Data bases Lab1

Part 1

6 different superkeys : {EmpID}, {SSN}, {Email}, {Phone}, {EmpID, Name}, {SSN, Department} .

Candidate keys: EmpID, SSN, Email, Phone

I would choose EmpID as primary key , because it’s more convenient to use, and it’s shorter.

I guess no , but some of them can have one number in case if they use unity work number.

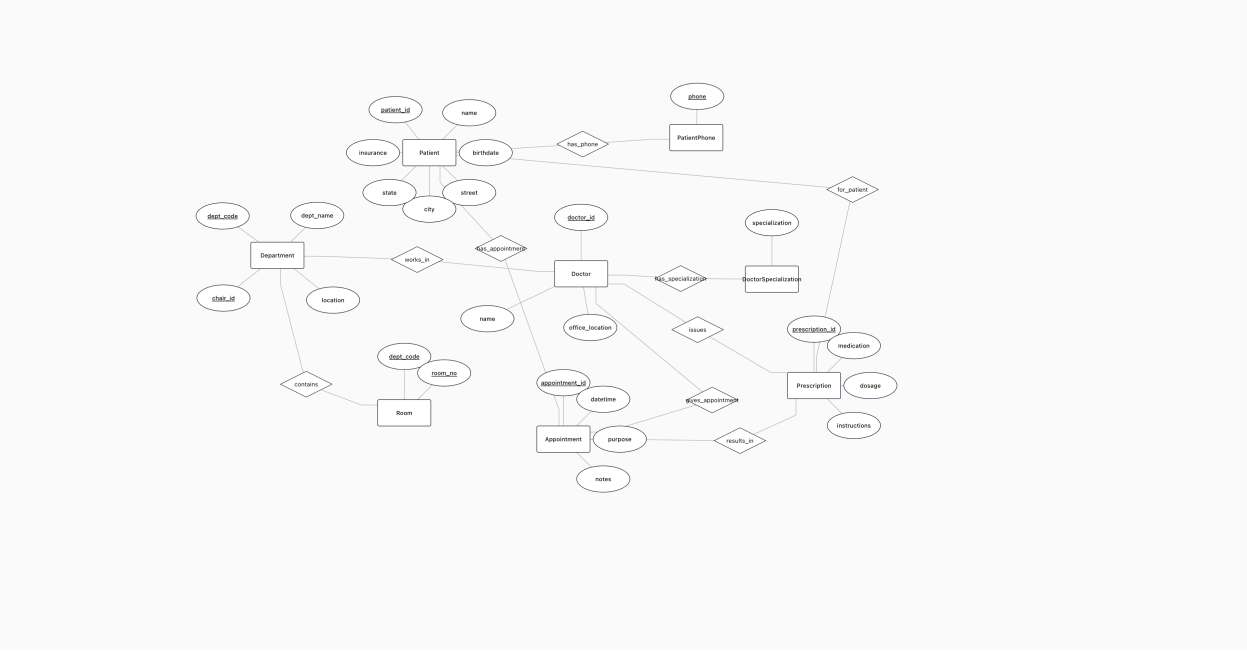
Task 1.2

Student.AdvisorID → Professor.ProfID
Student.Major → Department.DeptCode
Professor.Department → Department.DeptCode
Course.DepartmentCode → Department.DeptCode
Department.ChairID → Professor.ProfID
Enrollment.StudentID → Student.StudentID
Enrollment.CourseID → Course.CourseID

Part 2

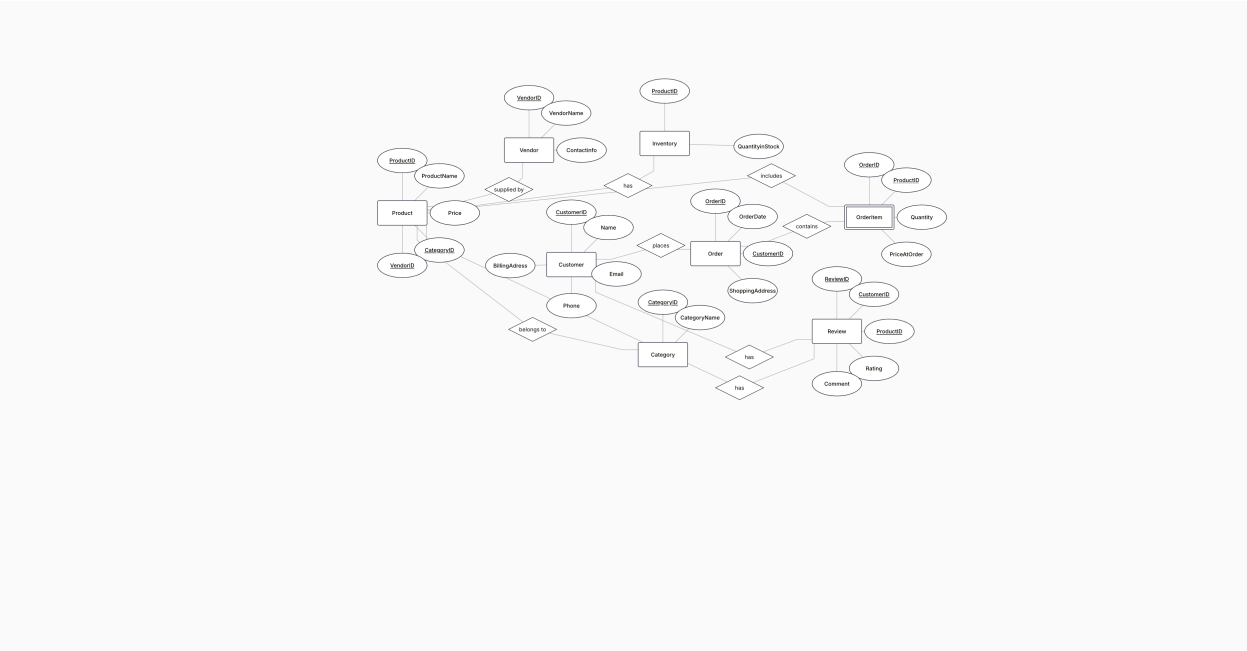
2.1 Hospital Management System

Strong entities: Patient, Doctor, Department, Room, Appointment, Prescription
Weak entities: PatientPhone, DoctorSpecialization



2.2 E-commerce Platform

1.



2. The weak entity is OrderItem.

- It cannot exist without both an Order and a Product.

- Its primary key is a composite key (OrderID + ProductID).
- It depends on the strong entities Order and Product for its identification.

3. The many-to-many relationship is between Order and Product, which is represented by the associative entity OrderItem.

- This relationship requires attributes such as Quantity and PriceAtOrder.
- Without these attributes, the system cannot track how many units of each product were ordered and at what price at the time of the order.

Part 4

4.1

1)

1. StudentID → StudentName, StudentMajor (A student has one name and one major)
2. ProjectID → ProjectTitle, ProjectType (a project has one title and one type)
3. SupervisorID → SupervisorName, SupervisorDept(a supervisor has one name and one department)
4. StudentID , ProjectID → Role, HourdsWorked, startdate, enddate (A student's roles, hours, and dates are specific to a project)

2)

Redundancy: Student details (StudentName, StudentMajor) are repeated for every project a student is on. Project details are repeated for every student on that project.

Update Anomaly: Changing a student's major requires updating multiple rows.

Insert Anomaly: A new project or supervisor cannot be added unless a student is assigned to that project.

Delete Anomaly: Deleting the last row for a project removes all project information.

3)

There are no 1NF violations. All attributes are atomic.

4)

The primary key is {StudentID, ProjectID}. The table is not in 2NF due to partial dependencies.

- StudentID → StudentName, StudentMajor (depends on a subset of the key)
- ProjectID → ProjectTitle, ProjectType, SupervisorID (depends on a subset of the key)

2NF Decomposition:

- Student(StudentID, StudentName, StudentMajor)
- Project(ProjectID, ProjectTitle, ProjectType, SupervisorID)
- StudentProject(StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate)

5)

The Project table from the 2NF decomposition has a transitive dependency.

ProjectID → SupervisorID and SupervisorID → SupervisorName, SupervisorDept.

3NF Decomposition (Final Schemas):

Student(StudentID, StudentName, StudentMajor)

Project(ProjectID, ProjectTitle, ProjectType, SupervisorID)

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

Supervisor(SupervisorID, SupervisorName, SupervisorDept)

4.2

1)

The primary key is {StudentID, CourseID}.

2)

StudentID → StudentMajor

CourseID → CourseName

InstructorID → InstructorName

{TimeSlot, Room} → Building

{CourseID, TimeSlot, Room} → InstructorID

{StudentID, CourseID} → all other attributes.

3)

The table is not in BCNF because there are several FDs where the determinant is not a superkey.

StudentID → StudentMajor

CourseID → CourseName

InstructorID → InstructorName

{TimeSlot, Room} → Building

{CourseID, TimeSlot, Room} → InstructorID

4) BCNF Decomposition:

Student Information: Student(StudentID, StudentMajor)

Course Information: Course(CourseID, CourseName)

Instructor Information: Instructor(InstructorID, InstructorName)

Room Information: Room(Room, Building)

Course Section Information: CourseSection(CourseID, TimeSlot, Room, InstructorID)

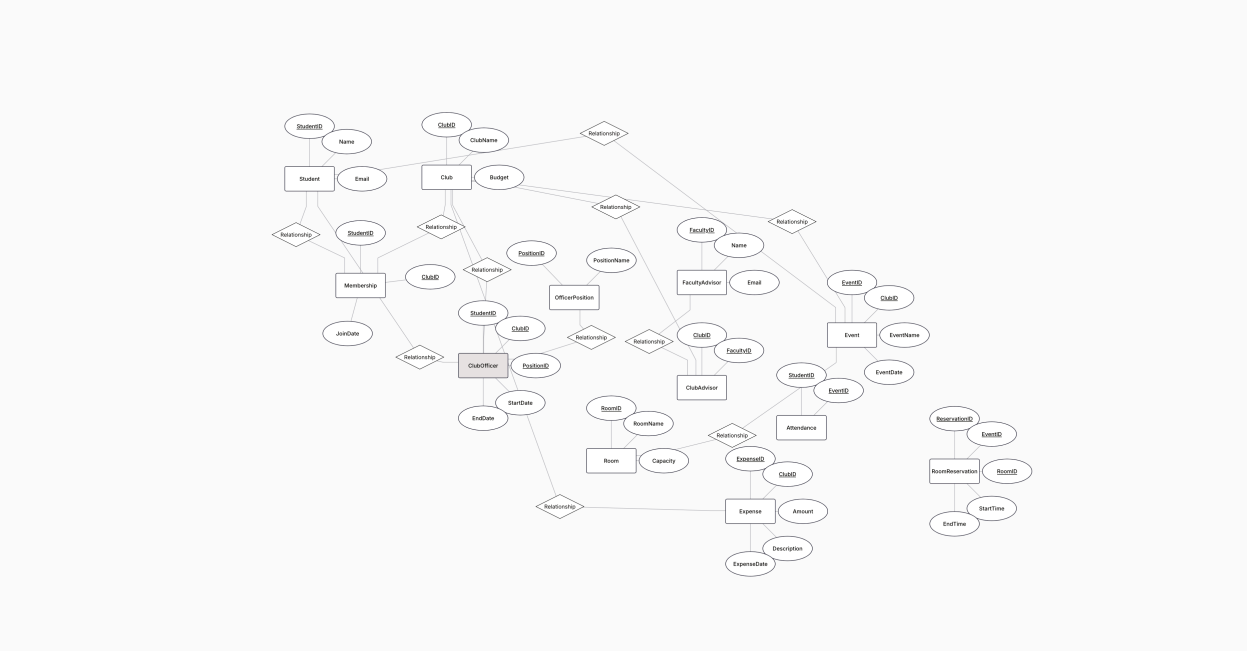
Student Enrollment: StudentEnrollment(StudentID, CourseID, TimeSlot, Room)

5)

The decomposition is lossless-join. No information is lost, but the original single table view is gone. To retrieve the complete course schedule for a student, you must perform joins between the decomposed tables. This is the goal of normalization, as it removes redundancy and prevents anomalies

Part 5

1)



2)

Student(StudentID PK, Name, Email)

Club(ClubID PK, ClubName, Budget)

Membership(StudentID PK, ClubID PK, JoinDate, FK → Student, FK → Club)

OfficerPosition(PositionID PK, PositionName)

ClubOfficer(StudentID PK, ClubID PK, PositionID PK, StartDate, EndDate, FK → Student, FK → Club, FK → OfficerPosition)

FacultyAdvisor(FacultyID PK, Name, Email)

ClubAdvisor(ClubID PK, FacultyID PK, FK → Club, FK → FacultyAdvisor)

Event(EventID PK, ClubID FK, EventName, EventDate)

Attendance(StudentID PK, EventID PK, FK → Student, FK → Event)

Room(RoomID PK, RoomName, Capacity)

RoomReservation(ReservationID PK, EventID FK Unique, RoomID FK, StartTime, EndTime)

Expense(ExpenseID PK, ClubID FK, Amount, Description, ExpenseDate)

3)

One design decision was how to model club officers. I chose to create a separate entity OfficerPosition and a relationship table ClubOfficer, instead of storing officer role inside Membership. This allows students to hold different officer roles over time and keeps the schema normalized.

4)

- Find all students who are officers in the Computer Science Club.
- List all events scheduled for next week with their reserved rooms and times.
- Show the total expenses of each club in the past semester.