CIT Assignment 1 – Querying with SQL

The main purpose of this assignment is to practice querying using SQL. The target is a larger university database with 2000 students and around 28.500 course enrollments (the takes relation). You can build this database on your local PostgreSQL server by running a script **university_large.sql**. Check out the instructions by the end of this text on: building the database, producing an output file and handing in.

The schema is exactly the same as that for the small database. To present an overview of the database, the schema diagram from the DB-book is shown below.

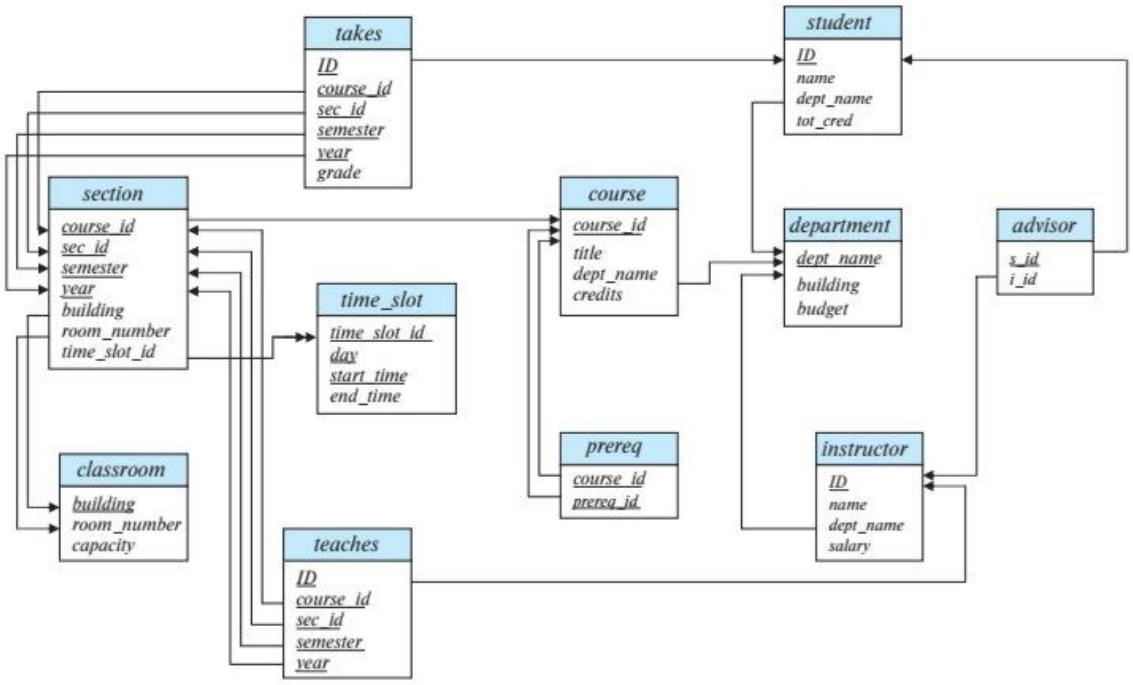

Figure 2.9 Schema diagram for the university database.

Notice that the primary keys are shown (underlined) and the foreign keys are indicated by arrows. Some of the primary keys (and thereby also some of the foreign keys) are composite.

The **university_large.sql** script contains SQL insert statements for larger, randomly created relations for a truly strange university (since course titles and department names are chosen randomly).
The 11 tables have the row counts shown to the right.
Try to study the content of the database before you begin. Run some example queries and check the type of values used for e.g. keys.

advisor	2000
classroom	30
course	200
department	20
instructor	50
prereq	100
section	103
student	2000
takes	28544
teaches	103
time_slot	20

What to do

Write SQL queries to answer the questions below. Notice that query-answers are listed, so for each you will "simply" have to find an SQL-expression, that produces the given answer.
See instructions, at the end of this note.

Important: Work individually, then work in groups, then hand-in by group

You are supposed to submit one hand in per group and we strongly recommend you to work with the questions individually as well as in groups. Discuss and decide on what you consider to be the best solutions and hand these in.

Questions

Observe: Most of the queries to answer the questions below should return almost immediately. Stop your query evaluation if you experience long waiting and reconsider your query expression. You can skip those questions that you cannot find an answer to.

-- 1. Find the names of all the instructors from Biology department

```
   name
----------
 Queiroz
 Valtchev
(2 rows)
Time: 2.481 ms
```

-- 2. Find the names of courses in Computer science department which have 3 credits

```
         title
----------------------
 International Finance
 Japanese
 Computability Theory
(3 rows)
Time: 1.146 ms
```

-- 3. For the student with ID 30397, show all course_id and title of all courses registered for by the student.

```
 course_id |          title
-----------+--------------------------
 105       | Image Processing
 158       | Elastic Structures
 200       | The Music of the Ramones
 319       | World History
 349       | Networking
 461       | Physical Chemistry
 468       | Fractal Geometry
 496       | Aquatic Chemistry
 626       | Multimedia Design
 631       | Plasma Physics
 642       | Video Gaming
 702       | Arabic
 760       | How to Groom your Cat
 795       | Death and Taxes
 959       | Bacteriology
 960       | Tort Law
(16 rows)
Time: 1.352 ms
```

-- 4. As above, but show the total number of credits for such courses (taken by that student).
-- Don't display the tot_creds value from the student table, you should use SQL aggregation on courses taken by the student.

```
 course_id |          title           | sum
-----------+--------------------------+-----
 105       | Image Processing         |   6
 158       | Elastic Structures       |   3
 200       | The Music of the Ramones |   4
 319       | World History            |   4
 349       | Networking               |   4
 461       | Physical Chemistry       |   3
 468       | Fractal Geometry         |   8
 496       | Aquatic Chemistry        |   3
 626       | Multimedia Design        |   4
 631       | Plasma Physics           |   4
 642       | Video Gaming             |   3
```

```
 702       | Arabic                   |   3
 760       | How to Groom your Cat    |   3
 795       | Death and Taxes          |   3
 959       | Bacteriology             |   4
 960       | Tort Law                 |   3
(16 rows)
Time: 0.468 ms
```

-- 5. Now display the total credits (over all courses) for each of the students having more than 85 in total credits, along with the ID of the student;
-- don't bother about the name of the student.
-- don't bother about students who have not registered

```
  id   | sum
-------+-----
 72669 |  87
 50039 |  86
 47379 |  86
 107   |  86
 44551 |  87
 12078 |  93
(6 rows)
Time: 36.310 ms
```

-- 6. Find the names of all students who have taken any course at the Languages department with the grade 'A+' (there should be no duplicate names)

```
     name
--------------
 Abraham
 Bhattacharya
 Boldin
 Carey
 Cheah
 Ching
 Cochran
 Damas
 Denso
 Ebou
 Frangeu
 Geißl
 Hughes
 Januszewski
 Kirtane
 Komori
 Kwan
 Macias
 Masri
 Nakajima
 Oki
 Oller
 Palomo
 Paniez
 Patne
 Pavlovico
 Planti
 Roses
 Saill
 Savolainen
 Thimm
 Vries
 Wingb
 Wood
 Zafar
(35 rows)
Time: 7.196 ms
```

-- 7. Display the IDs of all instructors from the Marketing department who have never taught a course (interpret "taught" as "taught or is scheduled to teach")

```
  id
-------
 58558
 96895
 74426
(3 rows)
Time: 1.017 ms
```

-- 8. As above, but display the names of the instructors also, not just the IDs.

```
  id   |   name
-------+----------
 96895 | Mird
 74426 | Kenje
 58558 | Dusserre
(3 rows)
Time: 0.474 ms
```

-- 9. Using the university schema, write an SQL query to find the number of students in each section in year 2009.
-- The result columns should be "course_id, sec_id, year, semester, num", where the latter is the number.
-- You do not need to output sections with 0 students.

```
 course_id | sec_id | semester | year | num
-----------+--------+----------+------+-----
 972       | 1      | Spring   | 2009 | 280
 960       | 1      | Fall     | 2009 | 307
 304       | 1      | Fall     | 2009 | 307
 604       | 1      | Spring   | 2009 | 300
 105       | 1      | Fall     | 2009 | 327
 334       | 1      | Fall     | 2009 | 268
 237       | 2      | Fall     | 2009 | 311
 486       | 1      | Fall     | 2009 | 304
(8 rows)
Time: 7.475 ms
```

-- 10. Find the maximum and minimum enrollment across all sections, considering only sections that had some enrollment, don't worry about those that had no students taking that section.
-- 10a. using a subquery in from

```
 max | min
-----+-----
 338 | 264
(1 row)
Time: 36.239 ms
```

-- 10b. using a with statement

```
 max | min
-----+-----
 338 | 264
(1 row)
Time: 19.227 ms
```

-- 11. Find all sections that had the maximum enrollment (along with the enrollment),
-- 11a. using a subquery.

```
 course_id | sec_id | semester | year | num
-----------+--------+----------+------+-----
 362       | 1      | Fall     | 2005 | 338
 192       | 1      | Fall     | 2002 | 338
(2 rows)
Time: 74.180 ms
```

-- 11b. using a with clause (e.g. defining temporary enrollment and maxnumber tables)

```
 course_id | sec_id | year | semester | num
-----------+--------+------+----------+-----
 362       | 1      | 2005 | Fall     | 338
 192       | 1      | 2002 | Fall     | 338
(2 rows)
Time: 19.809 ms
```

-- 12. As in in Q10, but now also include sections with no students taking them; the enrollment for such sections should be treated as 0.
-- Use aggregation and outer join -- use aggregation on a left outer join

```
 max | min
-----+-----
 338 |   0
(1 row)
Time: 56.797 ms
```

-- 13. Find all courses that the instructor with id '19368' have taught

```
  id   | course_id | sec_id | semester | year
-------+-----------+--------+----------+------
 19368 | 581       | 1      | Spring   | 2005
 19368 | 545       | 1      | Fall     | 2001
 19368 | 591       | 1      | Spring   | 2005
(3 rows)
Time: 0.469 ms
```

-- 14. Find instructors who have taught all the above courses
-- (use "... not exists ... except ...")

```
  id
-------
 19368
 41930
(2 rows)
Time: 20.633 ms
```

-- 15. Insert each instructor as a student, with tot_creds = 0, in the same department

```
INSERT 0 47
Time: 7.700 ms
```

-- 16. Now delete all the newly added "students" above (note: already existing students who happened to have tot_creds = 0 should not get deleted)

```
DELETE 47
Time: 5.191 ms
```

-- 17. Some of you may have noticed that the tot_cred value for students do not always match the credits from courses they have taken.

```
--      Write a query to compare these that show a students id and
tot_cred together with the correct calculated total credits.
--      Show only the cases where the two values match
```

```
  id   | tot_cred | sum
-------+----------+-----
 30177 |       41 |  41
 38336 |       39 |  39
 39520 |       43 |  43
 14094 |       45 |  45
 8378  |       47 |  47
 48901 |       57 |  57
 54508 |       41 |  41
 41596 |       51 |  51
 44584 |       58 |  58
 16907 |       59 |  59
 29920 |       62 |  62
 61232 |       48 |  48
 57787 |       60 |  60
 67340 |       38 |  38
 63390 |       41 |  41
(15 rows)
Time: 44.004 ms
```

```
-- 18.    Write and execute a query to update tot_cred for all
students to the correct calculated value based on the credits passed
- thereby bringing the database back to consistency.
```

```
UPDATE 2000
Time: 266.306 ms
```

```
-- 19. Run Q17 again, but now only show when the two values differ
```

```
 id | tot_cred | sum
----+----------+-----
(0 rows)
Time: 44.599 ms
```

```
-- 20.    Update the salary of each instructor to 29001 + 10000
times the number of course sections they have taught.
```

```
UPDATE 50
Time: 1.791 ms
```

```
-- 21. List name and salary for the 10 first instructors (alphabetic
order)
```

```
  id   |   name    |  salary
-------+-----------+----------
 37687 | Arias     | 29001.00
 95030 | Arinb     | 29001.00
 28400 | Atanassov | 49001.00
 52647 | Bancilhon | 29001.00
 15347 | Bawa      | 39001.00
 97302 | Bertolino | 29001.00
 90376 | Bietzk    | 39001.00
 34175 | Bondi     | 59001.00
 3335  | Bourrier  | 49001.00
 90643 | Choll     | 39001.00
(10 rows)
Time: 0.233 ms
```

How to (re)build the larger university database

You can download and install the database on your own computer. Do the following:

- Download the SQL script file **university_large.sql** from the CIT Moodle site.
- Open your command line interface and change to the directory where you placed the file **university_large.sql**.

- Run the two commands
  psql -U postgres -c "create database university_large"
  psql -U postgres -d university_large -f university_large.sql
- You may consider to drop the database if you need to start with a fresh database content while working or when you are finished using the database for Assignment 1. One way to drop the database is by using this command:
  psql -U postgres -c "drop database university_large"

How to produce an output file and hand in your solution

When you have tested all your solutions to the questions one by one, include all these in a single SQL script file **citXX-assignment1.sql** (where x is your group number) with content like:

```
-- GROUP: citXX, MEMBERS: <name1>, <name2>, ...
-- 1.
SELECT name from instructor where dept_name='Biology';

-- 2.
SELECT …
…

-- 3.
SELECT …
…
…
```

To indicate "no solution" for a question, just write "-- no solution" in place of the SELECT expression in the script. To include timings in the output (not required, not shown above) you can add a first line: **\timing.** To hand in your solution do the following.

- Generate an output file **citXX-assignment1.txt** (where XX is your group number) by running the command:
  psql -U postgres -a -d university_large -f citXX-assignment1.sql > citXX-assignment1.txt
  What's going on here is that all the SQL-code in your input file **citXX-assignment1.sql** will be processed and the output will be written to the file **citXX-assignment1.txt** including echoes of the SQL expressions and the -- comments (the -a option takes care of the echoing). See all the available options for the psql command is [PGMAN] psql command.
- Check your output file **citXX-assignment1.txt** and hand it in together with your script file **citXX-assignment1.sql** on the Moodle page.
  OBS: Just one hand-in per group, but remember to list all group members names in the beginning of the script file

Main reference for this assignment is:

- [DSC] Database System Concepts, Abraham Silberschatz, Henry Korth, S. Sudarshan, 7th Edition, 2019, chapter 3 and 4

To read more about how to use the command line tool **psql** consider:

- PostgreSQL Tutorial, 17 Practical psql Commands That You Don't Want To Miss
  http://www.postgresqltutorial.com/psql-commands/
- [PGMAN] psql command
  https://www.postgresql.org/docs/current/app-psql.html