CISC637 Fall 2016 Course Project

Due Friday Dec 16 at 11:55pm on Sakai. **Submissions will not be accepted *at all* after 8:00am on Saturday Dec 17.** Submit on time, and double-check your submission to make sure it is complete.

For this project, you will design a database for stackoverflow.com, a large question-and-answer website for programming. Real data from the site will be provided to populate your database. You may work in groups of 1–3.

1 Assignment

Complete the following steps:

1. Draw an E-R diagram to capture the following requirements:
   (a) The site has users, each of whom has a non-unique display name, an account creation date, and the ability to fill in their location if they desire.
   (b) Users author posts that appear on the site. All posts have a creation date, and posts can have a title and a body (though either or both fields may be empty). A post is authored by exactly one user.
   (c) There are two types of posts: questions and answers. A question post can have multiple answer posts, but each answer post belongs to exactly one question post. In addition, at most one of the the answer posts belonging to a question can be chosen as the "accepted answer" by the user that posted the question.
   (d) Users can tag posts using a pre-defined set of tags. Only question posts can be tagged, and only by the user that posted the question.
   (e) Users can also comment on posts. Comments have a creation date and body text. A comment belongs to exactly one post (which can be a question or a reply). Any post can have zero or more comments.
   (f) Users can mark a question as one of their "favorites" (we call this "favoriting"). Users can favorite many questions, and a question can be favorited by many users, but a user may only favorite a particular question once.
   (g) Users can upvote and downvote both questions and answers. Users can vote on many posts, and a post may be voted on by many users, but a user may only vote on any post once. They may change their vote later, though.
2. Translate your E-R diagram to relations, and when you are happy with the relations you have, create them as tables in MySQL.
3. Download the project data from `http://ir.cis.udel.edu/~carteret/data.zip` and import it into your MySQL database. Please see below for details about the project data and tips for importing it into MySQL.
4. Write queries to handle the following scenarios:
   (a) List open questions (questions with no accepted answer). The query should retrieve question titles, tags, the post date, the name of the user that posted, the total number of votes, and the total number of answers. See `stackoverflow.com/questions` for an idea of the information you need. Write three variations on this query:
      i. List open questions tagged with a particular tag (for example "java"). You can assume the tag is a constant value.

ii. List open questions in reverse order of when they were posted.

iii. List open questions in decreasing order of total number of votes received.

(b) Display a question with its replies. The query should retrieve the question title and body, the post date, the name of the user that posted, the total number of votes and favorites, all comments on the question, all answers with their vote totals, and all comments on all answers. See `http://stackoverflow.com/questions/11227809` for an example. You can assume the post ID is a constant value.

For each scenario, write just one SQL query that retrieves and summarizes all necessary information (this may not be how it works in reality, but it is how I would like you to write them).

5. Try to optimize your queries to run faster. Some possible ways to improve query time include:

   (a) Build indexes on fields that are not indexed by default (remember that MySQL automatically builds indexes on all primary and foreign keys).

   (b) Try using index hints (`FORCE INDEX` and `IGNORE INDEX`) to change the way MySQL processes joins, or use `STRAIGHT_JOIN` to force it to join tables in a certain order.

   (c) Consider adding fields to help retrieve some data faster. You may add any fields you like.

   (d) Consider refactoring data into more tables that are smaller (or fewer tables that are larger, depending on the query).

2 Project Data

The data is distributed as a zip file containing seven tab-separated text files. Download the data from `http://ir.cis.udel.edu/~carteret/data.zip` (it is a 117M file, too large for Sakai).

- `users.txt` contains data about 492,782 users; fields are ID, display name, creation date, and location.
- `posts.txt` contains data about 232,083 posts; fields are ID, post type (1=question, 2=answer), user ID, post date, title, body, parent post ID (if type is 2), and accepted answer post ID (if type is 1).
- `comments.txt` contains data about 352,613 comments; fields are comment ID, post ID, user ID, post date, and comment text.
- `tags.txt` contains data about 38,205 tags; fields are ID and tag name.
- `posttags.txt` contains data about 253,850 tags on posts; fields are post ID and tag ID.
- `votes.txt` contains data about 480,351 user votes; fields are post ID, user ID, vote type (2=upvote, 3=downvote), and vote date.
- `favs.txt` contains data about 45,556 favorites; fields are post ID, user ID, and favorite date.

These are large files (with the exception of `tags.txt`). Opening them in a text editor or word processing program is NOT recommended, as they will probably grind your computer to a halt. If you want to look at them, try the command-line `more` or `less` utilities (`more` should work on Windows PCs, both `more` and `less` work on Linux and Macs).

2.1 Importing Data Into MySQL

First, create the table that you want to import data into. Be sure that the order the fields are defined in matches the order of the fields in the text file you are importing. Then, at the MySQL prompt:

```
mysql> LOAD DATA LOCAL INFILE '/full/path/to/posts.txt' INTO TABLE post
       FIELDS TERMINATED BY '\t';
```

When done, you will see output like this:

```
Query OK, 232083 rows affected, 14 warnings (13.29 sec)
Records: 232083  Deleted: 0  Skipped: 0  Warnings: 14
```

Use `SHOW WARNINGS;` to see warnings, though you can ignore most of them (you only need to pay attention to warnings if you have modified the original data).

The relations you developed for step 2 of the assignment may be different from these files. In that case, you have two options for importing the provided data into your own tables:

1. Write scripts to change the provided text files into files that match the tables you defined, by redistributing rows and/or fields.
2. Or, import all the data as provided into temporary tables, then use SQL commands to move data into the tables you defined.

If you import the data directly, you will probably want to turn off foreign key checks for the `posts.txt` file. You can do that by typing `SET foreign_key_checks=0;` at the MySQL prompt.

Since the files contain a lot of data, importing may take a few minutes. Be sure to give yourself enough time to import the data and figure out the queries.

3 What to Turn In

One student from the group should turn in the following on Sakai:

1. A list of group members (please just type them into the text box on the submission page).
2. Your E-R diagram from step 1 of the assignment.
3. Your `CREATE TABLE` statements from step 2 of the assignment (but please do not turn in any data!). You can use `mysqldump` to get the table schema. Be sure to use the `--no-data` flag to suppress data output. `mysqldump -u [username] -p --no-data [dbname] >project.sql`
4. The SQL queries you wrote for step 4 of the assignment.
5. A written report of 2–4 pages. The report *must* consist of the following sections:
   (a) **E-R Model** In this section, explain your thinking in putting together your E-R diagram. If there are any requirements that you *could not* (or decided not to) capture in the E-R model, explain why in this section.

(b) **Relational Model** In this section, explain how you developed the relations from the E-R diagram. For instance, if you decided to merge entity sets and relationship sets into a single relation, explain why. If there was anything in your E-R diagram that you were not able to translate into relational schema, explain it here.

(c) **Data Import** In this section, explain decisions you made while importing the data. In particular, if you split up the data, describe how you split it and why.

(d) **Query Optimization** In this section, describe steps you took to try to optimize query processing time. Include any indexes you built, and any uses of `FORCE INDEX`, `IGNORE INDEX`, or `STRAIGHT_JOIN` that improved processing time. If you decided to add new fields or re-factor tables to make queries faster, discuss that in this section as well, describing what you did and why. It is appropriate in this section to include tables showing differences in un-optimized and optimized query time.

The report *must* be written in paragraphs and complete sentences. It should not contain any SQL DDL statements, or screenshots of MySQL, or any other overly-technical details. It can contain figures and tables that help illustrate improvements in query processing time. It can contain SQL queries as demonstrations of optimization techniques, but no more than 2 full queries.