Project 1 SQL
COMP9311 23T3

The deadline for project 1 is: Oct 27th 16:59:59 (Sydney Local Time)

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** (SQL schema)
- 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 qX_expected tables loaded as part of the **check.sql** file
- Solve each of the problems below, 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
- 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 result within 120 seconds on vxdb server.
- **Hardcode is strictly forbidden.**

3. Introduction

All Universities require a significant information infrastructure in order to manage their affairs. This typically involves a large commercial DBMS installation. UNSW's student information system sits behind the MyUNSW web site. MyUNSW provides an interface to a PeopleSoft enterprise management system with an underlying Oracle database. This back-end system (Peoplesoft/Oracle) is often called NSS.

UNSW has spent a considerable amount of money ($80M+) on the MyUNSW/NSS system, and it handles much of the educational administration plausibly well. Most people gripe about the quality of the MyUNSW interface, but the system does allow you to carry out most basic enrolment tasks online.

Despite its successes, MyUNSW/NSS still has several deficiencies, including:

- no waiting lists for course or class enrolment
- no representation for degree program structures
- poor integration with the UNSW Online Handbook

The first point is inconvenient, since it means that enrolment into a full course or class becomes a sequence of trial-and-error attempts, hoping that somebody has dropped out just before you attempt to enroll and that no-one else has grabbed the available spot.

The second point prevents MyUNSW/NSS from being used for three important operations that would be extremely helpful to students in managing their enrolment:

- finding out how far they have progressed through their degree program, and what remains to be completed
- checking what are their enrolment options for next semester (e.g., get a list of available courses)
- determining when they have completed all the requirements of their degree program and are eligible to graduate

NSS contains data about student, courses, classes, pre-requisites, quotas, etc. but does not contain any representation of UNSW's degree program structures. Without such information in the NSS database, it is not possible to do any of the above three. So, in 2007 the COMP9311 class devised a data model that could represent program requirements and rules for UNSW degrees. This was built on top of an existing schema that represented all the core NSS data (students, staff, courses, classes, etc.). The enhanced data model was named the MyMyUNSW schema.

The MyMyUNSW database includes information that encompasses the functionality of NSS, the UNSW Online Handbook, and the CATS (room allocation) database. The MyMyUNSW data model, schema and database are described in a separate document.

4. Setting Up

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

```
$ createdb proj1
$ psql proj1 -f /home/cs9311/web/23T3/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 needs 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 vxdb, assuming that vxdb is not under heavy load. (If you leave your project until the last minute, loading the database on 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 /localstorage directory is not a good idea).

If you have other large databases under your PostgreSQL server on vxdb or if you have large files under your /localstorage/YOU/ directory, it is possible that you will exhaust your vxdb disk quota. Regardless, it is certain that you will not be able to store two copies of the MyMyUNSW database under your 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:

$ **createdb proj1**
$ **psql proj1 -f /home/cs9311/web/23T3/proj/proj1/mymyunsw.dump**
$ **psql proj1**
... run some checks to make sure the database is ok
$ **mkdir *Project1Directory***
... make a working directory for Project 1
$ **cp /home/cs9311/web/23T3/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.

5. 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.
- When queries ask for people's names, use the Person.name field; it's there precisely to produce displayable names.
- When queries ask for student ID, use the People.unswid field; 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) once we release it.

6. 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 (3 marks)

Define a SQL view `Q1(course_code)` that gives the code of subjects that is equivalent to level 3 HIST courses (i.e., with a course code of the format HIST3***)

- `course_code` should be taken from `Subjects.code` field.

Question 2 (3 marks)

Define a SQL view `Q2(course_id)` that gives the distinct id of the courses that have over 400 local students enrolled.

- `course_id` should be taken from `Courses.id` field;
- local students refer to students whose `Students.stype` is '`local`'.

Question 3 (3 marks)

Define a SQL view `Q3(course_id)` that gives the distinct id of courses that only have 4 lectures, and each lecture taking place in a different building.

- `course_id` should be taken from `Courses.id` field;
- buildings refer to `Rooms.building`;
- Lecture refers to the class where its `Class_types.name` is '`Lecture`'.

Question 4 (4 marks)

Define a SQL view `Q4(unsw_id)` that gives the distinct id of students who only fail course in the semester `2011 X1`.

- `unsw_id` should be taken from `People.unswid` field;
- Failing a course means that the grade a student received for a course is '`FL`'.

Question 5 (4 marks)

Define a SQL view `Q5(course_code)` that gives the code of the courses that have the highest number of students failed among all the courses offered by the same faculty in the year `2010`. If there are multiple courses sharing the highest number of failed students, list all the `course_code` of these courses.

- `course_code` should be taken from `Subjects.code` of the corresponding subject;
- Faculties refer to the organization units where their `Orgunit_types.name` are '`Faculty`';
- Failing a course means that the grade a student received for a course is '`FL`'.

Question 6 (4 marks)
Define a SQL view `Q6(course_code, lecturer_name)` that gives the code of all the COMP courses (i.e., with a course code of the format COMP****) and the name of the corresponding lecturers who taught in the semester that achieved the highest average mark compared to all other semesters with the same course code (i.e. all the courses with the same `Courses.subject`). If there are multiple lecturers sharing the same highest average mark for the same `course_code`, list all of them.

- `course_code` should be taken from `Subjects.code` field of the corresponding subject;
- `lecturer_name` should be taken from `People.name` field;
- Lecturer refers the staff whose `Staff_roles.name` is '`Course Lecturer`';
- Mark refers to `Course_enrolments.mark` field;
- Do not consider courses without lecturers;
- When calculating the average mark, we do not count students whose mark is null.

Question 7 (4 marks)
Define a SQL view `Q7(semester_id)` that gives the id of semesters where the number of full-time students (i.e., students who enrolled in at least 4 courses within this semester) enrolled in programs offered by the Faculty of Engineering are more than those enrolled in programs offered by School of Mechanical and Manufacturing Engineering.

- `semester_id` should be taken from `Semesters.id` field;
- Faculty of Engineering refers to the organization unit where its `Orgunits.longname` is '`Faculty of Engineering`';
- School of Mechanical and Manufacturing Engineering refers to the organization unit where its `Orgunits.longname` is '`School of Mechanical and Manufacturing Engineering`'.

Question 8 (5 marks)
Define SQL view `Q8(unsw_id)` that gives the id of students who were enrolled in the same stream in their bachelor and master degree. Additionally, their average mark in the master degree should be higher than the average mark in the bachelor degree.

- `unsw_id` should be taken from `People.unswid` field;
- Bachelor degrees refer to the programs where `Program_degrees.name` contains '`Bachelor`' in a case-insensitive manner;
- Master degrees refer to the programs where `Program_degrees.name` contains 'Master' in a case-insensitive manner;
- When calculating the average mark of a program, we consider a course belong to a program if a student enrolled in this course and this program in the same semester (refers to `Semestsers.id`);
- When calculating the average mark, we do not consider the courses where students received a null mark.

Question 9 (5 marks)
Define SQL view `Q9(lab_id, room_id)` that gives the id of GEOS labs held in year `2007`(refer to `Semesters.year`) and corresponding id of rooms where the rooms of these labs do not have both slide projector and laptop connection facilities at the same time.

- GEOS labs refer to the classes where the corresponding course code(`Subjects.code`) with the format '`GEOS****`' and their `Class_types.unswid` is '`LAB`';
- Slide projector refers to the facility where its description contains '`Slide projector`' in a case-insensitive manner;
- Laptop connection facilities refers to the facility where its description contains '`Laptop connection facilities`' in a case-insensitive manner;
- `lab_id` should be taken from `Classes.id` field;
- `room_id` should be taken from `Rooms.id` field.

Question 10 (5 marks)

Define SQL view `Q10(course_id, hd_rate)` that gives the id of course and the corresponding HD rate of this course. We only consider the courses where the course convenor is a research fellow of `School of Chemical Engineering`.

Round hd_rate to the nearest 0.0001. (i.e., if hd_rate = 0.01 (i.e., 1%), then return 0.0100; if hd_rate = 0.01234, then return 0.0123; if hd_rate = 0.02345, then return 0.0235). This rounding behavior is different from the IEEE 754 specification for floating point rounding which PostgreSQL uses for float/real/double precision types. PostgreSQL only performs this type of rounding for numeric and decimal types.

- hd rate = (number of students with mark >= 85 ÷ number of students with mark);
- Only count the students with valid marks(refer to `Course_enrolments.mark`) that are not null.
- Course convenor refers to the staff of a course where the `Staff_roles.name` is '`Course Convenor`';
- Research fellow refers to the staff of an organization where the `Staff_roles.name` is '`Research Fellow`';
- To find the research fellow of a certain organization unit, you should check the `Affiliations` table;
- Do not include the courses that have no student enrolled;
- `course_id` should be taken from `Courses.id`;
- `hd_rate` should be in numeric type.

Question 11 (5 marks)

Define SQL view `Q11(unsw_id)` that gives the id of students who are eligible for scholarship. To qualify for the scholarship, a student should enroll into a program and meet these criteria within this program:

a) the program is offered by `School of Computer Science and Engineering` (refer to `orgunits.longname`);
b) Have earned over 60 UOC in non-high level COMP courses;
c) Have earned over 24 UOC in high level COMP courses with streams;
d) Their average marks of the high level COMP courses with streams must exceed 80.

From all eligible candidate students, return the top 10 students with the highest average marks in high-level COMP courses with streams.

Note:

- COMP courses refer to the course code with the format 'COMP****';

- High level COMP courses refer to the course code with the format 'COMP4***', 'COMP6***', 'COMP8***' or 'COMP9***';
- A student can only earn the UOC of the courses he/she pass, i.e., the mark for the course (refers to `Course_enrolments.mark`) should be no less than 50.
- If a student has enrolled into several different programs, you need to calculate the UOC and the average mark separately according to different programs. A course is included in a program if this student enrolled into the course and the program in the same semester (refer to `semester.id`);
- In criteria c) and d), the uoc and average mark are calculated at the program level, not stream level. If a student is enrolled in different streams, "with stream" indicates that you only need to consider uoc and marks of courses taken in semesters where the student was enrolled in any stream(refer to `stream_enrolments`), regardless of which specific stream it was.
- If multiple students achieved the same average mark, they should be assigned with the same ranking. The Rank() function in PostgreSQL will be able to do this for you to generate the ranking column.

Example: Say a student has enrolled in Bachelor of Computer Science and Master of Computer Science.

  - *Scenario A*
    He meets criteria a) and b) in Bachelor of Computer Science and meets criteria a), c) and d) in Master of Computer Science. Since he does not satisfy all 4 criteria in either of the programs, he would not be considered for the scholarship.
  - *Scenario B*
    He meets criteria a), b), c) and d) in Bachelor of Computer Science. Then he may be considered for the scholarship, depending on whether he ranked top 10.

Question 12 (5 marks)
Define a PL/pgSQL function `Q12(course_id Integer, i Integer)` that takes the id of a course and an integer, and returns a set of `student_id` whose `mark` ranked i^{th} in that course.

- `course_id` is taken from `Courses.id` field;
- `student_id` is taken from `Students.id` field;
- `mark` refers to `Course_enrolments.mark` field;
- `i` is a positive integer.

Each line of the output (in `text` type) should contain one element which is `student_id`.

Example: If 5 students enrolled in COMP9311 23T3 with a course id of 100, and their student ids, marks and ranks are as follows. If the input course id matches 100 and input parameter `i=1`, the returned result should have only one student_id 3. If `i=2`, the result should contain two student_ids: 2 and 5. If `i=3` or `i>5`, then the result should not contain any row. You don't need to consider any other invalid inputs.

student_id	mark	rank
1	70	4
2	80	2
3	90	1
4	60	5
5	80	2

7. Submission

You can submit this project by doing the following:

- Students must submit an electronic copy of their answers to the above questions to the course website in Moodle.
- The file name should be proj1_**studentID**.sql (e.g., **proj1_z5100000.sql**).
- If you submit your project more than once, the last submission will replace the previous one
- In case that the system is not working properly, you must take the following actions:
- Please keep a copy of your submitted file on the CSE server. If you are not sure how, please have a look at taggi.

The proj1.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 be **different** from the database that 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.sql file will be loaded
- Each checking function will be executed, and the results recorded

8. Check your Answers

Before you submit your solution, you should check that it loads correctly for testing by using something like the following operations. For function questions, we provide five testcases for each question (E.g., for question 10, they are q10a to q10e). Testcases can be found from line220 in check.sql:

$ **dropdb proj1** ... remove any existing DB
$ **createdb proj1** ... create an empty database
$ **psql proj1 -f /home/cs9311/web/23T3/proj/proj1/mymyunsw.dump** ... load the MyMyUNSW schema and data
$ **psql proj1 -f /home/cs9311/web/23T3/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_q12a();** … check your solution to question12 testcase (a)
proj1=# **select check_all();** … check all your solutions

Notes:

1. You must ensure that your proj1.sql file will load and runs 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. If we need to manually fix problems with your proj1.sql file in order to test it (e.g., change the order of some definitions), you will be fined via half of the mark penalty for each problem.
c. If your code loads with errors, fix it and repeat the above until it does not.

2. In addition, write queries that are reasonably efficient.
   a. For each question, you must output result within 120 seconds on vxdb server. This time restriction applies to the execution of the 'select * from check_Qn()' calls.
   b. For each question, you will be fined via half of the mark penalty if your solution cannot output results within 120 seconds.

9. Late Submission Penalty

- 5% reduction of max mark (-2.5 out of 50) will be deducted for each day late (1 second late is considered as one day late), up to 5 days (5*24 hours).
- Submissions that are more than five days late will not be marked.