Big Data

Grow Bigger

POSTED BY

NIJANTHANRAVI
POSTED ON

NOVEMBER 14, 2017
POSTED UNDER

UNCATEGORIZED
COMMENTS

LEAVE A COMMENT

SQOOP Cheat sheet

Transferring an Entire Table:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities
```

Specifying a Target Directory

Sqoop offers two parameters for specifying custom output directories: **–target-dir** and **–warehouse-dir.**

–target-dir option imports the actual data into the specified directory

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--target-dir /etl/input/cities
```

–warehouse-dir option creates a directory with table name and imports data under it.

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--warehouse-dir /etl/input/
```

Importing Only a Subset of Data

Using –where option we can specify more than one condition but we should not use any aggregated queries

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--where "country = 'USA'"
```

Protecting Your Password

Sqoop execution that will read the password from standard input:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--table cities \
-P
```

Here's an example of reading the password from a file:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--table cities \
--password-file my-sqoop-password
```

Using a File Format Other Than CSV

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--as-sequencefile
```

Avro can be enabled by specifying the –as-avrodatafile parameter:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--as-avrodatafile
```

Compressing Imported Data

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--table cities \
--compress
```

Change compression format

```
sqoop import --compress \
--compression-codec org.apache.hadoop.io.compress.BZip2Codec
```

```
Compression codecs
Splittable        Not Splittable
BZip2, LZO        GZip, Snappy
```

Speeding Up Transfers:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--table cities \
--direct
```

Rather than using the JDBC interface for transferring data, the direct mode delegates the job of transferring data to the native utilities provided by the database vendor.

Sqoop has **direct support only for MySQL and PostgreSQL.**

Overriding Type Mapping:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--table cities \
--map-column-java id=Long
```

```
sqoop import --map-column-java c1=Float,c2=String,c3=String ...
```

Controlling Parallelism

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--num-mappers 10
```

Encoding NULL Values:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--null-string '\\N' \
--null-non-string '\\N'
```

Importing All Your Tables

```
sqoop import-all-tables \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop
```

```
sqoop import-all-tables \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--exclude-tables cities,countries
```

Incrementally Importing Mutable Data

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table visits \
--incremental append \
--check-column id \
--last-value 1
```

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table visits \
--incremental lastmodified \
--check-column last_update_date \
--last-value "2013-05-22 01:01:01"
```

Preserving the Last Imported Value:

```
sqoop job \
--create visits \
-- \
import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table visits \
--incremental append \
--check-column id \
--last-value 0
```

```
sqoop job --exec visits
sqoop job --list
sqoop job --delete visits
```

Overriding the Arguments to a Saved Job

```
sqoop job --exec visits -- --verbose
```

Sharing the Metastore Between Sqoop Clients

```
sqoop job
--create visits \
--meta-connect jdbc:hsqldb:hsql://metastore.example.com:16000/sqoop \
-- \
import \
--table visits
```

Free-Form Query Import:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--query 'SELECT normcities.id, \
countries.country, \
normcities.city \
FROM normcities \
JOIN countries USING(country_id) \
WHERE $CONDITIONS' \
--split-by id \
--target-dir cities
```

where $CONDITIONS placeholder in the where clause of your query.

Using Custom Boundary Queries:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--query 'SELECT normcities.id, \
countries.country, \
normcities.city \
FROM normcities \
JOIN countries USING(country_id) \
WHERE $CONDITIONS' \
--split-by id \
--target-dir cities \
--boundary-query "select min(id), max(id) from normcities"
```

Renaming Sqoop Job Instances:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--query 'SELECT normcities.id, \
countries.country, \
normcities.city \
FROM normcities \
JOIN countries USING(country_id) \
WHERE $CONDITIONS' \
--split-by id \
--target-dir cities \
--mapreduce-job-name normcities
```

Transferring Data from Hadoop:

```
sqoop export \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--export-dir cities

sqoop export \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--update-key id

sqoop export \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--update-key id \
--update-mode allowinsert
```

Importing Data Directly into Hive:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--hive-import
```

Using Partitioned Hive Tables

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--hive-import \
--hive-partition-key day \
--hive-partition-value "2013-05-22"
```

Replacing Special Delimiters During Hive Import:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--hive-import \
--hive-drop-import-delims

sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--hive-import \
--hive-delims-replacement "SPECIAL"
```

Importing Data into HBase:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--hbase-table cities \
--column-family world

sqoop import \
-Dsqoop.hbase.add.row.key=true \
--connect jdbc:mysql://mysql.example.com/sqoop \
--username sqoop \
--password sqoop \
--table cities \
--hbase-table cities \
--column-family world
```

Improving Performance When Importing into HBase:

Create your HBase table prior to running Sqoop import, and instruct HBase to create more regions with the parameter NUMREGIONS. For example, you can create the HBase table cities with the column family world and 20 regions using the following command:
hbase> create ‘cities’, ‘world’, {NUMREGIONS => 20, SPLITALGO => ‘HexString Split’}

Exporting into PostgreSQL Using pg_bulkload:

```
sqoop import \
--connect jdbc:postgresql://postgresql.example.com/database \
--username sqoop \
--password sqoop \
--connection-manager org.apache.sqoop.manager.PGBulkloadManager \
--table cities
```

Importing from Oracle:

```
sqoop import \
--connect jdbc:oracle:thin:@oracle.example.com:1521/ORACLE \
--username SQOOP \
--password sqoop \
--table KATHLEEN.cities
```

Faster Transfers with Oracle:

You should consider using OraOop, a specialized connector for Oracle developed

Exporting into Teradata:

```
sqoop export \
-Dsqoop.export.records.per.statement=1 \
--connect jdbc:teradata://teradata.example.com/DATABASE=database \
--username sqoop \
--password sqoop \
--table cities\
--export-dir cities
```