SimpleBoun Registration

Project 2, CMPE 321, Introduction to Database Systems, Spring 2022

Due: **April 15, Friday, 23:59**

Ask the project related questions in the Moodle forum!

1 Introduction

As university students, you have all used some university registration systems. These systems allow the instructors and students to manage course registration tasks with minimal effort. Without such systems, we would need to handle everything manually, which clearly would be infeasible. On the other hand, if such a system is erroneous, it might bring more harm than benefit. Thus, a well-organized registration system is vital to ensure the validity and consistency of the actions and data.

There may be many constraints in these systems. For instance, courses might have prerequisites and students might not be allowed to take a course before passing its prerequisite courses. Another example constraint relates to the allocation of classrooms: two lectures may not be held in the same classroom simultaneously. As you can see, if the system fails to check the satisfaction of constraints, the results could be chaotic.

A university registration system is typically concerned with the instructors, students, courses, departments, grades, classrooms etc. There are many logical relationships among the elements of these systems. For instance, an instructor teaches some courses and works in a department; students take courses and are graded by the instructor. As a result, there is a great deal of information in such a system. A database is needed to store and manage the data safely and efficiently.

To understand university registration systems better, you can check Bogazici University Registration System and think about its structure, rules and constraints.

2 Project Description

In this project, you will design a **simplified** university registration database, called ***SimpleBounDB***. You will begin with a detailed description of the content. Then you will need to systematically go through parts of the standard database design process as you learned about in class, including conceptual design, logical design, and schema refinement.

2.1 *SimpleBounDB*: Structure

The *SimpleBounDB* should contain the following information:

1. **User** includes the following attributes; username, password, name, surname, email, Department ID. Each user has a unique username and is associated with exactly one department. In addition, each user is either a student or an instructor.
   (a) **Students** additionally have Student ID attribute and a list of added courses for the current semester. Student ID must be unique.
   (b) **Instructors** additionally have title attribute.
2. **Department** includes the following attributes: Department ID, name. Both name and id must be unique.
3. **Course** includes Course ID, name, Department ID, credits, instructor username, Classroom ID, campus, classroom capacity, time slot, quota and a list of prerequisites. Course ID must be unique.
   - Each course belongs to the department of its instructor. No two courses can overlap in terms of classroom and time slot.
   - In SimpleBoun, there are only 2 time slots for each weekday so there are 10 time slots in total. The time slots are represented with integers from 1 to 10 (e.g. First half of Monday corresponds to 1 and second half of Tuesday corresponds to 3). A course only occupies a single time slot.

- A prerequisite is a course that the student must pass before taking the other course. The Course ID of a prerequisite must be less than the ID of the succeeding course (e.g. MATH101 < MATH102). You may use string comparison to check this.
- Each Classroom ID corresponds to one physical location. Hence, campus and classroom capacity depends on the Classroom ID.

4. **Grades** have following attributes: Student ID, Course ID and grade. A pair of Student ID and Course ID uniquely identifies a grade. Grades will be given as floats (e.g. 3,5).

5. **Database manager** consists of the following attributes: username and password. There exists only one database manager with a certain username. There can be at most 4 database managers registered to the system.

2.2 *SimpleBounDB*: Sample data

username	password	name	surname	email	student_id	department_id	added_courses
berke.argin	newyork123	Berke	Argin	berke.argin@simpleboun.edu.tr	16080	MATH	[CMPE321, IE306]
niyazi.ulke	mypass	Niyazi	Ulke	ulke@simpleboun.edu.tr	17402	CMPE	[CMPE321]
ryan.andrews	pass4321	Ryan	Andrews	andrews@simpleboun.edu.tr	18321	PHIL	[PHIL101]
he.gongmin	passwordpass	He	Gongmin	he.gongmin@simpleboun.edu.tr	19333	IE	[IE306, IE310]
carm.galian	madrid9897	Carmelita	Galiano	carm.galian@simpleboun.edu.tr	19356	PHIL	[PHIL106, MATH102]
kron.helene	helenepass	Helene	Kron	kron.helene@boun.edu.tr	20341	CMPE	[CMPE250]

Table 1: Sample data from Students

student_id	course_id	grade
16080	IE310	3,5
16080	CMPE150	3
16080	CMPE250	4
16080	ENG493	3
17402	CMPE150	4
17402	CMPE250	3,5
17402	PHIL101	4
19333	MATH101	3,5
19333	MATH102	2,5
19356	MATH101	3
19356	PHIL101	3,5
20341	CMPE150	3,5

Table 2: Sample data from Grades

department_id	department_name
CMPE	Computer Engineering
MATH	Mathematics
PHIL	Philosophy
IE	Industrial Engineering

Table 3: Sample data from Departments

username	password
manager1	managerpass1
manager2	managerpass2
manager35	managerpass35

Table 4: Sample data from Database Managers

username	password	name	surname	email	title	department_id
faith.hancock	faithfaith11	Faith	Hancock	hancock@simpleboun.edu.tr	Associate Professor	MATH
rosabel.eerk	eerkens1984	Rosabel	Eerkens	eerk@simpleboun.edu.tr	Assistant Professor	IE
arzucan.ozgur	mypass4321	Arzucan	Ozgur	arzucan.ozgur@simpleboun.edu.tr	Associate Professor	CMPE
simon.hunt	123abc	Simon	Hunt	hunt.simon@simpleboun.edu.tr	Professor	PHIL
sevgi.demir	dmrblk1234	Sevgi	Demirbilek	sevgi.demir1@simpleboun.edu.tr	Professor	MATH
lyuba.boer	easypass12	Lyuba	Boerio	lyub.boerio15@simpleboun.edu.tr	Assistant Professor	PHIL
park.ho	linkinpark	Park	Ho	park.ho@simpleboun.edu.tr	Professor	CMPE
nur.ulku	1nurulku1	Nur	Ulku	ulku@simpleboun.edu.tr	Asistant Professor	CMPE
charles.sutherland	princecharles	Charles	Sutherland	sutherland@simpleboun.edu.tr	Professor	CMPE

Table 5: Sample data from Instructors

course_ID	name	department_id	course_code	credits	instructor_username	classroom_id	campus	classroom_capacity	slot	quota	prerequisites
CMPE150	Introduction to Computing	CMPE	150	3	arzucan.ozgur	HD201	Hisar Campus	300	1	200	
CMPE250	Data Structures and Algorithms	CMPE	250	4	park.ho	BMA2	North Campus	200	2	15	[CMPE150]
CMPE321	Introduction to Database Systems	CMPE	321	4	arzucan.ozgur	BMA2	North Campus	200	3	120	[CMPE250]
CMPE352	Fundamentals of Software Engineering	CMPE	352	4	nur.ulku	BMA3	North Campus	150	3	4	
CMPE451	Project Development in Software Engineering	CMPE	451	4	charles.sutherland	BMA3	North Campus	150	10	120	[CMPE321, CMPE352]
ENG493	Sp. Tp. in Software Engineering	CMPE	493	3	park.ho	BMA2	North Campus	200	4	20	
MATH101	Calculus I	MATH	101	4	faith.hancock	TB310	South Campus	100	1	100	
MATH102	Calculus II	MATH	102	4	sevgi.demir	TB310	South Campus	100	2	5	[MATH101]
IE306	Systems Simulation	IE	306	3	rosabel.eerk	M1171	South Campus	100	3	100	
IE310	Operations Research	IE	310	4	rosabel.eerk	M1171	South Campus	100	7	5	
PHIL101	Introduction to Philosophy	PHIL	101	3	simon.hunt	M1171	South Campus	100	10	5	
PHIL106	Philosophical Texts	PHIL	106	3	lyuba.boer	HD201	Hisar Campus	300	8	105	[PHIL101]

Table 6: Sample data from Courses

2.3 Part 1: Conceptual database design

Your task in Part 1 is to perform the Conceptual Database Design (or ER Design) – draw ER diagrams to capture all the information, following the approach described in lectures. While there are many ER-model variants, for this project, we expect you to use the ER notation from the **textbook and lecture**.

To receive full points for this part, you need to identify all the entity sets and relationship sets in a reasonable way. We expect there to be multiple correct solutions since the ER design is subjective. Your goal should be to reasonably capture the given information. For the entity set names, relationship set names, and attribute names that you will be using in your ER diagram, you can use the ones we have provided in Section 2.1 and Section 2.2. You can use underscores, spaces, numbers, uppercase, or lowercase letters to construct those names. It is required to use the features of ER modeling that you have learned from the lectures, including participation constraints, key constraints, weak entities, class hierarchy, and aggregation. We provide you concrete sample data because it may help you understand the problem better. You should use computer-based / online drawing tools, handcrafted diagrams will not be accepted.

2.4 Part 2: Logical database design

For the second part of the project, your task is to convert the ER diagrams into relational tables, based on the set of simple rules as described in the textbook and in lectures. You should provide the schema of each relation including the relation name, attribute names, and attribute domains.

2.5 Part 3: Schema refinement and normalization

For the third part of the project, your task is to analyze your design in Part 2 in terms of functional dependencies (FDs) and normal forms, then, refine your design if needed. You should explicitly list all of the non-trivial FDs. Then, for each relation you should determine if it is in **Boyce-Codd Normal Form** (BCNF) and you should explain how the requirements of BCNF are met (or not met) in terms of FDs. If a relation is not in BCNF, you should check whether it is 3NF and explain how the requirements for 3NF are met (or not met). If a relation is not in BCNF, you should either decompose it into BCNF relations or provide a justification if you decide not to decompose it. If you decompose a relation, you should explain whether the decomposition is lossless-join and dependency preserving. It is possible that your initial schema is already in BCNF. If this is the case, you still need to explain how the requirements of BCNF are met in terms of FDs for each one of your relations.

2.6 Part 4: Write SQL statements for the normalized schema

You are required to write SQL DDL statements that create the tables you designed for this part. You should specify all the constraints such as PK, FK, Unique, NOT NULL, and other general constraints with CHECK. You should turn in two files:

1. createTables.sql
2. dropTables.sql

Make sure that you include a comment in each file.
Note: For this part, you must use a relational database that supports servers. SQLite is not allowed.

3 Submission & Remarks

This project can be implemented either individually or as a team of two people. You are free to change teams in the upcoming projects. Place all `.sql` files and a PDF file contains the outputs of Part 1, Part 2, and Part 3 into a folder named with the student IDs of the team members separated by an underscore (e.g. `2017400200_2018700120`). Zip the folder for submission and name the `.zip` file with the same name. Submit the `.zip` file through Moodle until the deadline. **Any other submission method and late submissions are not allowed**.