CO2102 Assignment 1 - University Database Design and SQL Query
Group 91

TASK 1

Department (department_name(PK), building, budget)

Instructor (instructor_id(PK), name, salary, department_name(FK), advisor_id(FK))

Student (student_id(PK), name, tot_cred, advisor_id(FK), department_name(FK))

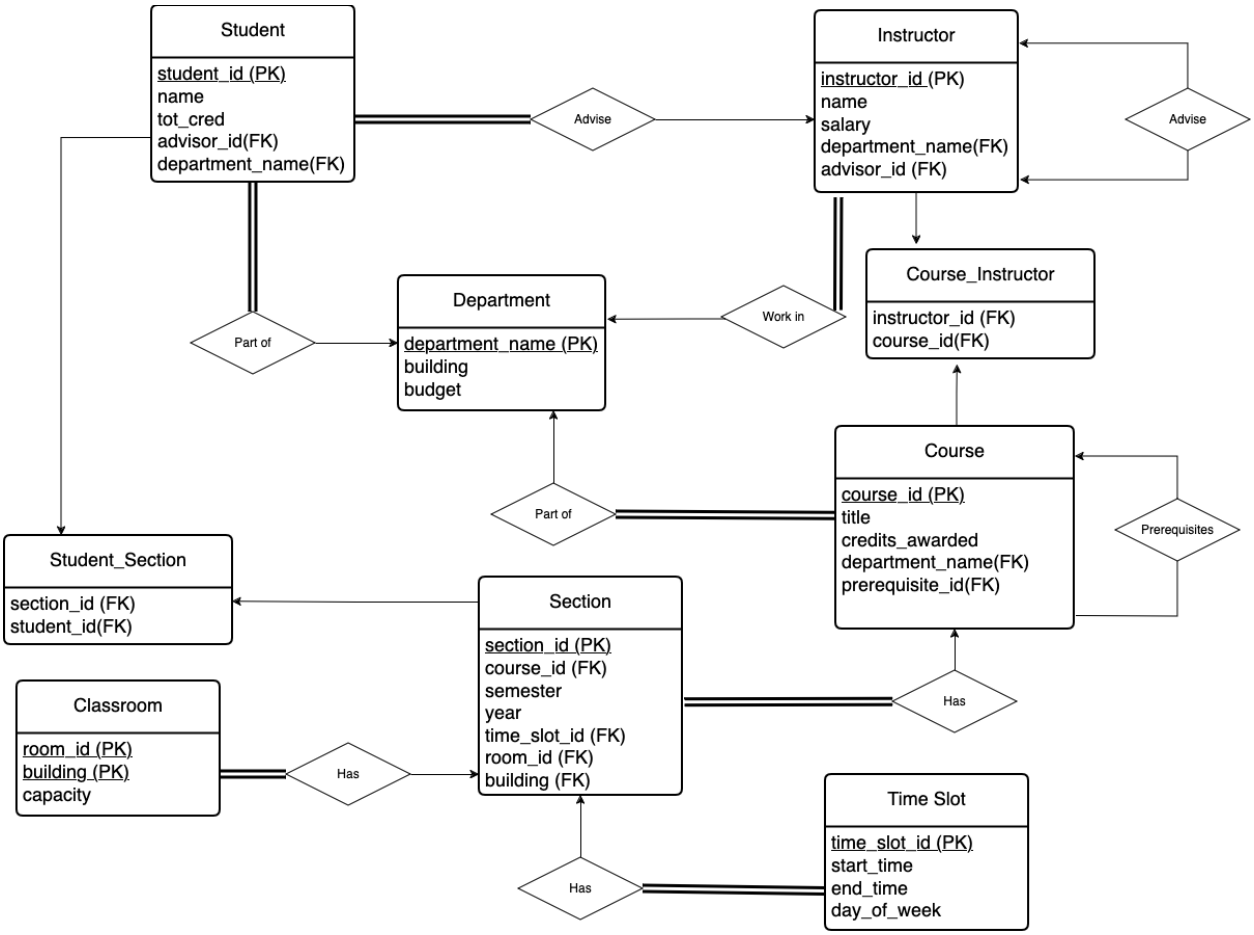
Course (course_id(PK),prerequisite_id(PK), title, credits_awarded, department_name(FK))

Section (section_id(PK), course_id(FK), semester, year, time_slot_id(FK), room_id(FK), building(FK))

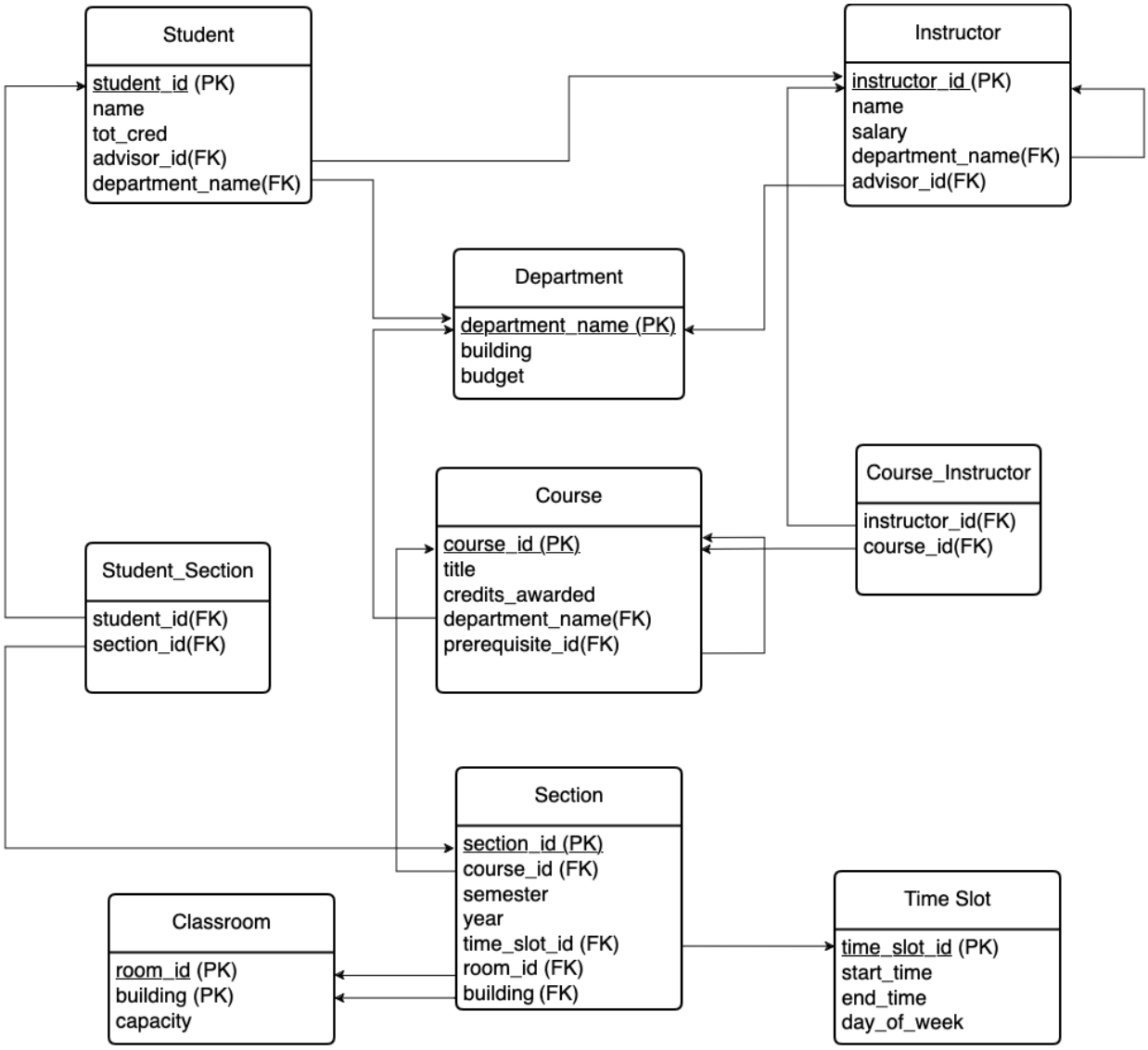
Time Slot (time_slot_id(PK), start_time, end_time, day_of_week)

Classroom (room_id(PK), building(PK), capacity)

TASK 2


Student
student_id (PK)
name
tot_cred
advisor_id(FK)
department_name(FK)
Advise
Instructor
instructor_id (PK)
name
salary
department_name(FK)
advisor_id (FK)
Advise
Course_Instructor
instructor_id (FK)
course_id(FK)
Work in
Department
department_name (PK)
building
budget
Part of
Course
course_id (PK)
title
credits_awarded
department_name(FK)
prerequisite_id(FK)
Prerequisites
Part of
Student_Section
section_id (FK)
student_id(FK)
Section
section_id (PK)
course_id (FK)
semester
year
time_slot_id (FK)
room_id (FK)
building (FK)
Has
Classroom
room_id (PK)
building (PK)
capacity
Has
Time Slot
time_slot_id (PK)
start_time
end_time
day_of_week
Has

TASK 3.1


Student
student_id (PK)
name
tot_cred
advisor_id(FK)
department_name(FK)
Instructor
instructor_id (PK)
name
salary
department_name(FK)
advisor_id(FK)
Department
department_name (PK)
building
budget
Course_Instructor
instructor_id(FK)
course_id(FK)
Course
course_id (PK)
title
credits_awarded
department_name(FK)
prerequisite_id(FK)
Student_Section
student_id(FK)
section_id(FK)
Section
section_id (PK)
course_id (FK)
semester
year
time_slot_id (FK)
room_id (FK)
building (FK)
Classroom
room_id (PK)
building (PK)
capacity
Time Slot
time_slot_id (PK)
start_time
end_time
day_of_week

TASK 3.2

First Normal Form (1NF):

The ER Diagram follows and achieves 1NF. The reason is because based on the Student entity table we can see that the attributes are as follows:

Student (student_id(PK), name, tot_cred, advisor_id(FK), department_name(FK))

For each of the attributes there can only be a single value for example, name is a single string and it is not a list. Likewise, tot_cred can only be a single value and not a list of values for each student. Therefore it follows 1NF as there are no multivalued attributes or repeating groups.

Second Normal Form (2NF):

The ER Diagram follows and achieves 2NF. The reason is because it follows 1NF and also based on the Course entity table we can see that the following attributes are as follows:

Course (course_id(PK), title, credits_awarded, department_name(FK))

From this we can see that “title”, “credits_awarded” and “department_name” all depend on the course_id therefore it follows 2NF as there is full dependency on the primary key.

Third Normal Form (3NF):

The ER Diagram follows and achieves 3NF. The reason is because it follows 1NF and 2NF and also based on the Instructor entity table we can see that the following attributes are as follows:

Section (instructor_id(PK), name, salary, department_name(FK), advisor_id(FK))

From this we can see that the instructor_id is the primary key which will identify each of the records in the section entity table. The attributes “name”, “salary”, “department_name” and “advisor_id” only depend directly on instructor_id. They do not depend on any other attributes in the Instructor entity table.

In this way we maintain and follow 3NF due to eliminating any risk of transitive dependencies and all non-key attributes depend directly on the primary key which is instructor_id.

TASK 4

```
CREATE DATABASE University;

USE University;

CREATE TABLE Department (
        department_name VARCHAR(50) PRIMARY KEY,
        building VARCHAR(50),
        budget DECIMAL(15, 2)
);

CREATE TABLE Instructor (
        instructor_id INT PRIMARY KEY,
        name VARCHAR(50) NOT NULL,
        salary DECIMAL(10, 2),
        department_name VARCHAR (50),
        advisor_id INT,
        FOREIGN KEY (department_name) REFERENCES Department(department_name),
        FOREIGN KEY (advisor_id) REFERENCES Instructor(instructor_id)
);

CREATE TABLE Classroom (
        room_id INT,
        building VARCHAR(50),
        capacity INT,
        PRIMARY KEY (room_id, building)
);

CREATE TABLE Time_Slot (
        time_slot_id INT PRIMARY KEY,
        start_time TIME,
        end_time TIME,
        day_of_week VARCHAR (20)
);

CREATE TABLE Course (
        course_id INT PRIMARY KEY,
        title VARCHAR (50),
        credits_awarded INT,
        department_name VARCHAR(50),
        prerequisite_id INT,
        FOREIGN KEY (department_name) REFERENCES Department(department_name),
        FOREIGN KEY (prerequisite_id) REFERENCES Course(course_id)
);
```

```
CREATE TABLE Student (
        student_id INT PRIMARY KEY,
        name VARCHAR (50) NOT NULL,
        tot_cred INT,
        advisor_id INT,
        department_name VARCHAR(50),
        FOREIGN KEY (department_name) REFERENCES Department(department_name),
        FOREIGN KEY (advisor_id) REFERENCES Instructor(instructor_id)
);

CREATE TABLE Section (
        section_id INT PRIMARY KEY,
        course_id INT,
        semester VARCHAR(20),
        year INT,
        time_slot_id INT,
        room_id INT,
        building VARCHAR(50),
        FOREIGN KEY (course_id) REFERENCES Course(course_id),
        FOREIGN KEY (time_slot_id) REFERENCES Time_Slot(time_slot_id),
        FOREIGN KEY (room_id, building) REFERENCES Classroom(room_id, building)
);

CREATE TABLE Course_instructor (
        instructor_id INT,
        course_id INT,
        FOREIGN KEY (instructor_id) REFERENCES Instructor(instructor_id),
        FOREIGN KEY (course_id) REFERENCES Course(course_id)
);

CREATE TABLE Student_section (
        student_id INT,
        section_id INT,
        FOREIGN KEY (student_id) REFERENCES Student(student_id),
        FOREIGN KEY (section_id) REFERENCES Section(section_id)
);
```

TASK 5

Question 1:

```
SELECT Instructor.name, Course_Instructor.course_id
FROM Instructor
INNER JOIN Course_Instructor on Instructor.instructor_id = Course_instructor.instructor_id;
```

Question 2:

```
SELECT Instructor.name, Course.title
FROM Instructor
NATURAL JOIN Course_Instructor
NATURAL JOIN Course;
```

Question 3:

```
SELECT Instructor.name
FROM Instructor
WHERE Instructor.salary >= 90000 AND Instructor.salary <= 100000;
```

Question 4:

```
SELECT DISTINCT Course.course_id, Course.title
FROM Course
INNER JOIN Section ON Course.course_id = Section.course_id
WHERE (Section.semester="Fall" AND Section.year=2009) OR (Section.semester="Spring" AND
Section.year=2010);
```

Question 5:

```
SELECT Department.department_name, COUNT(DISTINCT Instructor.instructor_id) AS
number_of_instructors
FROM Instructor
INNER JOIN Course_instructor ON Instructor.instructor_id = Course_instructor.instructor_id
INNER JOIN Course ON Course_instructor.course_id = Course.course_id
INNER JOIN Section ON Course.course_id = Section.course_id
INNER JOIN Department ON Instructor.department_name = Department.department_name
WHERE Section.semester = 'Spring' AND Section.year = 2010
GROUP BY Department.department_name;
```

Question 6:

```
SELECT Section.section_id,
AVG(Student.tot_cred) AS tot_cred
FROM Section
INNER JOIN Student_Section ON Section.section_id = Student_Section.section_id
INNER JOIN Student ON Student_Section.student_id = Student.student_id
WHERE Section.year = 2009
GROUP BY Section.section_id
HAVING COUNT(Student.student_id) >= 2;
```

Question 7:

```
SELECT Course.course_id, Course.title
FROM Course
INNER JOIN Section ON Course.course_id = Section.course_id
WHERE Section.semester = 'Fall' AND Section.year = 2009
AND Course.course_id NOT IN (
        SELECT Section.course_id
        FROM Section
        WHERE Section.semester = 'Spring' AND Section.year = 2010
);
```

Question 8:

```
SELECT Department.department_name
FROM Department
INNER JOIN Instructor ON Department.department_name = Instructor.department_name
GROUP BY Department.department_name
HAVING AVG(Instructor.salary) >= ALL (
        SELECT AVG(Instructor.salary)
        FROM Instructor
        GROUP BY Instructor.department_name
);
```

Question 9:

```
UPDATE Student
SET tot_cred = (
        SELECT SUM(Course.credits_awarded)
        FROM Student_Section
        INNER JOIN Section ON Student_Section.section_id = Section.section_id
        INNER JOIN Course ON Section.course_id = Course.course_id
        WHERE Student_Section.student_id = Student.student_id
);
```

Question 10:

```
SELECT Student.student_id, Student.name, Section.section_id, Course.title, Section.semester,
Section.year
FROM Section
LEFT JOIN Student_Section ON Section.section_id = Student_Section.section_id
LEFT JOIN Student ON Student_Section.student_id = Student.student_id AND
Student.department_name = 'Computer Science'
INNER JOIN Course ON Section.course_id = Course.course_id
WHERE Section.semester = 'Spring' AND Section.year = 2009;
```

Task 6

Daniel Anisoreac (Student ID: daa22) contributed 33.3% of the project. I provided solutions and ideas for all tasks and worked with the group on each of them.

Jayesh Patel (Student ID: jp644) contributed 33.3% of the project. I provided solutions and ideas for all tasks and worked with the group on each of them.

Sushant Jasra Kumar (Student ID: sjk50) contributed 33.3% of the project. I provided solutions and ideas for all tasks and worked with the group on each of them.