rm HData/sqoop/student.csv

hdfs dfs -copyFromLocal ~/LData/student.csv HData/sqoop/

\*\*Re-run sqoop export command

…

INFO mapreduce.ExportJobBase: Exported 1000 records.

1. Check the content of data in MySQL.

SELECT \* FROM STUDENT LIMIT 100;

**Sqoop Import**

1. Create a view in MySQL which only holds the names of students and the state they are part of, sorted by state in ascending order.

CREATE OR REPLACE VIEW STUDENT\_SUMMARY AS

SELECT CONCAT\_WS(" ", FIRST\_NAME, LAST\_NAME) AS NAME, STATE FROM STUDENT ORDER BY STATE;

1. Now, let’s import **everything** from this view back to HDFS.

sqoop import \

--connect jdbc:mysql://localhost/STUDENT\_DB \

--username STUDENT\_ADMIN \

--password p4ssw0rd \

--table STUDENT\_SUMMARY \

--target-dir output/sqoop-all/ \

-m 1

…

INFO mapreduce.ImportJobBase: Retrieved 1000 records.

1. Now, let’s import only the students from texas using a different delimiter. This would involve the “--**where**" option.

sqoop import \

--connect jdbc:mysql://localhost/STUDENT\_DB \

--username STUDENT\_ADMIN \

--password p4ssw0rd \

--table STUDENT\_SUMMARY \

--target-dir output/sqoop-texas/ \

--fields-terminated-by ‘|’ \

--where “STATE=’Texas’“ \

-m 1

…

INFO mapreduce.ImportJobBase: Retrieved 109 records.

hdfs dfs –cat output/sqoop-texas/\*

1. Let’s do an **incremental** import. We are going to need a numeric column in order for Sqoop to be able to figure out what to incrementally update. Let’s transform the SSN column into a number.

CREATE OR REPLACE VIEW STUDENT\_NUMERIC AS

SELECT CAST(REPLACE(SSN,’-‘,’’) AS UNSIGNED) AS SSN\_NUMERIC, FIRST\_NAME, LAST\_NAME, AGE FROM STUDENT;

1. Perform the initial import until the specified value. The correct way to do this, is assuming that you have some data already in HDFS, and you are telling Sqoop where to continue appending. For now, just set up the last-value as the first.
   1. We will have to reuse this job, so let’s use the sqoop job command.

sqoop job --create student\_job

-- import \

--connect jdbc:mysql://localhost/STUDENT\_DB \

--username STUDENT \

--table STUDENT\_SUMMARY \

--target-dir output/sqoop-append/ \

--fields-terminated-by ‘|’ \

--incremental append

--check-column SSN\_NUMERIC

-m 1

…

--Check job creation

sqoop job --list

…

--Execute the job

sqoop job --exec student\_job

\*\*It will prompt you the database password

…

INFO mapreduce.ImportJobBase: Retrieved 1000 records.

INFO tool.ImportTool: --incremental append

INFO tool.ImportTool: --check-column SSN\_NUMERIC

INFO tool.ImportTool: --last-value 999978620

…

--Check HDFS

\*\* Look at the generated Java file.

1. We currently have all 1000 records in HDFS, now, let’s simulate that records got added into the table. We will add additional numbers to the SSN so they are unique and Sqoop can determine what’s new. We will add 100 more records.

CREATE OR REPLACE VIEW STUDENT\_NUMERIC AS

SELECT CAST(REPLACE(SSN,’-‘,’’) AS UNSIGNED) AS SSN\_NUMERIC, FIRST\_NAME, LAST\_NAME, AGE FROM STUDENT

UNION

(SELECT CAST(REPLACE(SSN,’-‘,’0’) AS UNSIGNED) AS SSN\_NUMERIC, FIRST\_NAME, LAST\_NAME, AGE FROM STUDENT LIMIT 100);

--This would be the command that we’d have to run for incremental updates with no job state stored.

sqoop import \

--connect jdbc:mysql://localhost/STUDENT\_DB \

--username STUDENT \

--password p4ssw0rd \

--table STUDENT\_SUMMARY \

--target-dir output/sqoop-append/ \

--fields-terminated-by ‘|’ \

--incremental append

--last-value 999978620

--check-column SSN\_NUMERIC

-m 1

--Since we have the job stored we can run it

sqoop job --exec student\_job

…

INFO mapreduce.ImportJobBase: Retrieved 100 records.

1. If data got deleted in HDFS, and you re-run the job, Sqoop will say that there are no new changes. This happens because Sqoop is keeping track of the key states in its own Warehouse. To fix this issues, you have to --delete the job and then --create again.

sqoop job --delete student\_job

--Then recreate it and re-run it.