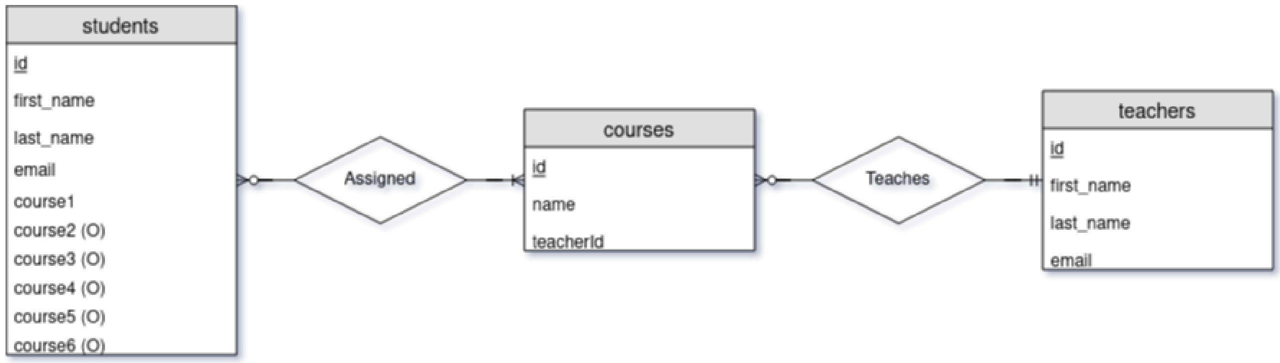
João Paulo Oliveira Cruz

Basics of Database Systems

Final Assignment

I started by thinking about what would be a simple way to lay out the tables to make a light but working database. I also decided to make a python “client” to go with it and manage the database.



The *Students* table has some basic info about the student, and 6 fields for courses. In my model, a student cannot have over 6 courses. All course fields except the first one are optional, meaning a student can be assigned from 1 to 6 courses (mandatory many).

The *Courses* table includes a course id, the course name, and the id of the teacher who teaches the course. A course can only be teached by one teacher (obligatory one). Courses can have from zero to unlimited students (optional many).

The *Teachers* table includes a teacher id, their name and email. A teacher can teach an unlimited number of courses (optional many).

To create all of these tables and populate them, I have created a Python “client”. When you start it up, it checks if the file “database.db” already exists. If it does, it won’t try to create new tables. If it doesn’t, it does the startup sequence of creating the tables and populating them with 4 rows of data each. The program is clearly commented, includes all of the required queries, and can be found at:

https://github.com/jpoCruz/lut-databases/blob/master/basicsOfDatabaseSystems/finalAssignment/assignment.py

Aditionally, all queries can be found on the following pages of this document.

How to insert (students, teachers and courses respectively)

```
sqlite> INSERT INTO students (id, first_name, last_name, email, course1, course2,
course3, course4)
    VALUES (0, 'Joao', 'Cruz', 'joao.cruz@student.lut.fi', 0, 1, 2, 3)
```

When inserting into students, the number of courses can vary from 1 to 6. In this example, I inserted 4.

```
sqlite> INSERT INTO teachers (id, first_name, last_name, email)
    VALUES (0, 'Shola', 'Oyedeji', 'shola.oyedeji@lut.fi')
```

```
sqlite> INSERT INTO courses (id, name, teacherId)
    VALUES (0, 'Basics of Database Systems', 0)
```

How to select

For this example, I decided to show how to select all students that are enrolled on the course with the ID of X.

```
sqlite> SELECT * FROM STUDENTS
   ...> WHERE course1 = X
   ...> OR course2 = X
   ...> OR course3 = X
   ...> OR course4 = X
   ...> OR course5 = X
   ...> OR course6 = X;
```

How to update (based on id)

For this example, I'm updating the 'course1' column. This overrides the current course. X is the new course id, and Y is the student's id.

```
sqlite> UPDATE students
   ...> SET course1 = X
   ...> WHERE id = Y;
```

How to delete (based on id)

For this example, I'm deleting a student.

```
sqlite> DELETE FROM students
   ...> WHERE id = Y;
```

All of the SQLite statements above can be used for the other tables, assuming you change the name of the tables and columns you are messing with.