COMP9311 24T2: Project 1

Deadline: 21:59:59 Sun 7 July

1 Aims

This project aims to give you practice in

- Reading and understanding a moderately large relational schema (MyMyUNSW).
- Implementing SQL queries and views to satisfy requests for information.
- Implementing PL/pgSQL functions to aid in satisfying requests for information.
- The goal is to build some useful data access operations on the MyMyUNSW database. The data may contain some data inconsistencies; however, they won't affect your answers to the project.

2 How to do this project:

- Read this specification carefully and completely.
- Familiarize yourself with the database **schema** (description, SQL schema, summary).
- Make a directory for this project, and put a copy of the **proj1.sql** template there.
- You **must** use the create statements in **proj1.sql** when defining your solutions.
- Look at the expected outputs in the expected_qX tables loaded as part of the **check.sql** file. (Please see 'Section 6. Test your solution' for details about **check.sql**.)
- Solve each of the problems in the 'tasks' section and put your completed solutions into **proj1.sql.**
- Check that your solution is correct by verifying against the example outputs and by using the check_qX() functions (following the 'AutoTest Checking' section).
- Test that your **proj1.sql** file will load *without error* into a database containing just the original MyMyUNSW data.
- Double-check that your **proj1.sql** file loads in a *single pass* into a database containing just the original MyMyUNSW data.
- Check your solution on the nw-syd-vxdb server before submission if your project is conducted on your own machine.
- For each question, you must output the result within 120 seconds on the nw-syd-vxdb server.
- **Hardcode is strictly forbidden. We will use different test data during marking.**

3 Setting Up the Project

To install the MyMyUNSW database under your nw-syd-vxdb server, simply run the following two commands:

```
$ createdb proj1
$ psql proj1 -f /home/cs9311/web/24T2/proj/proj1/mymyunsw.dump
```

If you've already set up PLpgSQL in your template1 database, you will get one error message as the database starts to load:

```
psql:mymyunsw.dump:NN: ERROR:  language "plpgsql" already exist.
```

You can ignore the above error message, but all other occurrences of ERROR during the load need to be investigated.

If everything proceeds correctly, the load output should look something like:

```
SET
SET
SET
SET
SET
psql:mymyunsw.dump:NN: ERROR:  language "plpgsql" already exists
... if PLpgSQL is not already defined,
... the above ERROR will be replaced by CREATE LANGUAGE
SET
SET
SET
CREATE TABLE
CREATE TABLE
... a whole bunch of these
CREATE TABLE
ALTER TABLE
ALTER TABLE
... a whole bunch of these
ALTER TABLE
```

Apart from possible messages relating to plpgsql, you should get no error messages.

The database loading should take less than 60 seconds on nw-syd-vxdb, assuming that nw-syd-vxdb is not under heavy load. (If you leave your project until the last minute, loading the database on nw-syd-vxdb will be considerably slower, thus delaying your work even more. The solution: at least load the database Right Now, even if you don't start using it for a while.) (Note that the mymyunsw.dump file is 50MB in size; copying it to your home directory or your '/srvr' directory is not a good idea).

If you have other large databases under your PostgreSQL server on nw-syd-vxdb or if you have large files under your '/srvr/YOU/' directory, it is possible that you will exhaust your nw-syd-vxdb disk quota. Regardless, it is certain that you will not be able to store two copies of the MyMyUNSW database under your nw-syd-vxdb server. The solution is to remove any existing databases before loading your MyMyUNSW database.

After you load the database successfully, you can now create a directory and copy the template solution to that directory. Now, you are ready to do the project by editing the template solution.

```
$ mkdir Project1Directory
... make a working directory for Project 1
$ cp  /home/cs9311/web/24T2/proj/proj1/proj1.sql  Project1Directory
```

The only error messages produced by these commands should be those noted above. If you omit any of the steps, then things will not work as planned.

4 Important Advice Before You Start

The database instance you are given is not a small one. The first thing you should do is get a feeling for what data is there in the database. This will help you understand the schema better and will make the tasks easier to understand. *Tip: study the schema of each table to see how tables are related and try write some queries to explore/ understand what each table is storing.* **Read these** before you start on the exercises:

- The marks reflect the relative difficulty/length of each question.
- Work on the project on the supplied **proj1.sql** template file.
- Make sure that your queries work on any instance of the MyMyUNSW schema; don't customize them to work just on this database; we may test them on a different database instance.
- Do not assume that any query will return just a single result; even if it phrased as "most" or "biggest", there may be two or more equally "big" instances in the database.
- When queries ask for people's names, use the People.name field; it's there precisely to produce displayable names.
- When queries ask for student ID, use the People.unswid field unless explicitly specified; the People.id field is an internal numeric key and of no interest to anyone outside the database.
- **Unless specifically mentioned in the exercise, the order of tuples in the result does not matter; it can always be adjusted using order by. In fact, our check.sql will order your results automatically for comparison.**
- The precise formatting of fields within a result tuple **does** matter, e.g., if you convert a number to a string using to_char it may no longer match a numeric field containing the same value, even though the two fields may look similar.
- We advise developing queries in stages; make sure that any sub-queries or sub-joins that you're using works correctly before using them in the query for the final view/function
- You may define as many additional views as you need, provided that (a) the definitions in proj1.sql are preserved, (b) you follow the requirements in each question on what you are allowed to define.
- If you meet with error saying something like "cannot change name of view column", you can drop the view you just created by using command "**drop view VIEWNAME cascade;**" then create your new view again.

Each question is presented with a brief description of what's required. If you want the full details of the expected output, look at the expected_qX tables supplied in the checking script (check.sql).

5 Tasks

To facilitate the semi-auto marking, please pack all your SQL solutions into view/function as defined in each problem (see details from the solution template we provided).

Question 1 (4 marks)

Define a SQL view `Q1(code)` that gives the subject codes (`subjects.code`) of all subjects that are related to 'Database' (`subjects.longname` contains `'Database'`) and are offered by the School of Computer Science and Engineering (`orgunits.longname = 'School of Computer Science and Engineering'`). The view should not contain duplicate subject codes.

Question 2 (4 marks)

Define a SQL view `Q2(id)` that gives the course IDs `(courses.id)` of all courses that have Lab classes (`class_types.name = 'Laboratory'`) in room MB-G4 (`rooms.longname = 'MB-G4'`). The view should not contain duplicate course IDs.

Question 3 (4 marks)

Define a SQL view `Q3(name)` that gives the names (`people.name`) of all students who have ever achieved a mark of 95 or above (`course_enrolments.mark >= 95`) in subject COMP3311 (`subjects.code = 'COMP3311'`). The view should not contain duplicate names.

Question 4 (4 marks)

Define a SQL view `Q4(code)` that gives the subject codes (`subjects.code`) of all the COMM subjects (`subjects.code` contains `'COMM'` as prefix) whose courses contain classes in a room with Student wheelchair access (`facilities.description = 'Student wheelchair access'`). The view should not contain duplicate subject codes.

Question 5 (5 marks)

Q5. Define a SQL view `Q5(unswid)` that gives the unswid (`people.unswid`) of all students who achieved a 'HD' grade (`course_enrolments.grade = 'HD'`) in all COMP9 courses (`subjects.code` has `'COMP9'` as prefix) they have enrolled in.

Question 6 (5 marks)

Define a SQL view `Q6(code, avg_mark)` that gives the subject codes `(subjects.code)` and the average mark (rounded to 2 decimal places) of all subjects.

- Only consider subjects for undergraduate students (`subjects.career = 'UG'`) with less than 6 UOC (`subjects.uoc < 6`) offered by the School of Civil and Environmental Engineering (`orgunits.longname = 'School of Civil and Environmental Engineering'`).
- When calculating the average mark, only consider the marks that are greater than or equal to 50 and the courses offered in the year 2008 `(semesters.year = '2008')`. The view should be ordered by the average mark in descending order.

Question 7 (5 marks)

Define a SQL view `Q7(student, course)` that gives the student IDs (`people.id`) and the course IDs (`courses.id`) of all students who achieved the highest mark in the course. Only consider the COMP93 courses (`subjects.code` contains `'COMP93'` as a prefix) that were offered in Semester 1 of 2008. If there are multiple students who achieved the highest mark, return all of them.

Question 8 (6 marks)

Define a SQL view `Q8(course_id, staffs_names)` that gives the course IDs (`courses.id`) of all courses that have at least 650 students enrolled in them. Only include courses that have exactly 2 AProf staffs (`people.title = 'AProf'`). For each course, list the given names of all AProf staff in ascending order and concatenate them with commas. For example, if there are two staff, `'John'` and `'Jane'`, the output should be: `'Jane, John'`.

Question 9 (6 marks)

Define a PL/pgSQL function `Q9(subject_code)` that takes a subject code (`subjects.code`) and returns all its direct prerequisites. A subject code `c2` is a direct prerequisite of code `c1` if the field `_prereq` of the subject of `c1` contains the `c2` a substring. If there is no prerequisite, return `'There is no prerequisite for subject <subject_code>.'`; otherwise, returns `'The prerequisites for subject <subject_code> are <prerequisite1>, <prerequisite2>, ... <prerequisite2>.'` (replace <...> with the actual codes, order by the codes in ascending order). (Note that different subjects may have the same subject code. To walk around this issue, for a subject code, you should consider all the subjects with that code.)

Question 10 (7 marks)

Define a PL/pgSQL function `Q10(subject_code)` that takes a subject code (`subjects.code`) and returns all its (direct and indirect) prerequisites. If code `c1` is a prerequisite for `c2`, `c2` is a prerequisite for `c3`, then `c1` is also a prerequisite for `c3`. If there is no prerequisite, return `'There is no prerequisite for subject <subject_code>.'`; otherwise, returns `'The prerequisites for subject <subject_code> are <prerequisite1>, <prerequisite2>, ... <prerequisite2>.'` (replace <...> with the actual codes, order by the codes in ascending order). (Note that different subjects may contain the same subject code. To walk around this issue, for a subject code, you should consider all the subjects with that code.)

6 Test your solution

We provided test function(s) for each of the questions. These test functions can be used to check if your solution has the expected output. These functions and expected outputs can be found in check.sql file. The following is a procedure for testing your solution using check.sql.

```
$ dropdb proj1                                          ... remove any existing DB
$ createdb proj1                                        ... create an empty database
$ psql proj1 -f /home/cs9311/web/24T2/proj/proj1/mymyunsw.dump
                                                        ... load the MyMyUNSW schema &
data
$ psql proj1 -f /home/cs9311/web/24T2/proj/proj1/check.sql
                                                        ... load the checking code
$ psql proj1 -f proj1.sql                               ... load your solution
$ psql proj1
proj1=# select check_q1();                              ... check your solution to question1
...
proj1=# select check_q6();                              ... check your solution to question6
...
proj1=# select check_q9a();                             ... check your solution to question9(a)
...
proj1=# select check_q10d();                            ... check your solution to question10(d)
...
proj1=# select check_all();                             ... check all your solutions
```

Notes:

1. You must ensure that your submitted proj1.sql file will be loaded and run correctly (i.e., it has no syntax errors, and it contains all your view definitions in the correct order).
   a. If your database contains any views that are not available in a file somewhere, you should put them into a file before you drop the database.
   b. For all the submission files, you must make sure there is no error occurring when using the autotest provided above. If we need to manually fix problems in your proj1.sql file to test it (e.g., change the order of some definitions), you will be fined half of the mark as penalty for each problem.
2. In addition, write queries that are reasonably efficient.

For each question, the result must be produced within 120 seconds on the nw-syd-vxdb server. Failure to do so will incur a penalty, deducting half of the mark. This time constraint applies to executing the command 'select * from check_Qn()'.

7 Project Submission

Final check before submission

We're aware that many students are doing this project on their own machine, this should be fine. However, the PostgreSQL installed on your own laptop may not be compatible with the one installed in nw-syd-vxdb, which may cause you to lose marks unnecessarily, as we will test your solution on the nw-syd-vxdb server. Hence, before you submit, you have to check your solution on the nw-syd-vxdb server with a newly created database. Please follow the detailed steps presented in Section 6: 'Test your solution'.

Submission

You can submit this project by doing the following:

- You are required to submit an electronic version of your answers via **Moodle**.
- We only accept the **.sql** format. Please name your files in the following format to submit: proj1_**zID**.sql (e.g., **proj1_z5000000.sql**).
- Only the **Latest Submission** is marked. The Latest Submission after the deadline will result in a late penalty.
- In case the system is not working properly, please ensure to take these steps: keep a copy of your submitted file on the CSE server without any post-deadline modifications. If you're uncertain how to do this, refer to the guidelines on Taggi.

The proj1_zID.sql file should contain answers to all the exercises for this project. It should be completely self-contained and able to load in a single pass, so that it can be auto-tested as follows:

- A fresh copy of the MyMyUNSW database will be created (using the schema from mymyunsw.dump).
- The data in this database may differ from the database you're using for testing.
- A new check.sql file may be loaded (with expected results appropriate for the database).
- The contents of your proj1_zID.sql file will be loaded.
- Each checking function will be executed, and the results will be recorded.

8 Late Submission Penalty

- 5% of the total mark (50 marks) will be deducted for each additional day.
- Submissions that are more than five days late will not be marked.

9 Plagiarism

The work you submit must be your own work. Submission of work partially or completely derived from any other person or jointly written with any other person is not permitted. The penalties for such an offence may include negative marks, automatic failure of the course and possibly other academic discipline.

All submissions will be checked for plagiarism. The university regards plagiarism as a form of academic misconduct and has very strict rules. Not knowing the rules will not be considered a valid excuse when you are caught.

- For UNSW policies, penalties, and information to help avoid plagiarism, please see: https://student.unsw.edu.au/plagiarism.
- For guidelines in the online ELISE tutorials for all new UNSW students: https://subjectguides.library.unsw.edu.au/elise/plagiarism.