MPCS 53001: Assignment 6

Project is due by 4:59pm on Thursday, November 13, 2014

Steps 5 of Your TBP App

The fifth part of your TBP project is to generate some more data, populate your MySQL database with it and run several join and aggregation queries on this larger database.

Develop a larger amount of data for your database and load it into your relations using the SQL `LOAD` or `INSERT` commands. The data you generate and load should be on the order of:

- At least two relations with thousands of tuples (we recommend you choose Users and Tweets)
- At least one additional relation with hundreds of tuples

To create more data for your relations, you can write a program in any programming language you like. If you use `LOAD` the program should create large files consisting of records in an **ASCII text-only** format acceptable to the MySQL bulk loader. Then you should load the data into your TBP relations. If you are using real data, your program will need to transform the data into files of records conforming to your TBP schema. Otherwise you will write a program to fabricate data: your program will generate either random or nonrandom (e.g., sequential) records conforming to your schema. Note that it is acceptable for your data values to be meaningless (i.e. random sequence of characters instead of actual names).

If the semantics of your schema includes relations that are expected to be relatively small (e.g., states in the US), it is fine to use some small relations, but please ensure that you have relations of the sizes prescribed above as well. When writing a program to fabricate data, there are two important points to keep in mind:

- Be sure not to generate duplicate values for your key attributes.
- Your TBP certainly includes relations that are expected to join with each other. For example, if the key of the Users relation is username, then it is expected to join with the username_leader and username_follower attributes of the Follows relation. In generating data, be sure to generate values that actually do join - otherwise all of your interesting queries will have empty results! One way to guarantee joinability is to generate the values in one relation (corresponding to an entity set in your E/R diagram), then use the generated values to select joining values for the other relation (corresponding to a relationship in your E/R diagram). For example, you could generate usernames first (either sequentially or randomly), then use these usernames to fill in the username_leader and username_follower values in the Follows relation.

You will need to submit your new `create_db_large.sql` and `populate_db_large.sql` scripts along with **all data files** referenced in your `populate_db_large.sql` script. **Create a directory *data* inside your hw6 directory and move all of your data files there. Also, reference these files in your populate_db_large.sql script by relative unix paths.** For example, if you have a data file `tweets.txt`, there will be a line inside `populate_db_large.sql` that looks like:

```
LOAD DATA LOCAL INFILE "data/tweets.txt" ...
```

To demonstrate the size of your new relations, please, generate a script file called `count.sql` that finds the number of tuples in every one of your relations.

Extra credit: Develop a single query that returns the name of every table along with its size. For example, if you had only three tables: Users, Tweets, and Follows and their sizes were 10000, 50000, and 100000, the result of your single query should look like this:

```
+------------+------------+
| table_name | table_size |
+------------+------------+
| Users      |      10000 |
+------------+------------+
| Tweets     |      50000 |
+------------+------------+
| Follows    |     100000 |
+------------+------------+
```

Develop and test at least ten SQL queries using the JOIN syntax discussed in class. At least five of your queries should involve aggregation. To receive full credit, you should use all JOIN variations discussed in class (ON, USING, NATURAL, OUTER) and aggregation with and without GROUP BY clause. Please, generate a script file called `query6.sql` with all of your join and aggregation queries. **Your queries must not produce empty results and also must complete in a reasonable amount of time (a few minutes).** If any of your queries takes a long time to complete, consider modifying it by adding more conditions that limit the size of the intermediate results or final result. For example, if your query finds all pairs of users that follow at least one user in common:

```
SELECT DISTINCT F_1.username_follower AS user1,
F_2.username_follower AS user2
FROM Follows F_1 JOIN Follows F_2
USING (username_leader)
WHERE F_1.username_follower != F_2.username_follower;
```

and it takes 30 minutes or more to produces millions of results, consider adding some conditions that limit the users from F_1 and F_2. For example, you can only consider usernames starting with the letters "a" and "b":

```
SELECT DISTINCT F_1.username_follower AS user1,
F_2.username_follower AS user2
FROM Follows F_1 INNER JOIN Follows F_2
ON (F_1.username_leader = F_2.username_leader
AND F_1.username_follower LIKE 'a%'
AND F_2.username_follower LIKE 'b%')
WHERE F_1.username_follower != F_2.username_follower;
```

Note that you **need to submit the scripts that populate your large database as well as the corresponding data files (if using LOAD)**. For this assignment, there should be the four scripts: `create_db_large.sql`, `populate_db_large.sql`, `count.sql` and `query6.sql`, and all the necessary large data files.

The instructions on how to submit your project can be found here. Please make sure that all of the relevant files and no others are in your hw6 directory.

There is no Gradiance Problem Set this week