DATABASE SYSTEMS

Coursework 1

Dr Paolo Guagliardo

Due date: Mon 29 October 2018 at 16:00

Before you begin, please read carefully the policy on Late coursework and Academic misconduct and related links, in particular the rules for the publication of solutions to coursework (which is not allowed unless you have explicitly obtained my permission in writing).

Database schema. The schema we use in this assignment models a simplified university scenario consisting of the tables below.

COURSES have a unique code (string), a name (string), and a credits value (non-negative integer).

DEGREES have a unique code (string), a name (string), and a type attribute (string) that is either "`UG`" (for *undergraduate*) or "`PG`" (for *postgraduate*).

STUDENTS have a unique UUN identifier (string), a name (string), and a reference to the code of the degree they are enrolled in.

EXAMS have a reference to the code of a course, a reference to a student's UUN, a date, and a grade (integer between 0 and 100). Each row is uniquely identified by the course code and the student's UUN.

PROGRAMMES have a reference to the code of a degree, and a reference to the code of a course. The values of these two references uniquely identify each row. This table indicates the mandatory courses that must be taken in order to obtain each degree.

The detailed SQL schema can be found in the following AFS folder:

`/afs/inf.ed.ac.uk/group/teaching/dbs/2018/cw1/`

as a file called "`schema.sql`".

Test instance. To work on the assignment, you can use the test database that is located in the same AFS folder as the schema, in a file called "`testdb.sql`". Along with the data to populate the test database, this file also includes the schema (even though the constraints are scattered around, for reasons we may discuss later on). You can execute it on PostgreSQL using pgAdmin or `psql`.

Please note that you cannot make assumptions based on the test data: your queries will be executed on different instances (of the same schema) on which such assumptions may not hold. The only constraints that are guaranteed to hold on every instance are those declared in the schema.

Assignment. Write the following queries in SQL.

(01) **Students who have not taken any exams**

Return the UUN of each student that satisfies this requirement, without repetitions. The output table will have a single column, which consists of UUNs. The number of rows depends on the instance.

(02) **Total number of postgraduate students**

The output table will have a single column, which consists of non-negative integers (and no nulls). The number of rows is always 1, independently of the instance. If there are no postgraduate students, the answer will be 0.

(03) **Students whose average grade is greater than or equal to** 75

For each student that satisfies this requirement, return their UUN, and their average grade formatted as a decimal number rounded to two decimal places (do *not* use the `ROUND` function). Students without exams do not appear in the output.

The output table will have two columns: the first consists of UUNs, the second consists of non-negative decimal numbers with two decimal places. The number of rows depends on the instance, and there are no repetitions.

(04) **Students who failed more than 30% of their exams**

Return the UUN of each student that satisfies this requirement, without repetitions. An exam is failed when the grade is below 40.

The output table will have a single column, which consists of UUNs. The number of rows depends on the instance.

(05) **Total number of credits in the programme of each degree**

For each degree, calculate the total number of credits of the courses listed in its programme. Return the code of the degree and the corresponding total. Degrees with no mandatory courses will be in the output with a total of 0.

The output table will have two columns: the first one consists of degree codes, the second consists of non-negative integers (and no nulls). The number of rows depends on the instance, but it is always the same as the number of rows in the Degrees table.

(06) **Number of A, B, C and D exam grades of each student**

Return the student's UUN, followed by columns A, B, C, D (in this order) with the total number of exam grades in each of the following categories:

- A is a grade above 80 (inclusive),
- B is a grade between 60 and 79,
- C is a grade between 40 and 59,
- D is a grade below 40.

Students without exams do not appear in the output.

The number of rows depends of the instance, but it will always be the same as the number of distinct UUNs in the first column of the Exams table.

(07) **Courses that are part of an undergraduate and a postgraduate degree programme**

That is, courses that are in the programme of some undergraduate degree and also in the programme of some postgraduate degree. Return the code of each course that satisfies these requirements, without repetitions.

The output table will have one column, which consists of course codes. The number of rows depends on the instance.

(08) **Courses included in one and only one postgraduate degree programme**

That is, courses that are in the programme of some postgraduate degree and not in the programme of any other postgraduate degree. Return the code of each course that satisfies this requirement, without repetitions.

The output table will have one column, which consists of course codes. The number of rows depends on the instance.

(09) **Students who took more than one exam on the date of their most recent exam**

For each student, return their UUN and the date of their most recent exam, if on that date they have taken more than one exam.

The output table will have two columns: the first consists of UUNs, the second consists of dates. The number of rows depends on the instance, and there are no repetitions.

(10) **Students who have taken the exam for every course in their degree programme**

For each student that satisfies this requirement, return their UUN and name. Students in degrees without mandatory courses (listed in the PROGRAMMES table) satisfy the requirement and must appear in the output.

The output table will have two columns: the first one consists of UUNs, the second consists of student names. The number of rows depends on the instance, and there are no repetitions.

Submission instructions.

- Each query must be written in a text file named `<xx>.sql` where `<xx>` is the two-digit number in the list of queries above. For example, the first query will be written in file `01.sql` and the last one in file `10.sql`. **The character encoding of the file must be ASCII; if you use UNICODE your queries will fail.**
- Each file consists of a single SQL statement, terminated by semicolon (`;`). Submitted files that do not contain *exactly one semicolon* will be discarded when the submission is processed and consequently they will not be assessed (as if they were not submitted). **Please pay attention to this; even if it looks like a trivial detail, it is not.**
- If a file contains more than one statement, or a statement that is not a query (such as `CREATE VIEW` or `UPDATE`), it will fail. Do not use comments either, as they may cause a correct query to fail if they contain semicolons.
- Submission is via the `submit` command on DICE.[1] You can submit all your files at once as follows:

  ```
  submit dbs cw1 01.sql 02.sql 03.sql 04.sql 05.sql 06.sql 07.sql 08.sql 09.sql 10.sql
  ```

  You can also submit your files one by one, or in smaller groups, and in any order. For example:

  ```
  submit dbs cw1 03.sql
  submit dbs cw1 01.sql
  submit dbs cw1 04.sql 02.sql
  ```

[1] See how to submit a practical exercise.

Assessment.

- Your queries will be executed on 5 randomly generated database instances that comply with the given schema (including the constraints). Each query is worth 10 marks, allocated as follows:
  - If the query returns a *single wrong row* on *any* of the test instances, 0 marks are given; otherwise
  - 1 mark is given for each test instance on which the query returns *at least 50%* of the correct rows, but not all of them; and
  - 2 marks are given for each test instance on which the query returns *all and only* the correct rows.
- The queries will not be assessed for their performance, even though each query should not take more than a few seconds to run. Your writing style will not be assessed either: you could write a query on a single line and in NonSenSicAL-CAse. Obviously, this is not recommended for your own sake.
- The order in which the rows appear in the answer to your queries is irrelevant for this assignment (no ordering is enforced on the answers, so the DBMS will output rows in an arbitrary order). The names of the columns in the answers are also irrelevant. What is **important** is the number of columns, their types, and the order in which they appear in the output.
- All queries can and **must** be written using only the SQL constructs we have seen in class. If in doubt, please check with me.