March 2024

Exercise 1^{A} An Elementary Student Registry

A1. Schema creation

In the PostGIS database, created in your name on the server 147.102.40.25:5432, create schema **exercise1**, with the necessary tables for the "elementary student registry" (Figure 1 below & relevant descriptions on the presentation slides) by running appropriate SQL statements (queries) from the PgAdmin interface.
The **person** table should have a serial PRIMARY KEY with an initial value of 202401. It should also include **sdb**:boolean and **comp_methods**:boolean columns, where the relationship with the respective courses (SPATIAL DATADASES and COMPUTATIONAL METHODS IN GI) is declared.
The **room** table should also have a serial PRIMARY KEY, with an initial value of 202401.

A2. Database population

In this database, you should enter data for professors and students of the Interdepartmental GEOINFORMATICS (GI) Program, acad. years 2023 and 2024. Your data sources will be the following:
a) The database **tutor**, schema **examle1**, table **student** (you have read-only access)
In this schema, issue appropriate query to get the data for the students of academic years 2023 and 2024: **select * from example1.student where …**
The query output should be saved to a csv file on your computer. Then insert this data into your database programmatically (in whatever programming language you want - preferably python, based on **Hint 2** at the end of this file).
b) The list of students who have chosen the SDB course (file gis-student_list-sdb2023.pdf). Based on this list, update **sdb**.
c) List of teachers

idperson	name	givenname	department	university	course
201950	ΜΗΤΡΟΥ	ΝΙΚΟΛΑΟΣ	ΣΗΜΜΥ	ΕΜΠ	SDB
201951	ΒΕΣΚΟΥΚΗΣ	ΒΑΣΙΛΕΙΟΣ	ΣΑΤΜ-ΜΓ	ΕΜΠ	SDB
201952	ΘΕΟΔΩΡΙΔΗΣ	ΙΩΑΝΝΗΣ		ΠΑΠΕΙ	SDB
201953	ΖΑΦΕΙΡΟΠΟΥΛΟΣ	ΑΝΑΣΤΑΣΙΟΣ	ΣΗΜΜΥ	ΕΜΠ	COMPUT. METHODS in GI
201954	ΚΟΚΛΑ	ΜΑΡΓΑΡΙΤΑ	ΣΑΤΜ-ΜΓ	ΕΜΠ	SDB

Subsequently:
d) Update the **personcourse** table appropriately.
Finally, with an appropriate SQL statement, set the **comp_methods** column to TRUE on students with an even **idperson**.

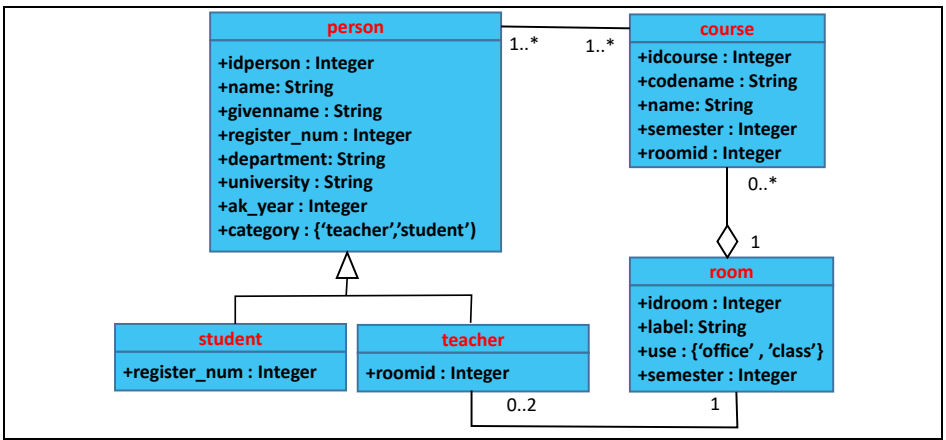

Figure 1: UML of Elementary Student Registry

A3. Modify/enrich the database and extract data according to the queries below.

Q 3.1 Add a **grade**:Integer column to the **personcourse** table and programmatically update it with random integers in the range 5-100, precision 5 units (ie with grades 5, 10, 15, ...).

Q 3.2 (A) Find the average grades of the students in the SDB course separately for the years 2023 and 2024 (a) of those who had not attended the COMP_METHODS course, (b) of those who had attended this latter course (one result) .
(B) Find the grade point averages per home university separately for the years 2023 and 2024 (result table).

Q 3.3 (a) Compile the list of different home universities of the students along with the number of students for each, in descending order of that number.
(b) Insert new office entries into the **room** table, with use category **use='office'** and **label='GI-2024-univ-x'** (**univ**=Institution name, **x**=1,2,... the university group number , e.g. 'GI-2024-ΑΠΘ-1'), which will house the new students (year 2024), up to 2 in each room, as follows:
Start from the university with the most students and give the room with x=1 in the first (alphabetical) pair, the room with x=2 in the second pair and so on. We continue in the same way with the next university, until all students are housed.
(c) Then, add a **roomID** column to the **person** table, as a FOREIGN KEY with reference to **room.IDroom** and insert (automatically, with a query) for each student the reference to the corresponding office.

Q 3.4. Compile a list of student offices with three columns: (a) the label of the office, (b) the home university of the students (c) the number of students it rooms.

A4. Design an **Entities-Relationships (ER)** diagram, equivalent to the UML of Figure 1.

Hint 1: Queries Q3.3 and Q3.4 should be implemented as pl/pgsql functions. For question Q3.4, in particular, you will be helped if in question Q3.3 you simultaneously create a table **roomuniversity**, which connects the office with the home university of the students it houses, as has been presented in the slides of the relevant presentation. Modify the function **more_rooms** of the slides in order to accept as input parameters the **academic year** (eg 2024) and the **maximum number of students** per office (eg 2).

Hint 2 (To access the database through python)
Connecting to a database and making queries can be done programmatically (client programming), eg. with python. The following is a simple example of inserting items (table entries **example1.student**, with elements **name** και **givenname**) which are obtained from a csv file:

```
# Connection to the database
import psycopg2  # http://initd.org/psycopg/docs/sql.html
try:
    conn=psycopg2.connect("dbname='<xxxx>' user='<yyyy>'
              host='147.102.40.25' password='<zzzz>'")
except:
    print("I am unable to connect to the database")
cur = conn.cursor()
```

```
# Get data from file and insert it into the database
import os
import csv
my_dir = '<mydir>'   # the directory where the respective file is located
f_name = ' list_of_students-2023.csv'
file_name = os.path.join(my_dir,f_name)

with open(file_name, encoding="iso-8859-7") as csvfile:
    reader = csv.reader(csvfile, delimiter=';')
    count=0
    for row in reader:
        count=count+1
        if not row[0].isdigit():
            print(row[0])
            header=row
        else:
            print('record: ', count,'   student:  ', row[1])
            cur.execute("""INSERT INTO example1.student (name, givenname)\
                VALUES  ('%s', '%s')""" %(row[1],row[2]))
    cur.execute("COMMIT")
    print(count, ' records done')
```