CS–A1150 Databases, 2020

Exercise Round 5 (11.5.–15.5.2020), deadline for submission 16.5.2020 in the morning.

Submit your solutions to all these problems as one zip-file containing the pdf-file with answers to Problems 1–3, .py-files containing the answer to Problems 4 and 5, and either text of pdf file containing an example execution of the Python program written in Problem 5. If you do not answer to Problems 4 and 5, you can also submit your solution as a single pdf-file. You are allowed to solve the problems in the groups of 1–3 students (each group submits only one common solution). However, all members of the group must take part in solving all submitted solutions. You are not allowed to divide the various problems among the group members. Although the problems are given in English, you may write your solutions in Finnish, English or Swedish.

1. (8 p.) Suppose that the relation `BelongsTo` discussed on the lecture slides of the lecture Apr 20th (the slides and English translation of the example will be published in MyCourses in Apr 20th) required 300 pages rather than 100, each order included 4 products on the average and each product belonged to 25 orders on the average, but all other assumptions of that example continued to hold. Give formulas in terms p_1 and p_2 to measure the cost of queries Q_1 and Q_2 and insertion I, under the four combinations of index / no index discussed there. Explain briefly where the numbers in your formulas come from.

2. (6 p.) Consider a bank database which contains Table `Account` whose schema is

   ```
   Accounts(accountNo, type, balance)
   ```

   Three computer programs read and possibly write the balance of the same bank account partly concurrently. Those programs use a local variables `var1`, `var2`, and `var3` respectively. Each program i runs a separate transaction T_i, but it is possible that the database management system does not satisfy the ACID properties of the transactions. We use the following shortcuts for some statements, and `varX` means either `var1`, `var2`, or `var3` depending on the program where it is used.

   `varX = read(balance)` means that the following SQL statement

   ```
   SELECT balance
   FROM Accounts
   WHERE accountNo = 'A123';
   ```

   is executed and the result of the query is assigned to variable `varX`.

   `balance = write(varX)` means that the following SQL statement

   ```
   UPDATE Accounts
   SET balance = varX
   WHERE accountNo = 'A123';
   ```

   is executed such that `varX` is replaced by its current value in the program.

   Statement `output(varX)` means that the program outputs the value of its local variable to the user interface.

Statements like `varX = varX + 100` are assignments to the local variable in the program. Each program has its own local variable and the programs do not have direct accees to the local variables of the other programs. However, all programs use the same database.

In the following schedules, the time runs from up to down i.e. the upper statement is executed before the lower statement. Which of the following schedules are serializable and which are not? If the schedule is serializable, give the corresponding serial schedule of the same transactions (i.e. the order in which the same transactions are executed one at a time such that the execution leads to the same result). If it is not, explain briefly what causes the problem with the serializability

a) The first schedule:

```
     T1                          T2                          T3
var1 = read(balance)
                            var2 = read(balance)
                            var2 = var2 * 1.05
var1 = var1 + 100
balance = write(var1)
                            balance = write(var2)
                                                    var3 = read(balance)
                                                    output(var3)
```

b) The second schedule:

```
     T1                          T2                          T3
var1 = read(balance)
var1 = var1 + 100
balance = write(var1)
                            var2 = read(balance)
                            var2 = var2 * 1.05
                                                    var3 = read(balance)
                                                    output(var3)
                            balance = write(var2)
```

3. (8 p.) Consider a database schema which consists of four relations, whose schemas are:

```
Students(ID, name, program, year)
Courses(code, name, credits)
Grades(studentID, courseCode, date, grade)
```

See Exercise Round 1 for detailed description of the relations and their attributes. Write the triggers described in parts (a), (b), and (c). After that, test the triggers in part (d). Use SQLite syntax, not SQL standard syntax such that you can test the triggers using SQLiteStudio. (*Hint:* Instead of writing **AFTER UPDATE**, it is also possible to write **AFTER INSERT** or **AFTER DELETE** when defining triggers.)

a) When inserting a row into table `Courses`, check that the value of attribute `credits` is not greater than 10. If it is, set the value of `credits` of the inserted row to be 10.

b) When updating the value of attribute `grade` in Table `Grades`, check that the new grade is not smaller than the old grade the student has in the same course. If the new grade is smaller, replace it with the old grade.

c) When updating the value of attribute `credits` of the course, check that the new number of the credits is at most 2.0 greater than the old number of the credits of the same course. If it is not, set the updated credits to be the old credits + 2.0.

d) Use SQLiteStudio and create a new database with the help of the file `create-database-round5.sql` given in MyCourses (this database is much simpler than the example database in Exercise Round 2). Write the definitions of the triggers from parts (a), (b) and (c). Write two INSERT statements and four UPDATE statements to test your triggers. List the contens of the tables `Courses`, and `Grades` both before and after running all those statements (intermediate results are not required).

4. (8 p.) Consider the music album database schema from Round 4, Problem 17. Write a Python program which executes the following operations using sqlite3 library:

   - Creates a new database with name musicalbums.db and connects to it.
   - Creates a `Cursor` object to use the database.
   - Creates tables `Albums`, `Companies`, `Artists`, and `Tracks` according to the definitions given in Problem 17, Round 4.
   - Inserts one row into Table `Artists`. You do not have to ask any input from the user, but you can just hard code the artist information in your program.
   - Executes one SQL query which finds the year of the birth of the artist inserted above. The program must output the result of the query to the user. The program does not have to ask any input from the user.

   Your program does not have to contain any error handling. Include the .py-file containing your Python program with your submission.

5. (10 p.) Continue your work with the Python program you wrote in Problem 4. Now, it is time to make your program to use the input read from the user (or from a text file) in SQL statements. Add to your program the following operations using sqlite3 library:

   - Reads input from either the user or from a text file and inserts at least 3 rows to Table `Companies` and 6 rows to Table `Albums` based on that input. The input must not include the actual INSERT commands to be executed, but the data which should be inserted to the tables.
   - Asks the user to input a name of a record company and outputs all albums in the database published by this company (name, year, and genre of the album).

   Your program does not have to contain any error handling except that it has to prevent SQL injections by using parameter substitution. You may assume that the user gives no NULL values to the attributes. If your program reads the input from text file when inserting rows to the tables, include this text file in the zip file you submit. In any case, include the .py-file contining your Python program and the text or pdf file containing an example execution of your program.