Sqoop = Sql + Hadoop

SQL import export utility.

Sqoop is an open source software product developed by the Apache Software Foundation.

* Sqoop is a command-line interface application for transferring data between relational databases and hadoop.
* Sqoop works with relational databases such as Teradata, Netezza, Oracle, MySQL, Postgres, and HSQLDB.
* You can use Sqoop to ***import*** data from a relational database management system (RDBMS and Data Warehouses) such as MySQL or oracle into Hadoop Distributed File System (HDFS), transform the data in Hadoop MapReduce, and then ***export*** the data back into RDBMS.
* Sqoop uses ***MapReduce*** to import and export the data, which provides parallel processing.
* It is used for integration.
* Sqoop import/export goes with only mapper not reducer jobs.
* Sqoop by ***default launches 4 no of mappers.***

Usage

With Sqoop you can import data from a relational database system into HDFS.  
The input to the import process is a database table.  
Sqoop reads the table row-by-row into HDFS.  
The output of this import process is a set of files containing a copy of the imported table.  
The import process is performed in **Parallel** For this reason, the output will be in multiple files.

**IN REAL TIME PROJECTS WE ARE NOT AUTHORIZED TO CONNECT TO DATABASES (RDBMS) (upstream)  
WE MUST IMPORT/EXPORT DATA ONLY THROUGH SQOOP  
WE CANNOT GIVE EXPLICIT CONNECTIONS LIKE sudo mysql;**

Declaration

**sqoop list-databases --connect jdbc:mysql://localhost --username root**

**Keyword Action Qualifier connection URL username and sometimes**

**mysql database if asked password**

**Here jdbc is a driver**

**Here we are not mentioning the password because we are going with root user. So, we don’t mention the password.**

Cdh5 cloudera quickstart VM

Sqoop Import

[cloudera@quickstart ~]$ **sqoop list-databases --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera**

**sqoop list-databases --connect jdbc:mysql://quickstart.cloudera:3306/ --username root --password cloudera**

Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.  
Please set $ACCUMULO\_HOME to the root of your Accumulo installation.  
18/04/05 08:27:39 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.14.0  
18/04/05 08:27:39 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.  
18/04/05 08:27:39 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.  
information\_schema  
cm  
firehose  
hue  
metastore  
mysql  
nav  
navms  
**oct2017**  
oozie  
retail\_db  
rman  
sentry

Here **jdbc**: is the driver

**mysql:** is the database

**quickstart.cloudera:** is the hostname

**3306:** is the port number

**oct2017:** is the database name

**username:** is the root

**password:** is the cloudera

[cloudera@quickstart ~]$ **sqoop list-tables --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera**

Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.  
Please set $ACCUMULO\_HOME to the root of your Accumulo installation.  
18/04/05 08:35:47 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.14.0  
18/04/05 08:35:47 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.  
18/04/05 08:35:47 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.  
employees  
employees\_exp\_stg  
employees\_export

[cloudera@quickstart ~]$ **sqoop import --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera --table employees**

Employees is the tablename

Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.  
Please set $ACCUMULO\_HOME to the root of your Accumulo installation.  
18/04/05 08:47:29 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.14.0  
18/04/05 08:47:29 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.  
18/04/05 08:47:29 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.  
18/04/05 08:47:29 INFO tool.CodeGenTool: Beginning code generation  
18/04/05 08:47:30 INFO manager.SqlManager: Executing SQL statement: SELECT t.\* FROM `employees` AS t LIMIT 1

SQL checks whether the root user can access the table employee or not. If user is not able to access the query stops here itself.

18/04/05 08:47:30 INFO orm.CompilationManager: HADOOP\_MAPRED\_HOME is /usr/lib/hadoop-mapreduce  
Note: /tmp/sqoop-cloudera/compile/ddc7368d29a6d56e4d2bdda4cbc1f21e/employees.java uses or overrides a deprecated API.  
Note: Recompile with -Xlint:deprecation for details.  
18/04/05 08:47:34 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-cloudera/compile/ddc7368d29a6d56e4d2bdda4cbc1f21e/employees.jar  
18/04/05 08:47:34 WARN manager.MySQLManager: It looks like you are importing from mysql.

Here we are importing the data from mysql

18/04/05 08:47:34 WARN manager.MySQLManager: This transfer can be faster! Use the –direct  
18/04/05 08:47:34 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.

By using –direct, the execution becomes faster because it has some MySQL specific fast path

18/04/05 08:47:34 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)  
18/04/05 08:47:34 INFO mapreduce.ImportJobBase: Beginning import of employees  
18/04/05 08:47:34 INFO Configuration.deprecation: mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address  
18/04/05 08:47:35 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar  
18/04/05 08:47:36 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps  
18/04/05 08:47:36 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032  
18/04/05 08:47:38 WARN hdfs.DFSClient: Caught exception   
java.lang.InterruptedException  
 at java.lang.Object.wait(Native Method)  
 at java.lang.Thread.join(Thread.java:1281)  
 at java.lang.Thread.join(Thread.java:1355)  
 at   
org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)  
 at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:705)

at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:894)  
18/04/05 08:47:38 WARN hdfs.DFSClient: Caught exception   
java.lang.InterruptedException  
 at java.lang.Object.wait(Native Method)  
 at java.lang.Thread.join(Thread.java:1281)  
 at java.lang.Thread.join(Thread.java:1355)  
 at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)

at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:705)  
 at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:894)  
18/04/05 08:47:39 WARN hdfs.DFSClient: Caught exception   
java.lang.InterruptedException  
 at java.lang.Object.wait(Native Method)  
 at java.lang.Thread.join(Thread.java:1281)  
 at java.lang.Thread.join(Thread.java:1355)  
 at   
org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)  
 at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:705)  
 at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:894)  
18/04/05 08:47:39 INFO db.DBInputFormat: Using read commited transaction isolation  
18/04/05 08:47:39 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`emp\_no`), MAX(`emp\_no`) FROM `employees`

Sqoop doesn’t launch reducer jobs. It only launches mapper jobs.

This runs because we have given primary key as emp\_no . so, it checks for min and max values of emp. Here, we have 12 entries in the table. Min will be 1 and max will be 12. So, it performs simple math. No of records/mapper number. Here 12/4=3. 3 records will be distributed equally into 4mappers

18/04/05 08:47:39 INFO db.IntegerSplitter: Split size: 2; Num splits: 4 from: 111 to: 122  
18/04/05 08:47:39 INFO mapreduce.JobSubmitter: number of splits:4  
18/04/05 08:47:40 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1522937875767\_0001  
18/04/05 08:47:41 INFO impl.YarnClientImpl: Submitted application application\_1522937875767\_0001  
18/04/05 08:47:41 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application\_1522937875767\_0001/  
18/04/05 08:47:41 INFO mapreduce.Job: Running job: job\_1522937875767\_0001  
18/04/05 08:47:57 INFO mapreduce.Job: Job job\_1522937875767\_0001 running in uber mode : false  
18/04/05 08:47:57 INFO mapreduce.Job: map 0% reduce 0%  
18/04/05 08:48:30 INFO mapreduce.Job: map 25% reduce 0%  
18/04/05 08:48:31 INFO mapreduce.Job: map 50% reduce 0%  
18/04/05 08:48:32 INFO mapreduce.Job: map 75% reduce 0%  
18/04/05 08:48:33 INFO mapreduce.Job: map 100% reduce 0%

Sqoop only launches map jobs

18/04/05 08:48:34 INFO mapreduce.Job: Job job\_1522937875767\_0001 completed successfully  
18/04/05 08:48:34 INFO mapreduce.Job: Counters: 31

File System Counters  
 FILE: Number of bytes read=0  
 FILE: Number of bytes written=690352  
 FILE: Number of read operations=0  
 FILE: Number of large read operations=0  
 FILE: Number of write operations=0  
 HDFS: Number of bytes read=441  
 HDFS: Number of bytes written=432  
 HDFS: Number of read operations=16  
 HDFS: Number of large read operations=0  
 HDFS: Number of write operations=8  
 Job Counters   
 Killed map tasks=1  
 Launched map tasks=4  
 Other local map tasks=4  
 Total time spent by all maps in occupied slots (ms)=123304  
 Total time spent by all reduces in occupied slots (ms)=0  
 Total time spent by all map tasks (ms)=123304  
 Total vcore-milliseconds taken by all map tasks=123304  
 Total megabyte-milliseconds taken by all map tasks=126263296  
 Map-Reduce Framework  
 Map input records=12  
 Map output records=12  
 Input split bytes=441  
 Spilled Records=0  
 Failed Shuffles=0  
 Merged Map outputs=0  
 GC time elapsed (ms)=1770  
 CPU time spent (ms)=4350  
 Physical memory (bytes) snapshot=471605248  
 Virtual memory (bytes) snapshot=6040731648  
 Total committed heap usage (bytes)=243531776  
 File Input Format Counters   
 Bytes Read=0  
 File Output Format Counters   
 Bytes Written=432  
18/04/05 08:48:34 INFO mapreduce.ImportJobBase: Transferred 432 bytes in 57.7219 seconds (7.4842 bytes/sec)  
18/04/05 08:48:34 INFO mapreduce.ImportJobBase: Retrieved 12 records.

|  |
| --- |
| To view  [cloudera@quickstart ~]$ **hadoop fs -ls employees** Found 5 items  By default no of mappers is 4 so it is split into 4 files  -rw-r--r-- 1 cloudera cloudera 0 2018-04-05 08:48 employees/\_SUCCESS -rw-r--r-- 1 cloudera cloudera 108 2018-04-05 08:48 employees/part-m-00000 -rw-r--r-- 1 cloudera cloudera 108 2018-04-05 08:48 employees/part-m-00001 -rw-r--r-- 1 cloudera cloudera 108 2018-04-05 08:48 employees/part-m-00002 -rw-r--r-- 1 cloudera cloudera 108 2018-04-05 08:48 employees/part-m-00003  [cloudera@quickstart ~]$ **hadoop fs -cat employees/part-m-00000** 111,1989-08-01,abc,aaa,M,2010-01-11 112,1989-09-01,bbb,sss,M,2010-01-12 113,1989-07-01,ccc,xxx,M,2010-01-13 |

|  |
| --- |
| **sqoop import --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera --table employees –direct**  it executes faster than normal import |

Set the number of mappers while import. Here, we have given mapper as 2

|  |
| --- |
| **sqoop import --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera --table employees -m 2** |

We cannot import a table without a primary key from RDBMS, we get the error  
Error during import: No primary key could be found for table tablename. Please specify one with –split-by or perform a sequential import with ‘-m 1’

One way

|  |
| --- |
| **sqoop import --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera --table employees -m 1** |

Other way

|  |
| --- |
| **sqoop import --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera --table employees –split-by emp\_no**  after the –split-by we need to mention the column name from the table (to perform the bounding val query) |

***The best way is to use –split-by because it uses the concept of parallelism and makes the execution faster. (If we use -m 1, it has only 1 mapper it takes much time for execution)***

# Import all tables

|  |
| --- |
| sqoop import-all-tables --connect jdbc:mysql://quickstart.cloudera:3306/retail\_db --username root --password cloudera |

Importing to a target path

If we use –query we must definitely mention –split-by column name

|  |
| --- |
| [cloudera@quickstart ~]$ **sqoop import --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera --query 'select EMP\_NO,birth\_date,first\_name,last\_name,gender,hire\_date from employees where EMP\_NO>114 and $CONDITIONS' --split-by EMP\_NO --target-dir /user/training/sqoop\_target\_import** |

18/04/05 11:17:07 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.  
18/04/05 11:17:08 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.  
18/04/05 11:17:08 INFO tool.CodeGenTool: Beginning code generation  
18/04/05 11:17:11 INFO manager.SqlManager: Executing SQL statement: select EMP\_NO,birth\_date,first\_name,last\_name,gender,hire\_date from employees where EMP\_NO>114 and (1 = 0)

1 = 0 is never equal

18/04/05 11:17:28 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(EMP\_NO), MAX(EMP\_NO) FROM (select EMP\_NO,birth\_date,first\_name,last\_name,gender,hire\_date from employees where EMP\_NO>114 and (1 = 1) ) AS t1

1 = 1 is always equal Here, we have 2 queries inner and outer query

18/04/05 11:17:28 INFO db.IntegerSplitter: Split size: 1; Num splits: 4 from: 115 to: 122  
18/04/05 11:17:28 INFO mapreduce.JobSubmitter: number of splits:4  
18/04/05 11:17:29 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1522937875767\_0003  
18/04/05 11:17:30 INFO impl.YarnClientImpl: Submitted application application\_1522937875767\_0003  
18/04/05 11:17:30 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application\_1522937875767\_0003/  
18/04/05 11:17:30 INFO mapreduce.Job: Running job: job\_1522937875767\_0003  
18/04/05 11:17:50 INFO mapreduce.Job: Job job\_1522937875767\_0003 running in uber mode : false  
18/04/05 11:17:50 INFO mapreduce.Job: map 0% reduce 0%  
18/04/05 11:18:45 INFO mapreduce.Job: map 50% reduce 0%  
18/04/05 11:18:47 INFO mapreduce.Job: map 75% reduce 0%  
18/04/05 11:18:48 INFO mapreduce.Job: map 100% reduce 0%  
18/04/05 11:18:49 INFO mapreduce.Job: Job job\_1522937875767\_0003 completed successfully  
18/04/05 11:18:49 INFO mapreduce.Job: Counters: 31  
18/04/05 11:18:49 INFO mapreduce.ImportJobBase: Transferred 288 bytes in 86.9083 seconds (3.3138 bytes/sec)  
18/04/05 11:18:49 INFO mapreduce.ImportJobBase: Retrieved 8 records.

***Import and compress the data***

|  |
| --- |
| **sqoop import --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera --query 'select EMP\_NO,birth\_date,first\_name,last\_name,gender,hire\_date from employees where EMP\_NO>114 and $CONDITIONS' -z --split-by EMP\_NO --target-dir /user/training/sqoop\_target\_import1** |

Here compress can be declared as ***-z*** or ***--compress***

Output will be

|  |
| --- |
| [cloudera@quickstart ~]$ hadoop fs -ls /user/training/sqoop\_target\_import1  Found 5 items  -rw-r--r-- 1 cloudera supergroup 0 2018-04-05 11:35 /user/training/sqoop\_target\_import1/\_SUCCESS  -rw-r--r-- 1 cloudera supergroup 74 2018-04-05 11:35 /user/training/sqoop\_target\_import1/part-m-***00000.gz***  -rw-r--r-- 1 cloudera supergroup 73 2018-04-05 11:35 /user/training/sqoop\_target\_import1/part-m-***00001.gz***  -rw-r--r-- 1 cloudera supergroup 74 2018-04-05 11:35 /user/training/sqoop\_target\_import1/part-m-***00002.gz***  -rw-r--r-- 1 cloudera supergroup 74 2018-04-05 11:35 /user/training/sqoop\_target\_import1/part-m-***00003.gz*** |

How to view the content of gz files

Use ***text*** or gunzip

***[cloudera@quickstart ~]$ hadoop fs -text /user/training/sqoop\_target\_import1/part-m-00000.gz***

115,1989-05-01,eee,zzz,M,2010-01-15  
116,1989-04-01,fff,rrr,M,2010-01-16

Sqoop eval  
To know how many records are there. So that we can set the no of mappers. Suppose if there are millions of records it will be a problem for us. So, we can set the number of mappers to 20

|  |
| --- |
| **sqoop eval --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 \**  **> --username root \**  **> --password cloudera \**  **> --query "select count(\*) from employees"** |

The other use of eval is to perform any DML into the table  
DML- Insert, update, alter

|  |
| --- |
| **sqoop eval --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera -e "insert into employees values(5500000,'1977-08-8','Mouse','Mickey','M','1992-04-12')"** |

# Sqoop incremental pull

On day 1 the data is 1 ……1000  
day 2 1001……..1500  
day 3 1501……..2000

sqoop import –connect jdbc:mysql://quickstart.cloudera:3306/oct2017 –table employees –username root –password cloudera –check-column emp\_no --incremental append –last-value 4

# Sqoop uses --split-by for writing bounding vals query

Sqoop Export

Sqoop export

|  |
| --- |
| **sqoop export --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera --table employees\_export -m 4 --export-dir /user/training/sqoop\_target\_import**  here  --export-dir is the path where data is present i.e., which is sent to mysql database from hdfs  --table tablename is the sql table name which is to be imported into that |

Temp table

Perm table = insert into perm select \* from temp

First data is loaded into temp and then loaded to perm table and then clear the data in temp

|  |
| --- |
| **sqoop export --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera --table employees\_export --staging-table employees\_exp\_stg --clear-staging-table -m 4 --export-dir /user/training/sqoop\_target\_import** |

If any data is present in table employees\_exp\_stg it will be deleted, and new data is inserted

18/04/05 12:29:17 INFO mapreduce.ExportJobBase: Starting to migrate data from staging table to destination.  
18/04/05 12:29:17 INFO manager.SqlManager: Migrated 8 records from `employees\_exp\_stg` to `employees\_export`

What if we wanted to update the data on mysql using sqoop

|  |
| --- |
| **sqoop export --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera --table employees\_export --direct --update-key emp\_no --update-mode updateonly --export-dir /user/training/sqoop\_target\_import** |

In this case, the records are updated and if there are newly added records in hdfs will not be reflected in MySQL

In **db2** we have Upsert = update + insert where newly added records will also be reflected in the database

|  |
| --- |
| **sqoop export --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera --table employees\_export --direct --update-key emp\_no --update-mode upsert --export-dir /user/training/sqoop\_target\_import** |

# HIVE

## RDBMS – HIVE

Import from rdbms to hive

|  |
| --- |
| **sqoop import --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera --table employees --hive-table oct2017.employees --create-hive-table --hive-import --hive-home /usr/lib/hive --split-by EMP\_NO**  Here  Oct2017.employees is databasename.tablename to be created in hive |

## HIVE – RDBMS

|  |
| --- |
| **sqoop export --connect jdbc:mysql://quickstart.cloudera:3306/oct2017 --username root --password cloudera --table employees\_exp\_stg --export-dir /user/hive/warehouse/oct2017.db/employees -m 4 --input-fields-terminated-by '\001'**  Here  /user/hive/warehouse/oct2017.db/employees is the hive table  input-fields-terminated-by '\001' delimiter in hive is ctrla (ascii value is 001) |

| **Argument** | **Description** |
| --- | --- |
| --append | Append data to an existing dataset in HDFS |
| --as-avrodatafile | Imports data to Avro Data Files |
| --as-sequencefile | Imports data to SequenceFiles |
| --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 |
| --inline-lob-limit <n> | Set the maximum size for an inline LOB |
| -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 |
| --warehouse-dir <dir> | HDFS parent for table destination |
| --where <where clause> | WHERE clause to use during import |
| -z,--compress | Enable compression |
| --compression-codec <c> | Use Hadoop codec (default gzip) |
| --null-string <null-string> | The string to be written for a null value for string columns |
| --null-non-string <null-string> | The string to be written for a null value for non-string columns |