Web: Inceptez.com Mail: info@inceptez.com Call: 7871299810, 7871299817

SQOOP WORKOUTS

Step 1: Installing Sqoop

Untar the sqoop 1.4.6 tar package and move to the common directory and give the respective permissions.

```
cd ~/install

tar xvzf sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz

sudo mv sqoop-1.4.6.bin__hadoop-2.0.4-alpha /usr/local/sqoop
```

Step 2: Copy mysql-connector-java

```
cp -p ~/install/mysql-connector-java.jar /usr/local/sqoop/
```

Step 3: Verifying Sqoop

The following command is used to verify the Sqoop version.

```
sqoop-version
```

MYSQL (Preparation of Source) :

Start the MYSQL Service :

sudo service mysqld start

mysql -u root -p

Password: Root123$

Select the custdb database:

```
create database custdb;

use custdb;

CREATE TABLE customer (custid INT,firstname VARCHAR(20),lastname VARCHAR(20),city varchar(50),age int,createdt date,transactamt int );

insert into customer values(1,'Arun','Kumar','chennai',33,'2017-09-20',100000);

insert into customer values(2,'srini','vasan','chennai',33,'2017-09-21',10000);

insert into customer values(3,'vasu','devan','banglore',39,'2017-09-23',90000);

insert into customer values(4,'mohamed','imran','hyderabad',33,'2017-09-24',1000);

insert into customer values(5,'arun','basker','chennai',23,'2017-09-20',200000);

insert into customer values(6,'ramesh','babu','manglore',39,'2017-09-21',100000);

create table customer_bkp as select * from customer;

create table customer_bkp1 as select * from customer;

select * from customer;
```

SQOOP WORKOUTS (Open a separate linux terminal)

To List Databases which are in MySql

```
sqoop list-databases --connect jdbc:mysql://localhost --username root --password Root123$;
```

To List Tables from custdb database

```
sqoop list-tables --connect jdbc:mysql://localhost/custdb --username root --password Root123$;
```

Import Table from SQL to HDFS with 1 mapper:

```
sqoop import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ -table customer -m 1 \

--delete-target-dir ;
```

Check whether the below import works?

```
sqoop import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ -table customer -m 2;
```

```
sqoop import --connect jdbc:mysql://localhost/custdb --username root -P -table customer -m 3 \
--split-by custid --target-dir sqoop_import --delete-target-dir --direct;
```

```
sqoop import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ -table customer -m 3 \
--split-by city --append --fetch-size 100;
```

Fields terminated by & Lines terminated by

```
sqoop import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ -table customer -m 1 \
--target-dir imp_del --fields-terminated-by '~' --lines-terminated-by '\n' --delete-target-dir;
```

Controlling Import:

Using Where condition:

```
sqoop import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ --table customer -m 1 \
--where "city ='banglore' or  age>33" --target-dir filtered  --delete-target-dir;
```

Using free form query:

```
sqoop import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ \
--query " select custid,age,transactamt from customer where (city ='banglore' or  age>33) and \$CONDITIONS " \
--target-dir filtered  --delete-target-dir -m 2 --split-by custid;
```

Incremental import:

Execute the below insert in mysql

```
insert into customer values(7,'inceptez','tech','Chennai',3,'2017-09-28',10000);
```

```
sqoop import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ -table customer -m 1 \
```

--target-dir incrimport --incremental append --check-column custid --last-value 6

Whether the below import works?

**sqoop import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ -table customer -m 1 **
--target-dir incrimport --incremental append --check-column city --last-value 'pune'

Working with Saved Jobs:

**sqoop job --create myjob1 -- import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ **
--table customer --target-dir savedjob --delete-target-dir -m 1

list saved jobs:

sqoop job -list

sqoop job --exec myjob1
password:Root123$

Incremental Saved Jobs:

**sqoop job --create myjob2 -- import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ **
--table customer --target-dir savedjob1 --m 1 --incremental append --check-column custid --last-value 0

sqoop job --exec myjob2
Passwod : Root123$

Insert the below row into mysql:

insert into customer values (8,'Iz','tech','pune',4,'2017-09-28',10000);

Execute once again the same job, now only newly added custid 8 will be imported:

sqoop job --exec myjob2

sqoop job --delete myjob2

Update the job:

**sqoop job --create myjob2 -- import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ **
--table customer --target-dir savedjob1 --m 1 --incremental append --check-column custid --last-value 6

Execute once again the job, now custid from 7 will be imported:

sqoop job --exec myjob2

Export from HDFS to SQL:

Create table in MYSQL before running this command

```
CREATE TABLE customer_hdfs (custid INT,firstname VARCHAR(20),lastname VARCHAR(20),city varchar(50),age int,createdt date,transactamt int );

sqoop export --connect jdbc:mysql://localhost/custdb --username root --password Root123$ --table customer_hdfs \
--export-dir savedjob1
```

Which one of the below export works?

```
sqoop export --connect jdbc:mysql://localhost/custdb --username root --password Root123$ --table customer_hdfs \
--export-dir imp_del;

sqoop export --connect jdbc:mysql://localhost/custdb --username root --password Root123$ --table customer_hdfs \
--export-dir imp_del --fields-terminated-by '~' --lines-terminated-by '\n';
```

Incremental export

Update only mode:

```
delete from customer_hdfs;

insert into customer_hdfs select * from customer;

vi ~/part-m-00000
7~inceptez~tech~Chennai~3~2017-09-28~10000
8~iz~tech~Calcutta~5~2015-09-28~10000
9~a~srini~chennai~38~2016-08-08~13000
```

Note: Delete the last blank line in the above file

```
hadoop fs -put -f part-m-00000 imp_del/

sqoop export --connect jdbc:mysql://localhost/custdb --username root --password Root123$ --table customer_hdfs \
--export-dir imp_del --fields-terminated-by '~' --lines-terminated-by '\n' --update-key custid \
--update-mode updateonly;

select * from customer_hdfs ORDER BY 1;
```

Allow insert mode:

```
ALTER TABLE customer_hdfs ADD PRIMARY KEY (custid);
```

vi ~/part-m-00000
8~iz~tech~Mumbai~33~2017-09-28~10000
10~inceptez~technologies~chennai~33~2017-09-28~13000

Note: Delete the last blank line in the above file

hadoop fs -put -f part-m-00000 imp_del/

**sqoop export --connect jdbc:mysql://localhost/custdb --username root --password Root123$ --table customer_hdfs **
**--export-dir imp_del --fields-terminated-by '~' --lines-terminated-by '\n' --update-key custid **
--update-mode allowinsert;

SQOOP Best Practices and Performance tuning

Import:

1. Definate number of mappers: - m

a. More mappers can lead to faster jobs, but only up to a saturation point. This varies per table, job parameters, time of day and server availability.

b. Too many mappers will increase the number of parallel sessions on the database, hence affect source DB performance affecting the regular workload of the DB.

2. Use Direct mode for all available DBs.

a. Rather than using the JDBC interface for transferring data, the direct mode delegatesthe job of transferring data to the native utilities provided by the database vendor. For Eg. In the case of MySQL, the mysqldump and mysqlimport will be used for retrieving data from the database server or moving data back.

b. Binary format don't work in direct mode.

3. Splitting Data --split-by: Boundary Queries --boundary-query

a. By default, the primary key is used. Prior to starting the transfer, Sqoop will retrieve the min/max values for this column. Changed column with the --split-by parameter

b. Boundary Queries - What if your split-by column is skewed, table is not indexed or can be retrieved from another table?

If --split-by is not giving you the optimal performance you can use this to improve the performance further to Use a boundary query to create the splits using the option --boundary-query

Eg.

```
sqoop import --connect jdbc:mysql://localhost/custdb --username root -P \
--boundary-query "select min(custid), max(custid) from customers"
--query 'Select a.custid master_custid,a.firstname,a.age,a.city,b.custid
detail_custid,a.createdt,b.fulladdress,category,transactiondt,transactamt
from customers a join customer_details b
on a.custid=b.custid
WHERE $CONDITIONS'

--split-by a.custid

Select min(temp.id) , max(temp.id) from (Select a.custid master_custid,a.firstname,a.age,a.city,b.custid
detail_custid,a.createdt,b.fulladdress,category,transactiondt,transactamt
from customers a join customer_details b
on a.custid=b.custid) as temp

-- boundary-query
--boundary-query "select min(custid), max(custid) from customers"

$CONDITIONS
```

For Eg. Above query will execute parallel like this.
SELECT a.id, a.name, b.id, b.name FROM customers a join customer_details b on a.id = b.id where a.id BETWEEN 0 AND 10;

SELECT a.id, a.name, b.id, b.name FROM customers a join customer_details b on a.id = b.id where a.id BETWEEN 11 AND 20;
SELECT a.id, a.name, b.id, b.name FROM customers a join customer_details b on a.id = b.id where a.id BETWEEN 21 AND 30;

4. Compression
Compress or z When you configure the compress or z argument, you can compress the data approximately by 60% and reduce the amount of disk space required in the target. You can configure compression when the target storage is limited.

5. fetch-size
Specifies the number of entries that Sqoop can import at a time.
Use the following syntax: --fetch-size 100
Where 100 represents the number of entries that Sqoop must fetch at a time. Default is 1000.

Export:

6. Defining mappers --num-mapper

a. Number of simultaneous connections that will be opened against database. Sqoop will use that many processes to export data (each process will export slice of the data). Here you have to take care about the max open connections to your RDBMS, since this can overwhelm the RDBMS easily.

7. BATCH mode --batch

a. Sqoop performs export row by row if we don't leverage batch mode option.

b. Enabling batch mode will export more than one row at a time as batch of rows.

8. Specify the number of records to export -Dsqoop.export.records.per.statement=10

a. The above option will define how many number of rows should be used in each insert statements.
e.g. INSERT INTO xxx VALUES (), (), (), ...

9. Specify the number of records per transaction - -Dsqoop.export.statements.per.transaction=10

The above option will define how much number of statements should be used in each transaction.
e.g BEGIN; INSERT, INSERT, …. COMMIT

10. Data Consistency --staging-table

a. In order to provide the consistent data access for the users in end database, using a staging table, Sqoop will first export all data into this staging table instead of the main table that is present in the parameter --table. Sqoop opens a new transaction to move data from the staging table to the final destination, if and only if all parallel tasks successfully transfer data.

Sqoop Additional commands and Use Cases

Usecase 1:

Import All tables from a DB :

```
sqoop import-all-tables --connect jdbc:mysql://localhost/custdb --username root --password Root123$ \
--warehouse-dir  '/user/hduser/sqoop/testtables' -m 1
```

Import All tables other than excluded tables from a DB :

```
sqoop import-all-tables --connect jdbc:mysql://localhost/custdb --username root --password Root123$ \
--warehouse-dir  '/user/hduser/sqoop/testtables' --exclude-tables customer_bkp1 -m 1
```

Sqoop evaluation:

```
sqoop eval --connect jdbc:mysql://localhost/custdb --username root --password Root123$ \
--query "select * from customer "
```

Batch export:

```
sqoop export -Dsqoop.export.statements.per.transaction=10 --connect jdbc:mysql://localhost/custdb \
--username root --password Root123$ --table customer1 --export-dir savedjob1 --batch
```

Usecase 2 :

Create the below Tables MYSQL for Import:

```
use custdb;

CREATE TABLE customers (custid INT,firstname VARCHAR(20),lastname VARCHAR(20),city varchar(50),age
int,createdt date );

CREATE TABLE customer_details (custid INT,firstname VARCHAR(20),fulladdress VARCHAR(200),category
varchar(50),transactiondt date,transactamt int,createdt date);

ALTER TABLE customers ADD PRIMARY KEY(custid);

insert into customers values(1,'karthik','vijay','chennai',5,'2018-02-01');

insert into customers values(2,'arun','kumar','chennai',25,'2018-01-30');

insert into customers values(3,'vishwa','ajit','hyderabad',null,'2018-02-03');

insert into customers values(4,'bala','palani','bangalore',30,'2018-02-02');

insert into customer_details values(1,'karthik','3/2, jeyaram street, chrompet,Chennai','household','2018-02-01',4000,'2018-02-01');

insert into customer_details values(1,'karthik','3/2, jeyaram street, chrompet,Chennai 44','Automobile','2018-02-02',6000,'2018-02-02');

insert into customer_details values(1,'karthik','3/2, jeyaram street, chrompet,Chennai 44','Foods','2018-02-02',3000,'2018-02-02');

insert into customer_details values(1,'karthik','3/2, jeyaram street, chrompet,Chennai 44',null,'2018-02-02',1000,'2018-02-03');
```

insert into customer_details values(2,'arun','11, palayam blvd, broadway,Chennai 01','tools','2018-02-02',11000,'2018-02-03');

insert into customer_details values(2,'arun','11, palayam blvd, broadway,Chennai 01','electronics','2018-02-02',15000,'2018-02-04');

insert into customer_details values(3,'vishwa','1A, Elango nagar, Vadapalani,Chennai 33','clothes','2018-02-02',15000,'2018-02-04');

Select a.custid master_custid,a.firstname,b.custid detail_custid,a.createdt,a.age,category,transactamt from customers a join customer_details b on a.custid=b.custid;

Tables for Export:

CREATE TABLE customer_stage (custid INT,fullname VARCHAR(40), city varchar(50),age int,createdt date);

CREATE TABLE customer_exp (custid INT,fullname VARCHAR(40),city varchar(50),age int,createdt date);

ALTER TABLE customer_exp ADD PRIMARY KEY(custid);

Import Approach:

1. Import all columns of customer and customer_details data by joining custid between the 2 tables. customer_details can have columns as given.

2. Both custid should be named as master_custid and detail_custid from customer and customer_details respectively.

3. Use column boundary queries using customer.custid column, split using custid

4. Insert null values in category column and age columns import as NA and age a 0 respectively in the hdfs.

5. Store the output in cust_details hdfs directory.

6. Compress the imported data.

7. Use direct mode to transfer the entire content.

8. Define number of mappers as 3.

9. Use fetch size 100.

10. Once the sqoop is executed view the content in hdfs using the command:

hadoop fs -text cust_details/part-m-0000*.gz

Import Solution Command:

**sqoop import --connect jdbc:mysql://localhost/custdb --username root -P **
**--boundary-query "select min(custid), max(custid) from customers" --query 'Select a.custid master_custid,a.firstname,a.age,a.city,b.custid detail_custid,a.createdt,b.fulladdress,category,transactiondt,transactamt from customers a join customer_details b on a.custid=b.custid WHERE $CONDITIONS' --split-by a.custid --target-dir cust_details **

--null-non-string '0' --null-string 'NA' --compress --direct --num-mappers 3 --fetch-size 100 --delete-target-dir;

Export Approach:

1. Import only the subset of columns from the customer table to the HDFS (custid, concatenation of firstname and lastname,age).

2. Export only subset of the above imported columns to the customer_exp table specifying only these 3 columns.

3. Use batch mode for fast export with the sqoop.export.records.per.statement=5.

4. Use staging table to provide consistent load with the clear staging table option to clean the table before each load.

Export Solution Command:

```
sqoop import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ \
--query " select custid,concat(firstname,' ',lastname),age  from customers where \$CONDITIONS  " \
--target-dir cust_exp  --delete-target-dir -m 1;

sqoop export -Dsqoop.export.records.per.statement=5 --connect jdbc:mysql://localhost/custdb \
--username root --password Root123$ --table customer_exp --export-dir cust_exp --batch \
--staging-table customer_stage --clear-staging-table --columns custid,fullname,age
```

Usecase 3:

Sqoop Import Incremental last modified

```
create table customer_lastmodified as select * from customer;

alter table customer_lastmodified add upddt date;

update customer_lastmodified set upddt='2018-10-10' where custid <=4;
```

Import only upddt greater or equal to 2018-10-10 and load in the HDFS location incrimportlm

```
sqoop import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ \
-table customer_lastmodified -m 1 --target-dir incrimportlm --incremental lastmodified --check-column upddt \
--last-value 2018-10-10 --append

hadoop fs -cat incrimportlm/*

update customer_lastmodified set upddt='2018-10-11' where custid>4;
```

Import and append in the HDFS location incrimportlm, only upddt greater or equal to 2018-10-11

```
sqoop import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ \
-table customer_lastmodified -m 1 --target-dir incrimportlm --incremental lastmodified \
--check-column upddt --last-value 2018-10-11 --append


update customer_lastmodified set upddt='2018-10-12',city='hyderabad' where custid=8;

insert into customer_lastmodified values(9,'inceptez1','tech','hyd',3,'2017-09-28',10000,'2018-10-12');
```

Sqoop uses Reducer in the below case (Import and update hdfs data if custid exist and insert if custid is not exist in the HDFS location incrimportlm, only upddt greater or equal to 2018-10-12)

```
sqoop import --connect jdbc:mysql://localhost/custdb --username root --password Root123$ \
-table customer_lastmodified -m 1 --target-dir incrimportlm --incremental lastmodified \
--check-column upddt --last-value 2018-10-12 --merge-key custid --append
```

Usecase 4 :
Exporting multiple tables using stored procedure

1) Create file and type data into it

```
vi empinfo
101,raja,dept1,Accounts
102,vinay,dept2,Finanace
103,karthik,dept3,IT
104,bala,dept4,Marketing
```

Note: Delete the last blank line in the above file

2) Create folder and copy file into hadoop

```
hadoop fs -mkdir -p /user/sqoop/spexport
hadoop fs -copyFromLocal -f empinfo /user/sqoop/spexport
```

3) In MySQL create tables and stored procedure

```
mysql -u root -p
password: Root123$

use custdb;

create table empinfo(empid int, empname varchar(20));
create table deptinfo(deptid varchar(10),deptname varchar(20));

delimiter //

create procedure sp_insert_empdeptinfo (IN pid int, IN pname varchar(20), IN pdeptid varchar(10), IN
pdeptname varchar(20))
BEGIN
INSERT INTO empinfo(empid, empname) VALUES(pid, pname);
INSERT INTO deptinfo(deptid,deptname) values(pdeptid, pdeptname);
END //

delimiter ;
```

Export the data from the file to the mysql stored procedure to load data into two tables using SQOOP

```
sqoop export --connect jdbc:mysql://localhost/custdb --username root --password Root123$ \
--call sp_insert_empdeptinfo --export-dir /user/sqoop/spexport -m 1
```

5) In mysql,select both the table empinfo and deptinfo to see the records

```
select * from empinfo;
select * from deptinfo;
```