Complex Systems Theory and Practice

Lab 2

Piotr Syga

October 31, 2021

1. Create `Music` database. Create two users[1] – *MusicAdmin* and *U*, where *U* is your name concatenated with 3 last digits of your index number. Set the passwords for both users (remember about securing the passwords). Give *MusicAdmin* full grants to the `Music` database and no privileges outside of it. Give *U* sufficient privileges to select from the whole `chinook` and `Music` databases, to modify the contents of the tables in `Music` database, but no privileges to create or modify tables nor views.

2. In the `Music` database create tables:

   - `bands` (id:int, name:varchar(90), noAlbums: int),
   - `albums`(title: varchar(90), genre: varchar(30), band:int),
   - `songs`(id:int, title: varchar(90), length: int, album: varchar(90), band: int).

   Note that

   - Underlined attributes or sets of attributes are primary keys.
   - IDs are autoincremented artificial keys.
   - The length of a song is given in seconds and is non-negative (establish a proper restrictions and verify if they work).
   - `noAlbums` is `NULL` by default, it will be filled out in later tasks.
   - In `songs` table, `album` and `band` columns should be set to foreign keys referencing relevant tables.

3. Import the data from `chinook` database to `Music`. Do not import data concerning TV series, movies or songs that include 'v' in their title.

4. Create a procedure that counts the number of albums of each of the band and fills out the table. Try to utilize cursors.

[1] @'localhost' or @'%'

5. Create a trigger that updates `noAlbums` column every time that the contents of `albums` table change.

6. Create a procedure with a single input parameter – the number of albums to be generated k – and adds k random albums to the table, and then updates the relevant information in the other tables. Each album should have between 4 and 25 songs, may belong to a new or already present in the database band, yet all generated in a single batch albums cannot be recorded by a single band (if $k > 1$). Utilize `WHILE` or `LOOP` constructions.

7. Create a function or a procedure that for a given band name returns the title of its shortest album. Make sure that it is secure against errors or SQL injections.

8. Create a view that includes all the information about songs longer than 3 minutes and their authors.

9. Create a `PREPARE statement` that for a given genre returns the number of albums in it, recorded by each of the bands. The genre name should be given during `EXECUTE`. Check if the input is safe.

10. Using `PREPARE STATEMENT`, create a function or a procedure that allows selecting all songs with length in a given relation `rel` with aggregate function `agg`. The relation `rel` should allow the values '<', '<=', '>', '>=', '='. The aggregate functions `agg` should allow MIN, MAX, AVG. Additionally, allow that entering `agg`=*std* results in selecting all the songs, that have length at most two standard deviations from the mean length of a song.

11. Create a procedure that selects a random[2] playlist consisting of songs that last not longer than a given input parameter `thresh`. Use transactions to make sure that the restriction is not violated.

12. Create table `playlist` and save the results of the previous task. The information stored should include the ID of the playlist, the threshold given and all the songs in the playlist. If needed, you may use additional columns or tables.

13. Create a backup of your database. Delete the database from your DBE and then restore it using the backup made. Write and present a report of the backuping and restoring both in a selected IDE and command line.

14. Try to do each of the tasks with both new users as well as a default *root* user – note which ones can be done by whom.

[2]If there are multiple solutions, each procedure call with the same parameter should result in a different solution.

15. Download WebGoat app[3] and run it locally. Pass through *Injection Flaws*:

    - *SQL Injection (introduction)* – optional
    - *SQL Injection (advanced)*
    - *SQL Injection (mitigation)*

    Select one of the topics (advanced or mitigation) and create a short report describing the performed tasks and the conclusions drawn from them. Send the report (in pdf format) to the lecturer at least 24h before the deadline.

[3]Note that the later versions may require a proper Java version, e.g. 11