C2102-2310 Group Project

Part 3: Database Programming

Objective

Overview: In Part 2, you have created a database in terms of **`CREATE TABLE`** statements. Now in Part 3, the objective is to achieve two main goals towards creating a practical application using your database. A first common task is to implement core functionalities using stored functions and procedures. These functionalities refer to common fundamental operations we need to perform on the data that

- Require multiple database operations *or*
- Require procedural logic (due to the limitations of basic SQL)

In practice, it is a common design decision to move such functionalities from the application into the database to improve performance, security, and maintenance.

The second common task is to address the limitations you have encountered in Part 2. You will recall that some of the constraints listed in the application description were impossible to capture using constructs such as **`PRIMARY`**, **`FOREIGN KEY`**, **`NOT NULL`**, **`CHECK`**, etc. as part of the **`CREATE TABLE`** statements. On an abstract level, such constraints typically include:

- Constraints on the *change* of the data
- Constraints that affect *multiple tables*

Based on your work in Part 1 and 2, you probably already have some good ideas what meaningful functionalities and constraints check Squiz can benefit from. Therefore, in Part 3, your task is therefore to implement a selected functionality and triggers.

Implementation & Testing: For Part 3 of the project you will get a reference database schema. Again, this only to ensure that all teams use the same schema. All your implementations should be based on this given schema. We also provide you with toy data to get started. However, you can and should test your solutions by creating additional example test cases in terms of creating example data for the involved tables.

Task 1 – Functionality: Generate Practice Quiz (9 Marks)

Squiz allows educators to create quizzes using the public questions from others. This means that creating a quiz does not necessarily require creating new questions. We want to support educators even further by offering the functionality to create practice quizzes. Your task is therefore to implement a stored procedure **`create_practice_quiz()`** that creates a quiz by selecting a list of public questions. The minimum input arguments for this procedure are:

- **`quiz_name`**: the name of the quiz
- **`creator_id`**: the id of the educator who created the quiz (i.e., called this procedure)
- **`num_questions`**: the total number of question that quiz should contain

All other considerations about how to create the quiz are up to you. This may include that your implementation of procedure **`create_practice_quiz()`** might feature additional input arguments. To keep it simple, you can assume:

- practice quizzes are not marked; all questions have 0 points
- practice quizzes are immediately published
- practice quizzes are available the moment they are created without a end date/time
- practice quizzes have no time limit
- practice quizzes are all public

Comments & hints: The simplest way to implement this procedure is to arbitrarily pick **`num_questions`** questions. However, this will only give you 6 marks. Think about what data you have available and what choices you can make to create more meaningful practice quizzes. There is no single correct solution for this task; you can and should be creative.

Task 2 – Constraint Check: Total Points of a Quiz (9 Marks)

Table **`quizzes`** has the attribute **`total_points`** which is required to always correctly reflect the total number of points achievable for a quiz. This number derives as the sum of points for all questions currently in the quiz. As such, adding or deleting a question to or from a quiz, as well as changing the number of points of question in a quiz, will generally violate this constraint.

Write a trigger or set of triggers that checks the application constraint that **`total_points`** always correctly reflects the sum of points for each question contained in this corresponding quiz! For this task, consider all the events that could potentially violate this constraint.

Submission

Deadline & Deliverable

The deadline for Part 3 is **Nov 9, 23.59**. This is a hard deadline, and late submissions will not be accepted and graded.

Each team is to upload a zip file named `teamNNN.zip` where `NNN` is the three digit team number according to your project group number. You should add leading zeroes to your team number (e.g., `team005.zip`). Submit your zip file on Canvas assignment named "Project - Part 3: Database Programming". Only one zip file is to be submitted per group. If there are multiple submissions for a group, the submission with the lower mark will be chosen.

The zip file should contain 3 files

- A text file named `code-teamNNN.sql` containing the implementations of your stored procedure(s) and trigger(s) as required by the task description. Make sure that this file can be imported into PostgreSQL without throwing errors.
- A text file named `cases-teamNNN.sql` containing 2-3 of your own test cases that illustrate the inner workings and the correctness of your solutions. Make sure that this file can be imported into PostgreSQL without throwing errors. This includes that any additional descriptions should be added only as comments.
- A pdf file named `teamNNN.pdf` with a font size of at least 12 point for normal text and consists of the following:
    - Project team number & names of all team members (on the first page).
    - A brief discussion of the most challenging parts in your implementations.
    - A brief discussion and justification for your design of procedure create_practice_quiz() – for example, briefly describe your motivation idea(s) behind your implementation of this procedure.

Grading

Part 3 of the project is worth 18 marks in total; the breakdown of the marks w.r.t. the different subtasks is given by the individual marks for each subtask – that is 9+9 marks.