COMP9311 24T3: Project 1

Deadline: 5pm Monday, 21 October

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 private 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.
- 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.
- Submit the project via Moodle.
- For each question, you must output the result within 120 seconds on the nw-syd-vxdb server.
- **Hardcode is strictly forbidden.**

3. Setting Up

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/24T3/proj/mymyunsw.dump
```

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

```
SET
SET
SET
SET
SET
```

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 under 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: remove any existing databases before loading your MyMyUNSW database.

Summary on Getting Started

To set up your database for this project, run the following commands in the order supplied:

$ **mkdir *Project1Directory***
$ **cd *Project1Directory***
Download from WebCMS the proj1.sql and check.sql and put them in *Project1Directory*
$ **createdb proj1**
$ **psql proj1 -f /home/cs9311/web/24T3/proj/mymyunsw.dump**
$ **psql proj1 -f check.sql**
Editing your proj1.sql, when you finished, loading it with
$ **psql proj1 -f proj1.sql**

If you omit any of the steps, then things may 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.*

```
$ psql proj1
proj1=# \d
... study the schema ...
proj1=# select * from Students;
... look at the data in the Students table ...
proj1=# select p.unswid,p.name from People p join Students s on (p.id=s.id);
... look at the names and UNSW ids of all students ...
proj1=# select p.unswid,p.name,s.phone from People p join Staff s on (p.id=s.id);
... look at the names, staff ids, and phone #s of all staff ...
proj1=# select count(*) from Course_Enrolments;
... get an idea of the number of records each table has...
proj1=# select * from dbpop();
... how many records in all tables ...
proj1=# ... etc. etc. etc.
proj1=# \q
```

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.
- **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 view Q1(count) to count the number of students who have ever got a mark (course_enrolments.mark) greater than 85 in COMP courses. COMP course refers to subjects.code that starts with 'COMP'.

Question 2 (4 marks)

Define a view Q2(count) to count the number of students whose average mark of all COMP courses the student has taken is greater than 85. Only consider the non-NULL marks.

Question 3 (5 marks)

Define a view Q3(unswid,name) to find the unswid (people.unswid) and name (people.name) of students whose average mark of all COMP courses the student has taken is greater than 85. Only consider the students who have taken at least 6 COMP courses and non-NULL marks. If the student has taken multiple courses of the same subject in different terms/semesters, consider them as two different courses.

Question 4 (6 marks)

Define a view Q4(unswid,name) to find the unswid (people.unswid) and name (people.name) of students whose WAM of all COMP courses the student has taken is greater than 85. Only consider those students who have taken at least 6 COMP courses of different subjects and non-NULL marks. If the student has taken multiple courses with the same subject in different terms/semesters, only consider the one with the highest mark. WAM refers to the weighted average mark, it is computed as sum(mark * uoc)/sum(uoc) where uoc refers to subjects.uoc.

Question 5 (4 marks)

Define a view Q5(count) to count the number of subjects offered by CSE (orgunits.longname='School of Computer Science and Engineering') in the year of 2012 (Semesters.year). Please note that it is not guaranteed that this subject is offered in the years between the year of subjects.firstoffer and subjects.lastoffer.

Question 6 (4 marks)

Define a view Q6(count) to count the number of teaching stuff who is a Course Lectuer (staff_roles.name='Course Lecturer') of a course in the year of 2012 (Semesters.year) and the teaching stuff who is from the CSE (orgunits.longname='School of Computer Science and Engineering'). A staff is from CSE refers to that in the affiliations, the staff holds a role at CSE. (Note that a staff may have multiple roles)

Question 7 (5 marks)

Define a view Q7(course_id,unswid) to find the courses (courses.id) and the Course Lectuers (people.unswid) of that course such that the course is offered by CSE in the year of 2012 and the Course Lectuers are also from CSE.

Question 8 (6 marks)

Define a view Q8(course_id,unswid) to find the courses (courses.id) and the Course Lectuers (people.unswid) of that course such that the course is offered by CSE in the year of 2012 and all the Course Lectuers of the course are from CSE.

Question 9 (6 marks)

Define a PL/pgSQL function Q9(subject1 integer,subject1 integer) which takes the id of two subjects and checks if the first is a direct prerequisite of the second. Return a text "subject1 is a direct prerequisite of subject2." or "subject1 is not a direct prerequisite of subject2.". subject1 is a direct prerequisite of subject2 if the field _prereq of subject2 contains the code (subjects.code) of subject1 as a substring (Note that some of the _prereq is a long text, you don't have to parse the semantic meaning of _prereq).

Question 10 (6 marks)

Define a PL/pgSQL function Q10(subject1 integer,subject1 integer) which takes the id of two subjects and checks if the first is a (direct or indirect) prerequisite of the second. Return a text "subject1 is a prerequisite of subject2." or "subject1 is not a prerequisite of subject2.". In this question, the prerequisite of a prerequisite is also a prerequisite. (Due to the definition of prerequisite in Q9, a subject may be a prerequisite of itself, so we accept the case where "subject1 is a prerequisite of subject1").

6. AutoTest Checking

Before you submit your solution, you should check and test its correctness by using the following operations. **For each PostgreSQL question, we provide several testcases**. The testcases can be found in check.sql file.

```
$ dropdb proj1                                  ... remove any existing DB
$ createdb proj1                                ... create an empty database
$ psql proj1 -f /home/cs9311/web/24T3/proj/mymyunsw.dump
                                                ... load the MyMyUNSW schema & data
$ psql proj1 -f <path to the check.sql>
                                                ... load the checking code
$ psql proj1 -f <path to your 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_q10e();                    … check your solution to question10(e)
…
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.
   a. 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.