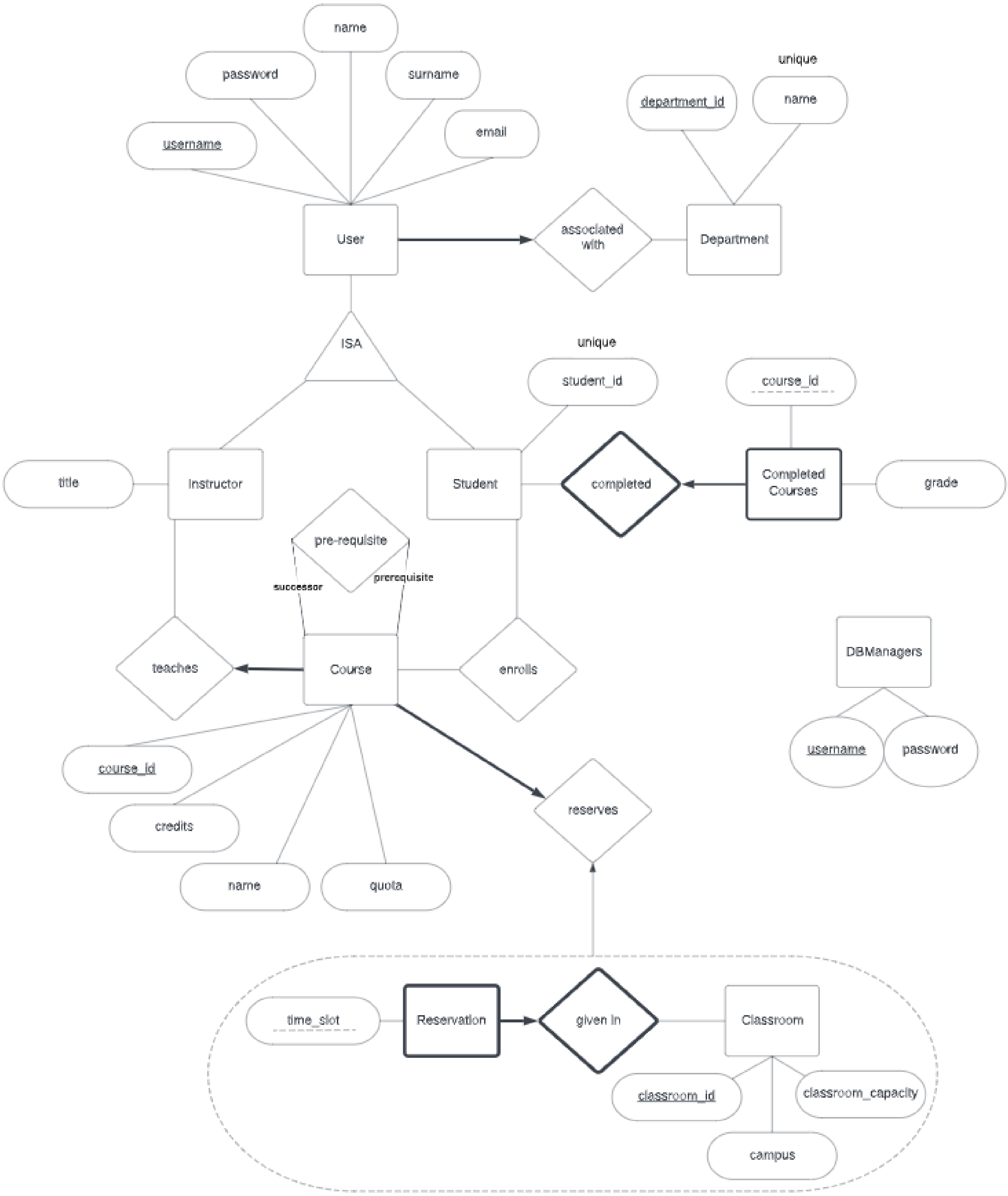
SimpleBOUN Registration
Project 2, CMPE 321, Introduction to Database Systems, Spring 2022

Karahan Sarıtaş (2018400174)
Cahid Arda Öz (2019400132)

Part 1: Conceptual Database Design



Part 2: Logical Database Design

User		
PK	username	Char(100)
	password	Char(100)
FK	department_id	Char(100)
	name	Char(100)
	surname	Char(100)
	email	Char(100)

Department		
PK	department_id	Char(100)
Unique	department_name	Char(100)

Student		
PK, FK	username	Char(100)
Unique	student_id	Integer

Instructor		
PK, FK	username	Char(100)
	title	Char(100)

Completed_course		
PK, FK	student_id	Integer
PK	course_id	Char(100)
	grade	Real

Teaches		
FK	username	Char(100)
PK, FK	course_id	Char(100)

Enrolls		
PK, FK	student_id	Integer
PK, FK	course_id	Char(100)

Course		
PK	course_id	Char(100)
	name	Char(100)
	quota	Integer
	credits	Integer

Prerequisite		
PK, FK	successor_id	Char(100)
PK, FK	prerequisite_id	Char(100)

Reserves		
FK, Unique	course_id	Char(100)
PK, FK	classroom_id	Char(100)
PK	time_slot	Integer

Classroom		
PK	classroom_id	Char(100)
	campus	Char(100)
	classroom_capacity	Integer

DBManagers		
PK	username	Char(100)
	password	Char(100)

Part 3: Schema Refinement and Normalisation

Let R be a relation schema, X be a subset of attributes of R, and let A be an attribute of R. R is in Boyce-Codd normal form if for every FD X → A that holds over R, one of the following statements is true:

- A is an element of X; that is, it is a trivial FD (1)
- X is a superkey. (2)

Let's apply this rule to explain how the requirements of BCNF are met (or not met) in terms of FDs for each relation.

- A **super key** is a set of one or more columns that can be used to identify a record uniquely in a table.
- **Candidate key** is a minimal super key, meaning that any proper subset of a candidate key cannot be a super key.
- **Primary key** is a column or a combination of columns that uniquely identifies a record. Only one candidate key can be the primary key in a relation.

By definition, both a candidate key and a primary key is also a super key since they uniquely identify a record in the table.

User

- $username \rightarrow username, password,\ department\_id,\ name,\ surname,\ email$

$\{username\}$ is the primary key, therefore it complies with the condition (2).

Department

- $department\_id \rightarrow department\_id,\ department\_name$

$\{department\_id\}$ is the primary key, therefore it complies with the condition (2).

- $department\_name \rightarrow department\_id$

Since $department\_id$ is unique, there is a functional dependency that doesn't comply with the conditions. However, we didn't split the tables at this point as it would make our design unnecessarily more complicated.

Student

- $username \rightarrow username,\ student\_id$

$Student$ relation extends the $User$ relation. $\{username\}$ is the primary key, therefore it complies with the condition (2).

- $student\_id \rightarrow username$

Since $student\_id$ is unique, there is a functional dependency that doesn't comply with the conditions. However, we didn't split the tables at this point as it would make our design unnecessarily more complicated.

Instructor

- $username \rightarrow username,\ title$

$Student$ relation extends the $Instructor$ relation. $\{username\}$ is the primary key, therefore it complies with the condition (2).

Completed_Course

- $student\_id,\ course\_id \rightarrow student\_id,\ course\_id,\ grade$

$\{student\_id,\ course\_id\}$ is the composite primary key, therefore it complies with the condition (2).

Teaches

- $course\_id \rightarrow username,\ course\_id$

$\{course\_id\}$ is the primary key, therefore it complies with the condition (2).

Enrolls

- $student\_id,\ course\_id \rightarrow student\_id,\ course\_id$ (trivial FD)

$student\ id,\ course\_id \rightarrow student\_id,\ course\_id$ is in the form of $AB \rightarrow AB$, therefore it complies with the condition (1).

Course

- $course\_id \rightarrow course\_id,\ name,\ quota,\ credits$

$\{course\_id\}$ is the primary key, therefore it complies with the condition (2).

Prerequisite

- $successor\_id,\ prerequisite\_id \rightarrow successor\_id,\ prerequisite\_id$ (trivial FD)

$successor\_id,\ prerequisite\_id \rightarrow successor\_id,\ prerequisite\_id$ is in the form of AB → AB, therefore it complies with the condition (1).

Reserves

- $classroom\_id,\ time\_slot \rightarrow course\_id,\ classroom\_id,\ time\_slot$
- $course\_id \rightarrow course\_id,\ classroom\_id,\ time\_slot$

$classroom\_id,\ time\_slot \rightarrow course\_id,\ classroom\_id,\ time\_slot \Rightarrow \{classroom\_id,\ time\_slot\}$ is the composite primary key, therefore it complies with the condition (1).

$course\_id \rightarrow course\_id,\ classroom\_id,\ time\_slot \Rightarrow \{course\_id\}$ uniquely identifies a *Reserves* relation since each course is associated with one and only one $\{classroom\_id,\ time\_slot\}$ pairs. Therefore it is a superkey, and it complies with the condition (1).

Classroom

- $classroom\_id \rightarrow classroom\_id,\ campus,\ classroom\_capacity$

$\{classroom\_id\}$ is the primary key, therefore it complies with the condition (2).

DBManagers

- $username \rightarrow username,\ password$

$\{username\}$ is the primary key, therefore it complies with the condition (2).

To conclude, all of our relations are in the Boyce-Codd Normal Form.

Part 4: Write SQL Statements for the Normalized Schema

We have already provided the createTables.sql and dropTables scripts as separate files in our submission. Here, we will explain some of the constraints we didn't cover, mainly because these constraints required triggers. TA advised on the forum that we don't add the triggers yet. We were asked to explain why those constraints required triggers instead of implementing them.

We list the two constraints we didn't cover and explain why we didn't cover them with the tables we generate:

- *Only 4 database managers are allowed.* We looked for ways of enforcing this constraint when creating the DBManagers table but we couldn't find a way of doing this without using triggers.
- *When two courses are in a prerequisite relation, the numerical code of the successor is higher than the prerequisite course.* Unfortunately we can't use the `CHECK` query. Apparently, MySQL doesn't allow using `CHECK` for foreign keys with `ON UPDATE CASCADE ON DELETE CASCADE`. Related source. If we remove that part, then we can't delete any course (which is a prerequisite or successor) from the database, it gives an error due to the foreign key checks. However, we can solve the problem using triggers as stated in the given source.