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

INCEPTEZ TECHNOLOGIES SQOOP WORKOUTS

Step 1: Installing Sqoop

The following commands are used to extract the Sqoop tar ball and move it to "/usr/lib/sqoop" directory.

```
cd ~/install
tar xvzf sqoop-1.4.5.bin__hadoop-2.0.4-alpha.tar.gz
sudo mv sqoop-1.4.5.bin__hadoop-2.0.4-alpha /usr/local/sqoop
```

Step 2: Configuring bashrc

You have to set up the Sqoop environment by appending the following lines to ~/.bashrc file:

```
vi ~/.bashrc
echo 'export SQOOP_HOME=/usr/local/sqoop'  >> ~/.bashrc
echo 'export PATH=$PATH:$SQOOP_HOME/bin' >> ~/.bashrc
```

The following command is used to execute ~/.bashrc file.

```
source ~/.bashrc
```

Step 3: Configuring Sqoop

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:

```
echo 'export HADOOP_COMMON_HOME=/usr/local/hadoop' >> sqoop-env.sh
echo 'export HADOOP_MAPRED_HOME=/usr/local/hadoop' >> sqoop-env.sh
```

Step 4: Download and Configure mysql-connector-java

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

Step 5: Verifying Sqoop

The following command is used to verify the Sqoop version.

```
source ~/.bashrc
sqoop-version
```

MYSQL (Preparation of Source) :
Login as mysql and start the Service :

su root
password : hduser
su mysql
service mysqld restart
mysql

Select the test database:
use test;
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,'2015-09-20',100000);
insert into customer values(2,'srini','vasan','chennai',33,'2015-09-21',10000);
insert into customer values(3,'vasu','devan','banglore',39,'2015-09-23',90000);
insert into customer values(4,'mohamed','imran','hyderabad',33,'2015-09-24',1000);
insert into customer values(5,'arun','basker','chennai',23,'2015-09-20',200000);
insert into customer values(6,'ramesh','babu','manglore',39,'2015-09-21',100000);

SQOOP WORKOUTS (Open a separate linux terminal)

To List Databases which are in MySql

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

To List Tables from test database

sqoop list-tables --connect jdbc:mysql://localhost/test --username root --password root;

Import Table from SQL to HDFS:

sqoop import --connect jdbc:mysql://localhost/test --username root --password root -table customer -m 1 ;

hadoop fs -rm -r /user/hduser/customer
sqoop import --connect jdbc:mysql://localhost/test --username root --password root -table customer -m 1 --direct;
hadoop fs -rm -r /user/hduser/customer
sqoop import --connect jdbc:mysql://localhost/test --username root --password root -table customer -m 1 --target-dir sqoop_import;

Check whether the below import works?

sqoop import --connect jdbc:mysql://localhost/test --username root --password root -table customer -m 2;

Import with –split-by option

sqoop import --connect jdbc:mysql://localhost/test --username root --password root -table customer -m 10 --split-by custid;
hadoop fs -rm -r /user/hduser/customer

sqoop import --connect jdbc:mysql://localhost/test --username root --password root -table customer -m 3 --split-by city;

Using Where condition:

sqoop import --connect jdbc:mysql://localhost/test --username root --password root --table customer -m 1 --where "city ='chennai'" --target-dir \filtered
hadoop fs -rm -r /user/hduser/customer

Incremental import:

insert into customer values(7,'md','irfan1','hyderabad',33,'2015-09-28',10000);
sqoop import --connect jdbc:mysql://localhost/test --username root --password root -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/test --username root --password root -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/test --username root --password root --table customer --target-dir savedjob -m 1

list saved jobs:

sqoop job -list

Execute

sqoop job --exec myjob1

password:root

Incremental Saved jobs:

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

sqoop job --exec myjob2

insert into customer values (8,'md','irfan1','pune',33,'2015-09-28',10000);

sqoop job --delete myjob2

sqoop job --create myjob2 -- import --connect jdbc:mysql://localhost/test --username root --password root --table customer --target-dir savedjob1 --m 1 --incremental append --check-column custid --last-value 7

sqoop job --exec myjob2

Export from HDFS to SQL: (Create table in MYSQL before running this commmand)

CREATE TABLE customer1 (custid INT,firstname VARCHAR(20),lastname VARCHAR(20),city varchar(50),age int,createdt date,transactamt int);

sqoop export --connect jdbc:mysql://localhost/test --username root --password root --table customer1 --export-dir savedjob1

Fields terminated by & Lines terminated by

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

Whether the below export works?

sqoop export --connect jdbc:mysql://localhost/test --username root --password root --table customer1 --export-dir imp_del;

Incremental export , insert else update

sqoop export --connect jdbc:mysql://localhost/test --username root --password root --table customer1 --export-dir imp_del --fields-terminated-by '~' --lines-terminated-by '\n';

sqoop export --connect jdbc:mysql://localhost/test --username root --password root --table customer1 --export-dir imp_del --fields-terminated-by '~' --lines-terminated-by '\n' --update-key custid --update-mode allowinsert;

--update-key custid
--update-mode updateonly
--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. Escape characters, type mapping, column and row delimiters may not be supported. Binary formats don't work.

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://.../...' \
    **--direct **
    --username uname --password pword \
    --hive-import \
    --hive-table query_import \
    **--boundary-query 'SELECT 0, MAX(id) FROM a' **
    --query 'SELECT a.id, a.name, b.id, b.name FROM a, b WHERE a.id = b.id AND **$CONDITIONS**'\
    --num-mappers 3
    **--split-by a.id **
    --target-dir /data/import \

    - - split-by

    Select min(a.id) , max(a.id) from (SELECT a.id, a.name, b.id, b.name FROM a, b WHERE a.id = b.id AND $CONDITIONS) as temp

    -- boundary-query

    SELECT 0, MAX(id) FROM a

4. **Using $CONDITIONS**
    a. $CONDITIONS is used by Sqoop process, it will replace with a unique condition expression internally to get the data-set. If you run a parallel import, the map tasks will execute your query with different values substituted in for $CONDITIONS.

For Eg. Above query will execute parallel like this.

SELECT a.id, a.name, b.id, b.name FROM a, b WHERE a.id = b.id AND **a.id BETWEEN 0 AND 10**;
SELECT a.id, a.name, b.id, b.name FROM a, b WHERE a.id = b.id AND **a.id BETWEEN 11 AND 20;**
SELECT a.id, a.name, b.id, b.name FROM a, b WHERE a.id = b.id AND **a.id BETWEEN 21 AND 30;**

Export:

5. **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.
6. **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.

7. **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 (), (), (), ...
8. **Specify the number of records per transaction - -Dsqoop.export.statements.per.transaction=10**
    a. The above option will define how much number of statements should be used in each transaction.
    e.g BEGIN; INSERT, INSERT, ... COMMIT
9. **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 simple Use Cases:

Import:

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

CREATE TABLE customer (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);

Select a.custid c_custid,a.firstname,b.custid cd_custid, createdt
from customer a join customer_details b
on a.custid=b.custid;

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.
4. Insert null values in category column and import as NA 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 10.

Export:

1. Export the content of the HDFS into customer_exp table that has same data structure as customer table.
2. Use batch mode for fast export.
3. Use staging table.
4. Perform insert else update in the mysql table.
5. Export only subset of columns (for eg. create a file with only 2 columns such as custid and firstname and export to the customer_exp table specifying only these 2 columns).