CS-A1150/CS-A1153 Databases, Summer 2020
Project, Part 1 (UML Model & Relational Data Model)

1. Introduction

The purpose of the project is to learn to design databases in practice. The project consists of two parts. In this part, you are asked to design a database using UML and define the relations based on the diagram. In the second part, you will implement the database (using SQL) you designed in this part.

The project is designed for groups of 2-3 students. **You are not allowed to do your project alone without a group.** Both project parts must be done with same group. You can search for group members in course Slack.

If something in this project description is not clear enough, you can use the course Slack to ask questions. However, all details are not defined, because one of your tasks is to make decisions about the details.

2. Instructions

In this project you will have to design a database for a university. The database contains information about courses, rooms, room reservations and enrollments.

Tasks

- Draw an UML diagram based on the information in the following sections. Use the notation taught in the course.
- Convert the UML diagram to the relational data model. Present the schemas of the relations and underline the attributes which form the key for each relation.
- **Provide answers to the following questions:** What are the functional dependencies of the database? Are there any form of redundancy or other anomalies in the database structure? Is your database in Boyce-Codd Normal Form? If it is not, please use the decomposition algorithm (please submit both original and decomposed version).

3. Detailed Information about Database

The unversity database

- The database contains information about the courses of the university. For each course, the information contains the course code, the name, and the number of credits.
- The same course may be organized several times either in the same semester or in different semesters. Below, we call different implementations of the same course *course instances*.
- A course may have lectures, exercise groups and exams. The lectures and exercise groups belong to a certain course instance, the exams belong to the course.
- The time of the lectures of the course instance may vary i.e. they are not necessarily the same every week.
- The same exercise group may meet several times and the timetable may be different in different weeks. The group may also have several meetings during the same week.
- A course may have several exams (for example, an exam immediately after a course instance and retake exams later).
- The student enrolls for a course by enrolling for one of its exercise groups. We assume that

every course instance has at least one exercise group. An exercise group may have a limit for the number of students and the student cannot enroll for the group which has already reached the limit.
- If a student wants to take an exam, he/she has always enroll for it (it is not enough to enroll for a course).
- For each student, the system stores at least the name, student ID, date of birth, degree program, the year the student started his/her studies and the expiration date of study right.
- The university has several buildings, which contain rooms for teaching purposes (like lecture halls and class rooms). They are called *rooms* below. For each building, the information about its name and street address are stored. The database also contains information about which rooms are located in a certain buiding.
- For each room, the following information is stored: the number of seats, the number of seats in an exam (may be smaller, because the students cannot sit too near each other in the exam), the equipment in the room (like video projector, two video projectors, computer for the teacher, document camera, computers for students etc.) It must be able to store information also about other type of equipment than listed above.
- The database must contain information on the reservations of the rooms. There can been only one exercise group or lecture in a certain room at the same time, but it is possible to have exams of several courses simultaneously in the same room. The reservation for a exam may be longer than the actual exam to give time for practical arrangements.

For example, the following operations must be possible (in addition, your system can support more operations). These operations are not presented in the UML diagram and in the relational model, but you must design them such that the operations are possible:

- Store information about courses, course instances, students, lectures, exercises groups and their times, exams and enrollments.
- Store information about the buildings, rooms and their reservations. The system must also store information about the history of the reservations, for example which exercise group of which course had a reservation for a certain room at a certain time a year ago.
- Search for the courses which are arranged in a certain time interval.
- Find out, which exams a certain course has during a certain time interval.
- Find out, when a certain course has been arranged or when it will be arranged.
- Find the lectures belonging to a certain course instance.
- Find the exercise groups belonging to a certain course instance and find out, when and where a certain exercise group meets.
- Find a room which has at least a certain number of seats and which is free for reservation at a certain time.
- Find out for which purpose a certain room is reserved at a certain time.
- Enroll a certain student for a certain exam or exercise group.
- List all students who have enrolled for a certain course instance, exercise group or exam.
- Find out which exercise groups at a certain course instance are not full yet.

Note that the database does not contain information about the following things, for example:

- The courses the students have completed or their grades.
- Whether a student has enrolled for the university at a certain semester.
- The teachers of the courses.
- More information about the students than told above.
- Degree requirements of the programs, i.e. information about which courses the student has to complete to be able to graduate.

4. Principles of Grading

All members of the group will get the same grade of the project. The project consists of two parts. The parts are worth 20 points each.

- To pass the project, you must get at least 20 points in total and submit the solution to the both parts.
- If you get at least 30 points, your course grade will be increased by one full grade (however, you must pass the exam without this bonus!)

The project grade is valid until the exam in May 2021, but not after that.

You can get up to 20 points for this project part:

- UML modeling 8 p
- Converting the UML diagram to relational data model 4 p
- Functional dependencies, BCNF normal form, etc. 4 p
- The quality of documentation 4 p

5. Submitting the Project

The project part 1 will be submitted to A+-system (Module *Project Part 1*). The link to the system has been published in MyCourses. The deadline of the first round is Aug 7th 2020 at 8:00 p.m. If your submission is late at most 7 days, your submission will be graded, but you will lose 5 points, if the delay is 3-7 days and 3 points, if the delay is under 3 days. If the delay is over 7 days, your project will not be graded except if you have an exceptional reason (for example, a long sick leave signed by a doctor). If you submit your solution already before June 30th at 8:00 p.m., your submission will be graded in the beginning of July and you get feedback by July 12th at the latest. This gives you a possibility to finish your Project Part 2 already in July.

The only acceptable file format is PDF. You may write it in English, Finnish or Swedish.

The report must consist of the following parts:

- Cover page, which includes the following information: Student names and e-mail addresses.
- UML diagram
- Relation schema converted from the UML diagram.
- Explanation of the solution (recommended length 3-4 pages). For example, you can explain how the functions described in Section 3 can be performed in your system and explain unusual solutions.
- Answers (with justifications) to the questions given in Section 2 about normal form, functional dependencies, anomalies and the possible decomposition into Boyce-Codd Normal Form when needed.