Revision: 11/16/2020

Document providing guidance about the project

Checking the data

To easily see the content of the tweets.csv, copy the file but change the file extension to txt like tweets.txt. Run Excel and open this text file. Excel will ask to use a delimiter to separate a row into columns. Specify the semi-colon as the delimiter for the 2016 Twitter data. By inspecting the data, you will know whether your import process imports all the data or not.

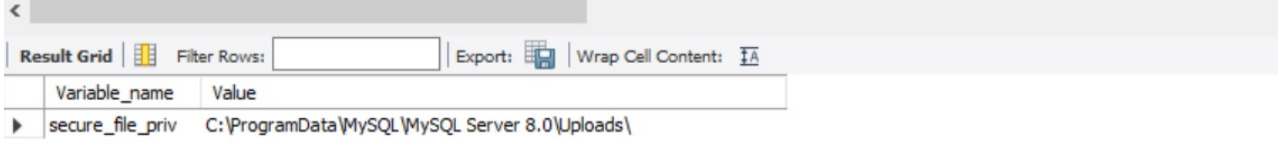
Extract Transform Load process

1. Load data from a csv file. There are multiple ways. The description here shows an example of importing the data from tweets.csv using the load data infile statement.

   Run forimport.sql to create a database named "test" and two relations, tweet and newtweet, in the newly created database. Then, import the data from tweets.csv.

   Use the load data infile statement. This method is the fast, but has a cryptic error message that is difficult to pinpoint where the error is. See https://dev.mysql.com/doc/refman/8.0/en/load-data.html. This requires that the data file you want to import is in a certain directory as indicated by the variable name 'secure_file_priv'. To know the directory name, execute the following statement.

   ```
   show variables like 'secure_file_priv';
   ```



On Windows 10, the output shows that the directory "C:\ProgramData\MySQL\MySQL Server 8.0\Uploads". Put the csv files in the directory.

If the output shows null and the load data infile statement below does not work, you will need to set the configuration for your MySQL server.

The following example loads the data in tweets.csv in C:\ProgramData\MySQL\MySQL Server 8.0\Uploads into the tweet relation, ignoring the last column, which is the posting_user. The IGNORE 1 lines is to ignore the header line. Use this statement if your input file has header lines.

Set the default database to test first before executing the statement.

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/tweets.csv' INTO
TABLE tweet
FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '"' LINES
TERMINATED BY '\n'
IGNORE 1 LINES
(tid,textbody,retweet_count,retweeted,posted,@col6);
```

In the above example, the line after "IGNORE 1 LINES" indicates how the data for each line in the input file tweets.csv are entered into the relation "tweet". For each row, the first column goes into tweet.tid; the second

column goes into tweet.textbody; the third column goes into tweet.retweet_count, the forth column goes into tweet.retweeted, the fifth column goes into tweet.posted, and the last column goes into a user-defined variable called col6. This user-defined variable is how we ignore some columns or to process the value that we import before it is entered into the database. We know that col6 is a user-defined variable because of the @ in front of the name.

If the data type you use for your table is not large enough to store the data for each column, the data won't get loaded and an error is indicated.

The following example shows transformation of the data before they are entered into newtweet.

Note that @col6, the posting_user, is not entered into the newtweet. Also, the posting_time is transformed to day, month, and year through the functions day(), month(), and year() and inserted into the respective attributes, day_posted, month_posted, and year_posted.

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/tweets.csv'
INTO TABLE newtweet
FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(tid,textbody,retweet_count,retweeted,@col5,@col6)
set day_posted= day(str_to_date(@col5, '%Y-%m-%d %H:%i:%s')),
month_posted= month(str_to_date(@col5, '%Y-%m-%d %H:%i:%s')),
year_posted= year(str_to_date(@col5, '%Y-%m-%d %H:%i:%s'));
```

If you do not want to use the load data infile to transform the input directly, you can load the data into the tweet table and do the following to do the data conversion. Either way is acceptable for the project.

```
INSERT into test.newtweet
SELECT tid, textbody, retweet_count, retweeted, day(posted), month(posted), year(posted)
FROM test.tweet;
```

2. The following statement reports the size of the database buffer pool in gigabytes. The size of the database buffer pool impacts query execution time and is kept in the system variable innodb_buffer_pool_size. System variables in MySQL are prefixed with @@.

```
SELECT @@innodb_buffer_pool_size/1024/1024/1024
```

For database application development:

Suppose we want to use the user account pak to access the database from any machines. We create the user account "pak" with the modifier "@%" to tell the DBMS to accept this user's database connections also from any other machines. Observe the single quotes around the username and after the @ symbol. Then, we grant this user only necessary privileges to access the intended database or specific tables of the intended database.

3. The command to create a database user "pak@%" with the standard password "cs363pak" is below.

```
CREATE USER 'pak'@'%' IDENTIFIED WITH mysql_native_password BY 'cs363pak'
```

4. The command to grant the user "pak@%" privileges to select rows from all the tables of the database learnsql is below. Replace learnsql with your group database name. The use of learnsql.* means all the tables of the learnsql database.

```
GRANT SELECT ON learnsql.* TO 'pak'@'%';
```

5. The command to grant "pak@%" privileges to select, create, drop, execute, or display triggers on all the tables in the learnsql database is below.

```
GRANT TRIGGER ON learnsql.* TO 'pak'@'%';
```

6. The command to grant the user “pak@%” to do select, insert, update, delete rows from tables in the learnsql database and perform DDL such as create tables, alter tables, references, and create indexes as well as create temporary tables and lock tables is below.

```
GRANT ALL PRIVILEGES ON learnsql.* TO 'pak'@'%';
```