Task 1.1

Relation A

1) Superkeys:

1. {EmpID}
2. {SSN}
3. {Email}
4. {EmpID, SSN}
5. {SSN, Email, Phone}
6. {Email, Name, Department}

2) Candidate keys:

1. {EmpID}
2. {SSN}
3. {Email}

3) EmpID because of its stability and simplicity. It won't change ever and it is effective to use it as foreign key.

4) Yes, they can.

Relation B

1) {StudentID, CourseCode, Semester, Year}

2) StudentID is necessary to identify student, CourseCode to identify the course. Since a student can take the same course at different times, we need time attributes(Semester, Year) to distinguish the entries.

3) {StudentID, CourseCode, Section, Semester, Year}

Task 1.2

Student(AdvisorID) – Professor(ProfID)

Department(ChairID) – Professor(ProfID)

Course(DepartmentCode) – Department(DeptCode)

Enrollment(StudentID) – Student(StudentID)

Enrollment(CourseID) – Course(CourseID)

Task 2.1

1) Strong Entities: Patient, Doctor, Department, Medication

Weak Entities: Appointment, Prescription, Hospital Room

2) Patient: Patient_ID (Simple, PK), Name (Simple), Birthdate (Simple), Address (Composite – {Street, City, State, ZIP}), Phone Number (Multi-Valued), Insurance_Info (Simple)

Doctor: Doctor_ID(Simple, PK), Name(Simple), Specialization(Multi-Valued), Phone Number(Multi-Valued), Office_Location(Simple)

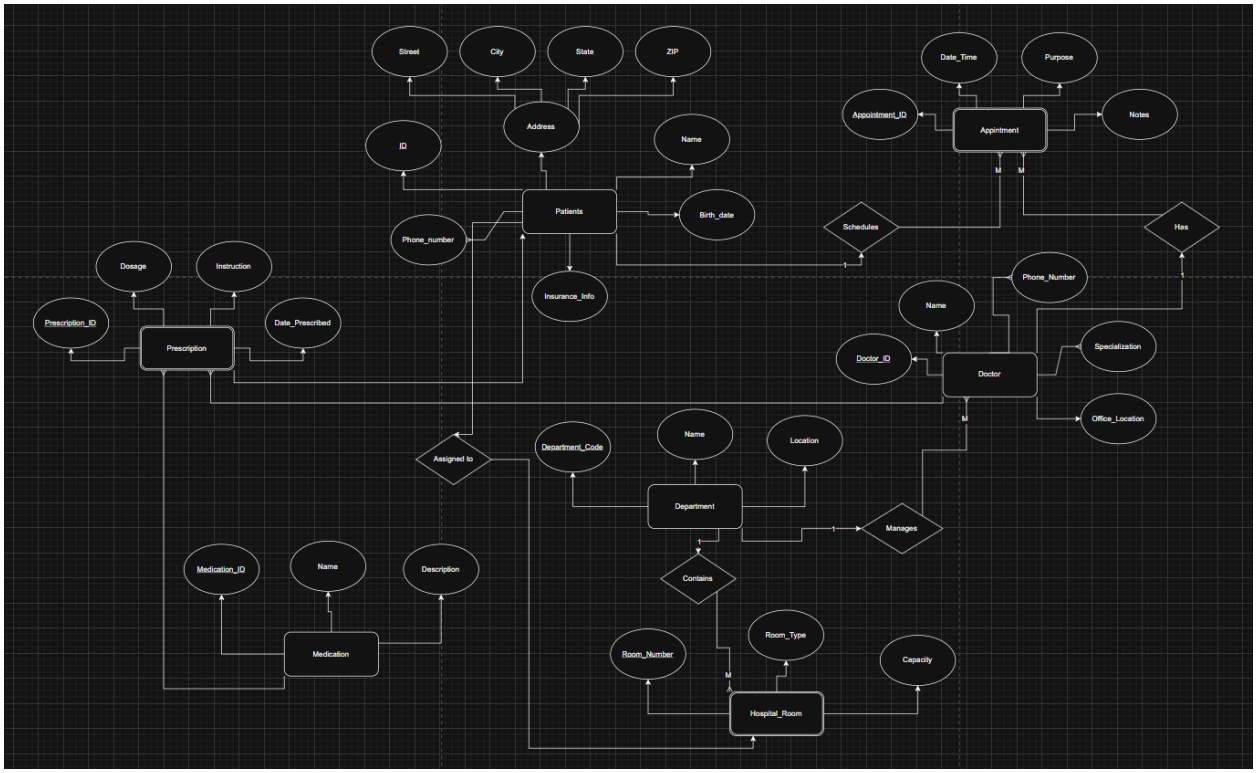
Department: DeptCode(Simple, PK), DeptName(Simple), Location(Simple)

Medication: MedicationID(Simple, PK), Name(Simple), Description(Simple)

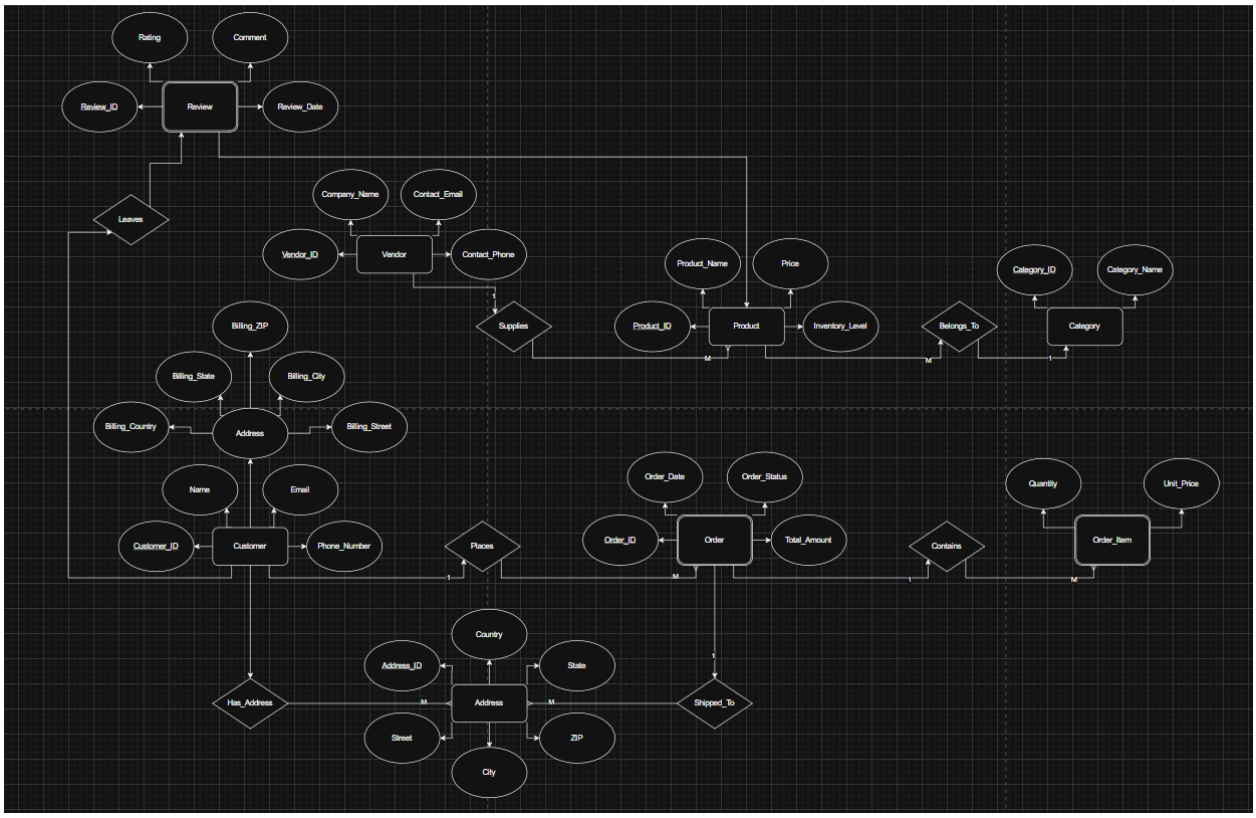
Appointment: Appointment_ID(Simple, PK), Date_Time(Simple), Purpose(Simple), Notes(Simple)

Prescription: PrescriptioonID(Simple, PK), Dosage(Simple), Instruction(Simple), Date(Simple)

Hospital Room: Room_Number(Simple, PK), Room_Type(Simple), Capacity(Simple)



Task 2.2



2) Entity Order is weak, because order can not exist without Customer

3) Customer and Product

Task 4.1

1) StudentID → StudentName, StudentMajor

ProjectID → ProjectTitle, ProjectType, SupervisorID

SupervisorID → SupervisorName, SupervisorDept

{StudentID, PProjectID} → Role, HoursWorked, StartDate, EndDate, StudentName, StudentMajor, ProjectTitle, ProjectType, SupervisorID, SupervisorName, SupervisorDept

2) Redundancy: studentName, StudentMajor are repeated for each project in which he participates.

Example: If a student, Ivan Petrov, participates in 3 projects, his name and specialty will be recorded 3 times.

The project data (ProjectTitle, ProjectType, SupervisorID) is repeated for each student working on this project. The supervisor's data (SupervisorName, SupervisorDept) is repeated for each project that he directs, and, as a result, for each student record in this project.

1. Update Anomaly:

   If a student changes his major, it is necessary to update all the rows where he meets. If we forget to update some row, the data will become inconsistent.

2. Insertion Anomaly:

It is not possible to add a new project to the database until at least one student has been assigned to it.

3. Deletion Anomaly:

   If the last student working on the project is removed, we also permanently lose all information about the project itself (its name, type, and supervisor).

3) The table is already in 1NF

4) a) {StudentID, ProjectID}

b) StudentID → StudentName, StudentMajor

ProjectID → ProjectTitle, ProjectType, SupervisorID

c) Students (StudentID, StudentName, StudentMajor)

PK: StudentID

Projects (ProjectID, ProjectTitle, ProjectType, SupervisorID)

PK: ProjectID

Supervisors (SupervisorID, SupervisorName, SupervisorDept)

PK: SupervisorID

Student_Project_Assignment (StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate)

PK: {StudentID, ProjectID}

FK: StudentID REFERENCES Students(StudentID)

FK: ProjectID REFERENCES Projects(ProjectID)

5) a) ProjectID → SupervisorID

SupervisorID → SupervisorName, SupervisorDept

b) Students (StudentID, StudentName, StudentMajor)

PK: StudentID

Supervisors (SupervisorID, SupervisorName, SupervisorDept)

PK: SupervisorID

Projects (ProjectID, ProjectTitle, ProjectType, SupervisorID)

PK: ProjectID

FK: SupervisorID REFERENCES Supervisors(SupervisorID)

Student_Project_Assignment (StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate)

PK: {StudentID, ProjectID}

FK: StudentID REFERENCES Students(StudentID)

FK: ProjectID REFERENCES Projects(ProjectID)

Task 4.2

1) {StudentID, CourseID, Timeslot, Room}

2) StudentID → StudentMajor

CourseID → CourseName

InstructorID → InstructorName

Room → Building

{CourseID, TimeSlot, Room} → InstructorID, InstructorName

{StudentID, CourseID, TimeSlot, Room} → StudentMajor, CourseName, InstructorID, InstructorName

3) The table is not in BCNF, because StudentID, CourseID, InstructorID and Room are not superkeys

4) Decomposition based on StudentID → StudentMajor

R1(StudentID, StudentMajor)

R2(StudentID, CourseID, CourseName, InstructorID, InstructorName, TimeSlot, Room, Building)

Decomposition of R2 based on CourseID → courseName

R21(CourseID, CourseName)

R22(StudentID, CourseID, InstructorID, InstructorName, TimeSlot, Room, Building)

Decomposition of R22 based on InstructorID → InstructorName

R31(InstructorID, InstructorName)

R32(StudentID, CourseID, InstructorID, TimeSlot, Room, Building)

Decomposition of R32 based on Room → Building

R41(Room, Building)

R42(StudentID, CourseID, InstructorID, TimeSlot, Room)

We check R42. Remaining dependencies:

{CourseID, TimeSlot, Room} → InstructorID (superkey)

{StudentID, CourseID, TimeSlot, Room} (primary key)

R42 is now in BCNF.

The final table of the BCNF:

Students (StudentID, StudentMajor)

Courses (CourseID, CourseName)

Instructors (InstructorID, InstructorName)

Rooms (Room, Building)

Course_Sections (CourseID, TimeSlot, Room, InstructorID)

FK: InstructorID REFERENCES Instructors(InstructorID)

FK: room REFERENCES Rooms (Room)

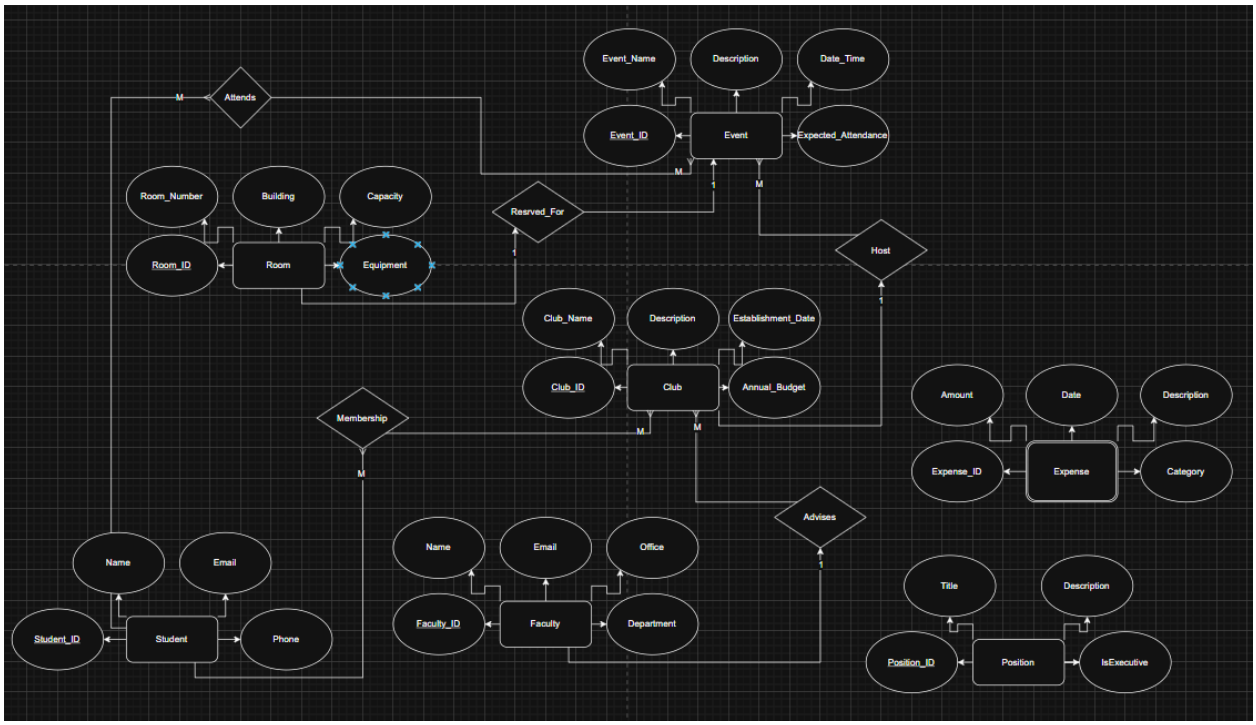
Student_Enrollment (StudentID, CourseID, TimeSlot, Room)

FK: StudentID REFERENCES Students(StudentID)

FK: (CourseID, TimeSlot, Room) REFERENCES Course_Sections (CourseID, TimeSlot, Room)

5) There is no loss of information when decomposing to BCNF. All the attributes of the original table are present in the new schema.

Task 5.1



3) Create one Expenses table with two foreign keys (ClubId, EventID), where one of them should always be NULL (as in the diagram above). It preserves normalization and makes it easy to make reports on all expenses.

4) 1. Find the names and email addresses of all active members of the 'Debate Club' who joined in the last 6 months.

2. List all room reservations for events happening next week. Show the event name, club name, date, and room location

3. For the 'Photography Club', generate a report of all expenses in the current academic year, categorizing them into general club expenses and expenses for specific events. Show the total amount spent in each category.