Using Sqoop to Import Data

Apache Sqoop is an open source tool that was originally created at Cloudera. Its name comes from the contraction of “SQL to Hadoop”; it moves data between a relational database management system (RDBMS) and HDFS. For example, it can import all the tables from a database, just one table, or just part of a table, such as specific columns or specific rows. It can also export data from HDFS to a relational database. You see more about some of these options in the next couple of readings.

Sqoop is a command-line tool offering several commands. The Sqoop import command is used to import the data in a single table in an RDBMS to a directory in HDFS. The following example will import all the columns from the customers table in the company database in MySQL. In the example below, the $ character represents the operating system shell (terminal) prompt, and the \ (backslash) character is used to continue the command on multiple lines.

```
$sqoop import \

    --connect jdbc:mysql://localhost/company \

    --username jdoe \

    --password bigsecret \

    --table customers
```

This command creates a subdirectory named customers in the user’s home directory in HDFS, and populates that subdirectory with files containing all the data from the customers table in the RDBMS. By default, Sqoop stores the data in plain text files, where each line of the file is one record from the table and the fields are separated by commas. These defaults can be changed by adding options, which are described in the next reading.

By default, Sqoop uses JDBC to connect to the database. However, depending on the database, there may be a faster, database-specific connector available, which you can use by using the --direct option.

If the table whose data you're importing does not have a primary key, then you should specify one using --split-by *column.* If you're going to split by a string column, or if the primary key for a table is a string column and you're using some newer versions of Sqoop (as of 1.4.7), include the setting

-Dorg.apache.sqoop.splitter.allow_text_splitter=true

immediately after the import command.

To import *all* the tables from a database into HDFS, use the Sqoop import-all-tables command. This example brings all the tables from the company database into HDFS.

$ sqoop import-all-tables \

--connect jdbc:mysql://localhost/company \

--username jdoe \

--password bigsecret

Try It!

The VM has tables in a MySQL database. Although these are already imported into the Hive metastore, do the following to re-import one, just to give you some practice using the Sqoop import command.

1. Open a terminal window, if you don't have one already, and execute the following command. The card_rank table you're importing doesn't have a primary key, so you need to specify one. The most reasonable one is rank, which is a text column, so include the

-Dorg.apache.sqoop.splitter.allow_text_splitter=true

setting as well.

```
$sqoop import \
    -Dorg.apache.sqoop.splitter.allow_text_splitter=true \
     --connect jdbc:mysql://localhost/mydb \
    --username training \
    --password training \
    --table card_rank \
    --split-by rank
```

2. Check that HDFS now has /user/training/card_rank, with at least one file in it.

3. Review one of the files to see how the fields are delimited. (You'll need that when you create the table.)

4. Although you've imported the data, you don't yet have a table for it. (The existing card_rank table is in the fun database, and its data is in /user/hive/warehouse/fun.db/card_rank.) Run a CREATE TABLE statement to create a table in the default database, default.card_rank, with the following columns. Be sure to use ROW FORMAT to specify the delimiter, and use a LOCATION clause to specify the location of the data as '/user/training/card_rank/'.

name	type
rank	STRING

value	TINYINT

5. Run a query on your new default.card_rank table or check the sample data from the data source panel in Hue, to see that your table does have the data you imported. *Note:*If you created the table in Impala, you should not need to refresh the metadata, because the data was there when you created the table.

6. You can now drop the default.card_rank table. (Be careful *not* to drop the *other* card_rank table.)