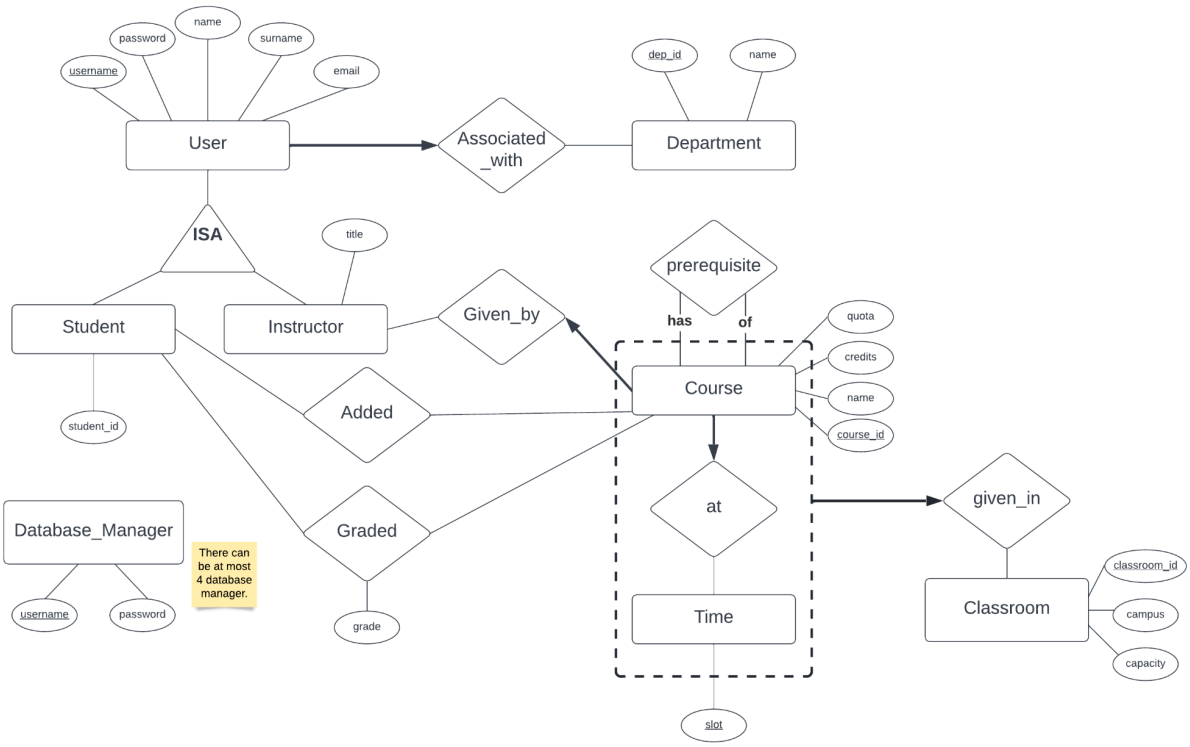
Spring 2022 CMPE 321 Project 2

Berat Damar (2018400039) - Halil Burak Pala (2019400282)

Part 1- Our ER Design is as follows:



(You can see a more detailed version here.)

- **User** entity set has 5 attributes: *username*, *password*, *name*, *surname* and *email*. *username* is the key for User. There are two types of User: Students and instructors. So, we created two entity sets: **Student** and **Instructor**. They are connected to **User** with a ISA relationship. Student has *student_id* attribute, Instructor has *title* attribute in addition to attributes that come from **User**.

- Each User is associated with a department. So, we created **Department** entity set which has *dep_id* and name attributes and then we created ***Associated_with*** relationship set between **User** and **Department**. From **User** to ***Associated_with*** relationship set, there are key constraint and total participation since every user is associated with one and only one department. From **Department** to ***Associated_with***, there is partial participation since a department can be related to

zero or more user.

- **Course** entity set has attributes *quota*, *credits*, *name*, and *course_id*, and course_id is the key. We also created **Time** entity set which has a single key attribute *slot*, and created **Classroom** entity set which has three attributes, *classroom_id*, *campus*, *capacity* and *classroom_id* is the key.

- Between Course and Time, we created ***at*** relationship set. From **Course** to ***at***, there are key constraint and total participation since every course is given in a single time slot. From **Time** to ***at***, there is partial participation since a time slot can be related to zero or more courses. Here, we created an aggregation that covers **Course**, ***at*** and **Time**. We created a relationship set ***given_in***. From this aggregation to ***given_in***, there is a key constraint and total participation since every course that is given at a single time slot can be given in only one classroom physically.

- We created a ***prerequisite*** relationship to show the ***Course*** and its prerequisite ***Course***. It is a relationship between the same type of entity set(***Course***). We connected the relationship and entity set in that way(shown in figure) since a **Course** has zero or more prerequisite and a **Course** can be a prerequisite of zero or more courses.

- We created ***Graded*** relationship between **Student** and **Course**. From **Student** to ***Graded*** there is partial participation since a student need not have a grade in a course. From **Course** to ***Graded***, there is partial participation since a course need not have a graded student. (For example, a newly created course does not have any graded students.)

- We created ***Added*** relationship between **Student** and **Course** to show courses that are added by a student. In this relationship, a student is able to add zero or more courses to him/her course list.A course can be added by zero or more students.

- We created a **Database_Manager** entity set for database managers. It has two attributes: *username* and *password*. *username* is the key for **Database_Manager**.

Part 2- We have 9 tables. Their schemas are as follows:

1) Student(username:string, student_id:integer, password:string, name:string, surname:string, email:string, dep_id:integer)

```
CREATE TABLE IF NOT EXISTS Student (
    username VARCHAR(20),
    student_id INT,
    password VARCHAR(20),
    name VARCHAR(20),
    surname VARCHAR(20),
    email VARCHAR(20),
    dep_id INT NOT NULL,
    PRIMARY KEY(username),
    UNIQUE(student_id),
    FOREIGN KEY(dep_id) REFERENCES Department(dep_id) ON DELETE CASCADE
)
```

2) Instructor(username:string, title:string, password:string,name:string,surname:string,email:string, dep_id:integer)

```
CREATE TABLE IF NOT EXISTS Instructor (
    username VARCHAR(20),
    title VARCHAR(20),
    password VARCHAR(20),
    name VARCHAR(20),
    surname VARCHAR(20),
    email VARCHAR(20),
    dep_id INT NOT NULL,
    PRIMARY KEY(username),
    FOREIGN KEY(dep_id) REFERENCES Department(dep_id) ON DELETE CASCADE
)
```

3) Department(dep_id:integer, name:string)

```
CREATE TABLE IF NOT EXISTS Department(
    dep_id INT,
    name VARCHAR(20),
    PRIMARY KEY(dep_id),
    UNIQUE(name)
)
```

4) Classroom(classroom_id:integer, campus:string, capacity:integer)

```
CREATE TABLE IF NOT EXISTS Classroom (
    classroom_id INT,
    campus VARCHAR(20),
    capacity INT,
    PRIMARY KEY(classroom_id)
)
```

5) Course(course_id:integer, name:string, credits: integer, quota:integer, classroom_id:integer, time_slot:integer, instructor_username:string)

```
CREATE TABLE IF NOT EXISTS Course (
    course_id INT,
    name VARCHAR(20),
    credits INT,
```

```
    quota INT,
    classroom_id INT,
    time_slot INT,
    instructor_username VARCHAR(20) NOT NULL,
    UNIQUE(classroom_id, time_slot),
    UNIQUE(time_slot, instructor_username),
    PRIMARY KEY(course_id),
    FOREIGN KEY(classroom_id) REFERENCES Classroom(classroom_id) ON DELETE
    CASCADE,
    FOREIGN KEY(instructor_username) REFERENCES Instructor(username) ON DELETE
    CASCADE,
    CHECK (1<=time_slot<=10)
)
```

6)Prerequisite(course_id:integer,pre_course_id:integer)

```
CREATE TABLE IF NOT EXISTS Prerequisite (
    course_id INT,
    pre_course_id INT,
    PRIMARY KEY(course_id, pre_course_id),
    FOREIGN KEY(course_id) REFERENCES Course(course_id) ON DELETE CASCADE,
    FOREIGN KEY(pre_course_id) REFERENCES Course(course_id) ON DELETE
    CASCADE,
    CHECK (STRCMP(course_id,pre_course_id)=1)
)
```

7) Student_Grade(student_id:integer, course_id:integer, grade:real)

```
CREATE TABLE IF NOT EXISTS Student_Grade(
    student_id INT,
    course_id INT,
    grade REAL,
    PRIMARY KEY(student_id,course_id),
    FOREIGN KEY(student_id) REFERENCES Student(student_id) ON DELETE CASCADE,
    FOREIGN KEY(course_id) REFERENCES Course(course_id) ON DELETE CASCADE
)
```

8) Course_Added(student_id:integer, course_id:integer)

```
CREATE TABLE IF NOT EXISTS Course_Added(
    student_id INT,
    course_id INT,
    PRIMARY KEY(student_id, course_id),
    FOREIGN KEY(student_id) REFERENCES Student(student_id) ON DELETE CASCADE,
    FOREIGN KEY(course_id) REFERENCES Course(course_id) ON DELETE CASCADE
)
```

9) Database_Manager(username:string, password:string)

```
CREATE TABLE IF NOT EXISTS Database_Manager(
    username VARCHAR(20),
    password VARCHAR(20),
    PRIMARY KEY(username)
)
```

Part 3- We check every table for normality:

1) **Student(username:string, student_id:integer, password:string, name:string, surname:string, email:string, dep_id:integer):**

   Set of Attributes: {U,St,P,N,Su,E,D}
   Functional Dependencies:
   U → UStPNSuED
   St → UStPNSuED

   In this relation, there are two dependencies. Since username and student_id is unique for every student, we have two key constraints. Since only constraints that we have are key constraints, this table is in BCNF form.

2) **Instructor(username:string, title:string, password:string, name:string, surname:string, email:string, dep_id:integer)**
   Set of Attributes: {U,T,P,N,S,E,D}
   Functional Dependencies:
   U → UTPNSED

   In this relation, there is only one dependency. Since username is unique for every instructor, we have one key constraint only. Since the only constraint that we have is a key constraint, this table is in BCNF form.

3) **Department(dep_id:integer, name:string)**
   Set of Attributes: {D,N}
   Functional Dependencies:
   D → DN
   N → DN
   In this relation, there are two dependencies. Since dep_id and name both are unique for every department, we have two key constraints. Since only constraints that we have are a key constraint, this table is in BCNF form.

4) **Classroom(classroom_id:integer, campus:string, capacity:integer)**
   Set of Attributes: {Cl,Cam,Cap}
   Functional Dependencies:
   Cl → ClCamCap

   In this relation, there is only one dependency. Since clasroom_id is unique for every classroom, we have one key constraint only. Since the only constraint that we have is a key constraint, this table is in BCNF form.

5) **Course(course_id:integer, name:string, credits: integer, quota:integer, classroom_id:integer, time_slot:integer, instructor_username:string)**
Set of Attributes: {Co,N,Cr,Q,Ci,T,I}
Functional Dependencies:
Co → CoNCrQCiTI
CiT → CoNCrQCiTI
TI → CoNCrQCiTI

In this relation, there are three dependencies. Since course_id, (classroom_id, time_slot) pair and (time_slot,instructor_username) pair are unique for every course, we have three key constraints. Since only constraints that we have are key constraints, this table is in BCNF form.

6) **Prerequisite(course_id:integer, pre_course_id:integer)**
Set of Attributes: {C,P}
Functional Dependencies:
CP → CP

In this relation, there is only one dependency. Since (course_id, pre_course_id) pair is unique for every prerequisite, we have one key constraint only. Since the only constraint that we have is a key constraint, this table is in BCNF form.

7) **Student_Grade(student_id:integer, course_id:integer, grade:real)**
Set of Attributes: {S,C,G}
Functional Dependencies:
SC → SCG

In this relation, there is only one dependency. Since (student_id, course_id) pair is unique for every grade, we have one key constraint only. Since the only constraint that we have is a key constraint, this table is in BCNF form.

8) **Course_Added(student_id:integer, course_id:integer)**
Set of Attributes: {S,C}
Functional Dependencies:
SC → SC

In this relation, there is only one dependency. Since (student_id, course_id) pair is unique for every added course, we have one key constraint only. Since the only constraint that we have is a key constraint, this table is in BCNF form.

9) **Database_Manager(username:string, password:string)**
Set of Attributes: {U,P}
Functional Dependencies:

$U \rightarrow UP$

In this relation, there is only one dependency. Since username is unique for every database manager, we have one key constraint only. Since the only constraint that we have is a key constraint, this table is in BCNF form.