SD202 “Databases” class: final project

The goal of this lab session is to to design a Web application to manage students and classes in a university. The code for the Web application is already written, but it is missing everything that relates to the database, i.e., the schema definition and queries. Your goal is to design the database schema and complete the queries.

We strongly recommend working on the Télécom machines, as this ensures that the environment for the lab is already installed.

Step 0: Retrieving the files

Retrieve the lab session skeleton from Moodle and decompress it. It consists of several files:

- `README`: readme file for step 1
- `forms.py`: form definitions (do not edit)
- `model.py`: definition of the model (to edit)
- `requirements.txt`: list of required Python modules for step 1
- `rooms.json`: list of Télécom rooms for step 6
- `run.py`: definition of the Web interface (run, but do not edit)
- `static/style.css`: CSS style of the Web interface (do not edit)
- `templates/*.html`: HTML templates (do not edit)

Step 1: installing dependencies

On Télécom machines, run the command:

```
pip3 install flask wtforms
```

On your own machine, you may need to do more setup: please check the `README` file.

Step 2: getting the Web application to work

Open a terminal, navigate to the folder of the Web application, and run:

```
python3 run.py
```

You should get a message of the form:

```
$ ./run.py
 * Serving Flask app "run" (lazy loading)
 * Environment: production
   WARNING: This is a development server.
   Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with watchdog (inotify)
```

```
 * Debugger is active!
```

Leave the terminal open, open a Web browser on your machine, and go to: `http://127.0.0.1:5000/`. Check that the Web interface loads. Of course, at this point, only the main page will work, and other Web pages will give an error.

Step 3: designing the basic schema

In the first step of this project, we want to represent people, rooms, curriculums (sets of courses), and courses. Here are the constraints:

- People have a first name and last name, an address, and a phone number.
- Rooms have a name and a capacity.
- Curriculums have a name, a secretary, and a director (who are persons).
- Courses have a name and a teacher (who is a person).
- Students can be registered in a curriculum.
- Courses can be part of a curriculum, with a number of ECTS indicating their value. The same course can be part of multiple curriculums, possibly with different ECTS values.
- Each course can have an unlimited number of room reservations, with a starting and ending date and time.

Make an entity-relationship diagram and create a database schema to represent this information. Feel free to make any choices you wish for constraints not specified here, e.g., can the teacher of a course be NULL.

Step 4: creating the basic schema

SQLite is a self-contained database engine which stores a database in an individual file. For the project, create a database called `univ.db` by opening a terminal, navigating to the project folder, and running the command:

```
sqlite3 univ.db
```

You can then create the database designed in the previous step by running SQL commands, e.g., `CREATE TABLE`, and creating some test data.

You can also write SQL commands in a file and run all commands at once by issuing the following in the SQLite prompt:

```
.read file.sql
```

Once you are done with SQLite3, run `.exit`.

Hint: if you wish to use foreign keys, you have to enable them by running `PRAGMA foreign_keys = ON` before running any other command.

Step 5: completing the queries

Open the file `model.py` and complete the queries marked `TODO`, from `TODO2` to `TODO15`.

Note that the queries are *prepared queries*, i.e., they take as parameters the values that are provided as a second argument to the `self.cursor.execute` call. These parameters should be used in the query, in the order in which they are given, with the character `?`. The first query (for Q1) can be used as an example.

Whenever you modify the file `model.py`, the Web application should automatically reload, and you can then refresh pages in your Web browser to test.

Once you have completed questions 2 to 15, you should be able to open the tabs "Persons", "Curriculums", "Courses", and "Rooms", you should be able to create and delete every type of objects, and you should be able to go to the details of a room and add room reservations. It is normal for now that going to the detail of other objects will cause an error, as not all queries are completed yet.

Step 6: importing rooms

The list of all classrooms in Télécom is provided as the file `rooms.json`. Import this dataset into your database, so that the tab "Rooms" lists the actual rooms available at Télécom.

To do this, one first option is to write a Python program that imports the JSON (using `import json` and `json.loads`) and runs database queries using SQLite (using `import sqlite3` as in `model.py`).

Alternatively, you can transform the JSON file into SQL commands or a CSV file, e.g., using the `jq` command-line utility.

Step 7: advanced schema

We now consider a more advanced version of the database schema:

- Each course has one or several *examinations*, which have names (e.g., "final exam", "mid-term exam"), a date when they take place, and a coefficient
- Every student following a class (i.e., registered to a curriculum containing the class) may have a *grade* for each examination in the class (if the grade is not known in the database, then by default it is 0).

Extend the database schema with tables to contain this information.

Complete the `models.py` file to write the corresponding queries and finish the Web application.

Acknowledgements

This lab is adapted from material by Louis Jachiet.