University of Victoria
CSC 370: Database Systems

Assignment 8

Due on: Friday, November 22 at 11:59pm (26 marks)

Introduction

The purpose of this assignment is to gain experience interfacing with a relational database (PostgreSQL) using a procedural programming language (Python).

You will be working with the database and case study from assignments 5 and 6 implementing parts of the written description of the scenario that you did not implement using the DBMS.

Requirements from the Case Study

The case study had many requirements within it that you would have recognized were not possible to implement within our database given our current understanding and skill level working with database systems. However, all of these requirements can be implemented within our application code when interfacing with the DBMS and our instance of the database.

These were:

- "Trolls (troublesome users) can also simply be banned, meaning that they are no longer able to make new posts."
- "Administrator and moderators should also be able to 'pin' topics so that they always show up at the top of the topic listings for any particular category."
- "Only administrators should be allowed to define new tags."
- "If a user mistakenly rates a post, they can re-rate it, or they can clear it, resulting in a rating of 0."
- "When a topic is watched, then any time a new post is added to that topic an email should be sent to the user notifying them of the new post."
- "If the user views the topic that they have visited previously and there are new posts, those posts should be highlighted in green."
- "Additionally, categories cannot be deleted if they contain any topics at all — all of the topics in that category should be migrated to another category before deletion."

Your goal for this assignment is to implement some of these requirements within application code.

Questions

Use the provided `assignment8_start.py` file and implement the functions within the file, as indicated by each question. Use the provided `.sql` file to create a blank database with the given structure. Feel free to define your own helper functions when implementing your solution, but do not use any additional libraries other than those that are already imported. You will need to install `psycopg2-binary` via the `pip` command (there are many online tutorials describing how to use `pip`):

```
pip install psycopg2-binary
```

Functions to insert users and categories are provided for reference, and to give you the ability to test the functions you implement (as they rely on categories and users existing within the database).

Warning: If any of your functions are vulnerable to SQL injection attacks, you will lose all points associated with the question. As an example of what not to do, the bottom of the provided file contains a function that is vulnerable to SQL injection attacks. You can use a similar approach to test your own functions.

1. (6 points) Implement the `insert_post()` function.

   This function should `INSERT` a new Post into a Thread, based on the function's parameters. Check the User's permissions to see if they are banned. If they are banned, then `print` an error message and do not perform any inserts.

   Your function should return the new PostID if everything was successful.

   (You won't be able to properly test this function unless you manually add a new Thread or complete the next question.)

2. (6 points) Implement the `insert_thread()` function.

   This function should `INSERT` a new Thread into the database. A Post should also be inserted alongside the Thread. Both should be done based on the function's parameters. You should make use of the `insert_post()` function you implemented in the previous question.

   Your function should check the User's permissions. Users that are banned should be unable to create threads. If the thread is going to be pinned, then check to see if the user creating the thread is a moderator or administrator. Perform these checks in your function. If any of these checks fails, then `print` an error message and do not perform any inserts.

   The function should return the new ThreadID if everything is successful.

3. (6 points) Implement the `rate_post()` function.

   This function should perform either an `UPDATE` or a `INSERT` to the Rating table. If a rating already exists (the PostID + RatedByUserID combination), then an `UPDATE` should be done, otherwise, an `INSERT` should be done.

   Note: The surrogate primary key, RatingID, exists because RatedByUserID is nullable. This was done so that the ratings of Users that get deleted can continue to be stored. (This wasn't a requirement in the written description of the scenario.)

4. (8 points) Implement the `delete_category()` function.

   The written description describes how Categories cannot be deleted if they contain threads, and instead those threads should be migrated to a new category before deletion. This function should implement this logic. For a given CategoryID, `UPDATE` that CategoryID with a new CategoryID for every Thread, and then `DELETE` the original CategoryID.

Submit your entire Python file to Crowdmark, renaming it to include your VNumber: "assignment8_<VNumber>.py"