Lab work 1(week 2)

Part 1 : identification key Exercises

task 1.1

Relation A: Employee

1. Superkeys:1){EmpID};

2) {SSN};

3) {Email};

4){ Phone}

5){ EmpID, SSN, Email, Phone}

6) {EmpID, SSN, Email, Phone, Name, Department, Salary}

7) {EmpID, SSN}

8) {SSN, Email, Name}

2.Candinate keys: {EmpID};{SSN};{Email};{Phone}

3.Primary key:{EMpID}

EmpID is the primary key because it is a stable, efficient, and purpose-built unique identifier.

4. Based on the data provided, no, so all values in the Phone column are unique.

Relation B: Course Registration

1.. Minimum Attributes for the Primary Key:

{StudentID, CourseCode, Section, Semester, Year}

2. StudentID: Uniquely identifies the student. The core of the registration record.

CourseCode: Specifies which course the student is taking. A student registers for multiple courses.

Section: Distinguishes between different groups or lectures of the same course offered in the same semester. A student can register for multiple sections of one course.

Semester & Year: Define when the course was taken. Allows a student to retake the exact same course and section in a future term.

3. Additional Candidate Key: {StudentID, Section, Semester, Year} exists if Section uniquely identifies a course within a term.

Task 1.2 Foreign Key Design

1.Table:Student

AdvisorID -> Professors (ProfId)

2.Table:Professors

Department -> Departament(DeptCode)

3.Table:Course

DepartmentCode -> Departament(DeptCode)

4.Table:Departament

ChairID -> Professors(ProfID)

5.Table:Enrollment

StudentID -> Student(StudentID)

CourseID -> Course(CourseID)

Part 2: ER Diagram Construction

Task 2.1: Hospital Management System

1. Entities and Types

Patient: Strong Entity

Doctor: Strong Entity

Department: Strong Entity

HospitalRoom: Weak Entity (existence depends on Department)

Appointment: Weak Entity (existence depends on Patient and Doctor)

Prescription: Weak Entity (existence depends on Patient and Doctor)

2. Attributes and Classification

Patient

PatientID: Simple, Primary Key (PK)

Name: Composite (Can be split into FirstName, LastName)

Birthdate: Simple

Address: Composite (Composed of Street, City, State, Zip)

PhoneNumber: Multi-valued

InsuranceInfo: Simple

Doctor

DoctorID: Simple, Primary Key (PK)

Name: Composite

Specialization: Multi-valued

PhoneNumber: Simple

OfficeLocation: Simple

Department

DeptCode: Simple, Primary Key (PK)

DeptName: Simple

Location: Simple

HospitalRoom (Weak Entity)

RoomNumber: Partial Key

Capacity: Simple (Often stored as simple)

Primary Key: Composite (DeptCode, RoomNumber)

Appointment (Weak Entity)

AppointmentID: Simple, Primary Key (PK) (Surrogate key approach)

DateTime: Simple

Purpose: Simple

Notes: Simple

Prescription (Weak Entity)

PrescriptionID: Simple, Primary Key (PK) (Surrogate key approach)

MedicationName: Simple

Dosage: Simple

Instructions: Simple

IssueDate: Simple

3. Relationships and Cardinalities

1)DOCTOR *works_in* DEPARTMENT

- **Cardinality:** N:1 (Many-to-One)
- **Participation:** Doctor **mandatory** (A doctor must work in a department), Department **optional** (A department can exist with no doctors yet).

2)APPOINTMENT *links* PATIENT and DOCTOR

- This is a **recursive relationship** realized through the weak entity Appointment.
- **Cardinality:**
  - PATIENT *has* APPOINTMENT: 1:N (One-to-Many)
  - DOCTOR *has* APPOINTMENT: 1:N (One-to-Many)
- **Participation:** Appointment has **mandatory** participation with both Patient and Doctor.

3)PRESCRIPTION *links* PATIENT and DOCTOR

- This is a **recursive relationship** realized through the weak entity Prescription.

- **Cardinality:**
  - PATIENT *receives* PRESCRIPTION: 1:N (One-to-Many)
  - DOCTOR *writes* PRESCRIPTION: 1:N (One-to-Many)
- **Participation:** Prescription has **mandatory** participation with both Patient and Doctor.

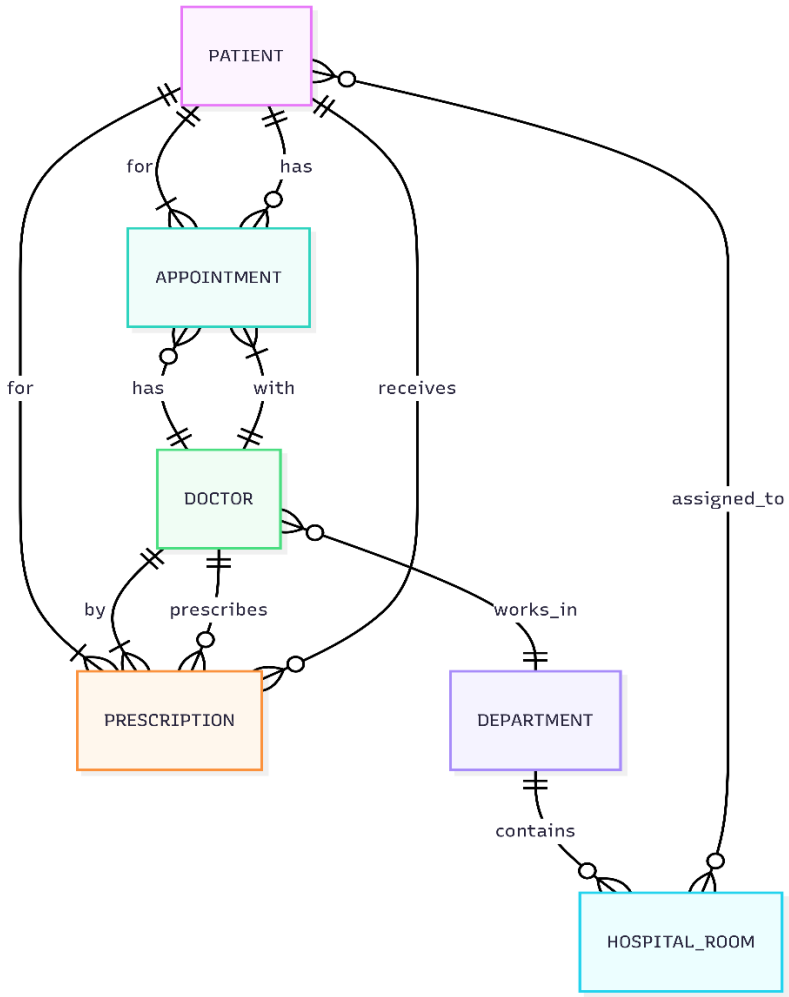
4)HOSPITALROOM *is_located_in* DEPARTMENT (Identifying Relationship)

- **Cardinality:** N:1 (Many-to-One)
- **Participation:** HospitalRoom **mandatory**, Department **optional**.

5)PATIENT *assigned_to* HOSPITALROOM

- **Cardinality:** M:1 (Many-to-One) (Assuming a patient is in one room at a time, and a room can house many patients over time).
- **Participation:** Optional for both (A patient might not be admitted to a room; a room might be empty).

4. Complete ER Diagram



5.Primary keys

Patient:PatientID

Doctor:DoctorID

Departament:DeptCode

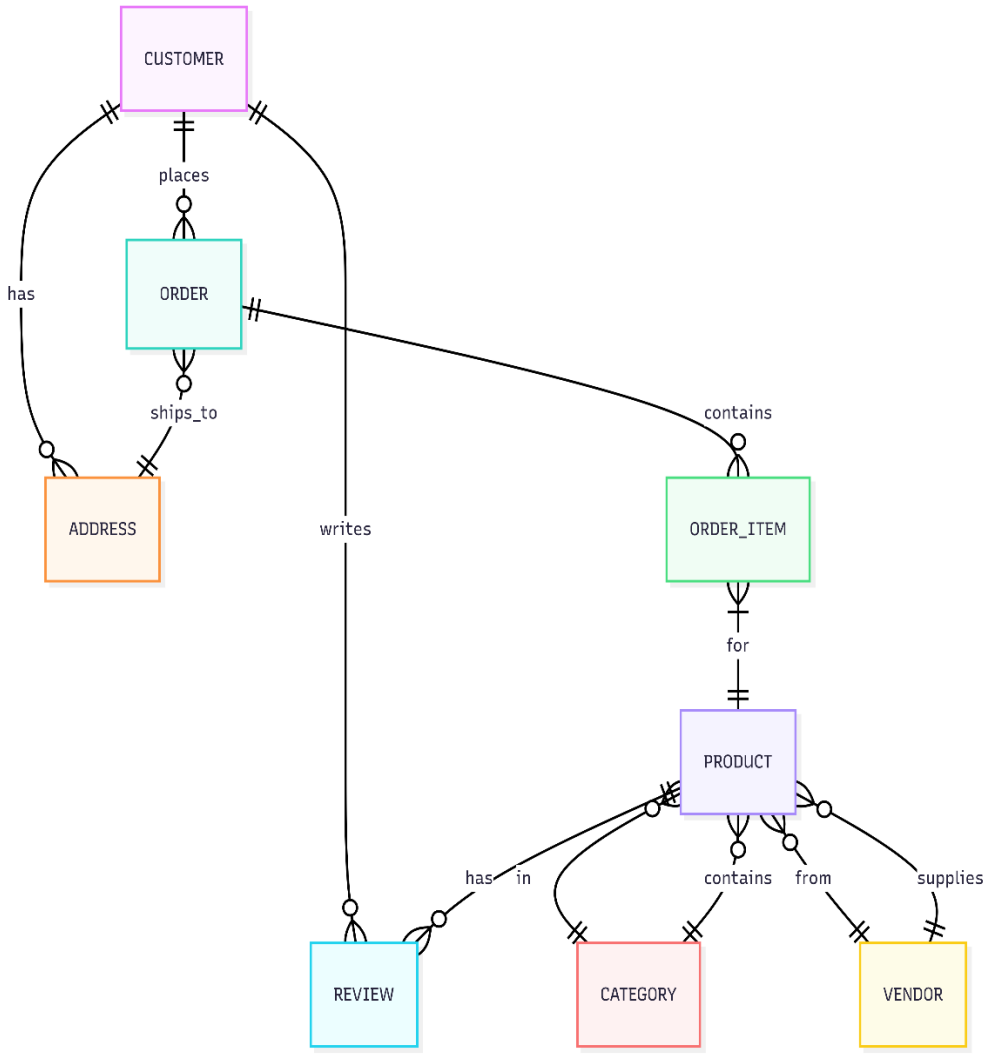
HospitalRoom:RoomNumber

Appointment:AppointmentID

Prescription:PrescriptionID

Task 2.2

1.



2. Weak entinity:ORDER_ITEM,because cannot exist without ORDER

3. M:N relationship with atributies:

CUSTOMER rates PRODUCTS -> becomes REVIEW entity with:

1)rating

2)comment

3)date

Part 4: Normalization Workshop

Task 4.1 Normalization

1. Functional Dependencies (FDs):

StudentID -> StudentName, StudentMajor

ProjectID -> ProjectTitle, ProjectType, SupervisorID

SupervisorID -> SupervisorName, SupervisorDept

{StudentID, ProjectID} -> Role, HoursWorked, StartDate, EndDate, StudentName, StudentMajor, ProjectTitle and all other attributes

2. Problems:

   Redundancy: Repeated student, project, and supervisor data for every project assignment.

Update Anomaly: Changing a student's major requires updates in all their project records.

Insert Anomaly:Cannot add a new supervisor without assigning them to a project first.

Delete Anomaly:Deleting a project might accidentally remove supervisor information.

3. 1NF:

No violation. All attributes contain atomic values.

4. 2NF:

Primary Key:(StudentID, ProjectID)

Partial Dependencies:

StudentID -> StudentName, StudentMajor

ProjectID -> ProjectTitle, ProjectType, SupervisorID

2NF Decomposition:

Students(StudentID, StudentName, StudentMajor)

Projects(ProjectID, ProjectTitle, ProjectType, SupervisorID)

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

5. 3NF:

Transitive Dependency: SupervisorID -> SupervisorName, SupervisorDept (in Projects table)

3NF Decomposition:

Students(StudentID, StudentName, StudentMajor)

Supervisors(SupervisorID, SupervisorName, SupervisorDept)
Projects(ProjectID, ProjectTitle, ProjectType, SupervisorID)

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

Task 4.2: Advanced Normalization

1. Primary Key: (StudentID, CourseID, TimeSlot)

2. Functional Dependencies (FDs):

StudentID → StudentMajor

CourseID → CourseName

InstructorID → InstructorName

TimeSlot, Room → Building(Since rooms are unique across campus)

CourseID, TimeSlot → InstructorID, Room, Building(Each course section is taught by one instructor at one time in one room)

3. BCNF Check:

No, the table is not in BCNF. There are violations because of the FDs where the left side is not a superkey (e.g., StudentID → StudentMajor,CourseID → CourseName, etc.).

4. BCNF Decomposition:

1. Remove StudentID → StudentMajor:

Students(StudentID, StudentMajor)

Schedule1(StudentID, CourseID, CourseName, InstructorID, InstructorName, TimeSlot, Room, Building)

2. Remove CourseID → CourseName:

Courses(CourseID, CourseName)

Schedule2(StudentID, CourseID, InstructorID, InstructorName, TimeSlot, Room, Building)

3. Remove InstructorID → InstructorName:

Instructors(InstructorID, InstructorName)

Schedule3(StudentID, CourseID, InstructorID, TimeSlot, Room, Building)

4. Remove TimeSlot, Room → Building:

Rooms(TimeSlot, Room, Building)

Schedule4(StudentID, CourseID, InstructorID, TimeSlot, Room)

5. Remove CourseID, TimeSlot → InstructorID, Room:

Sections(CourseID, TimeSlot, InstructorID, Room)`

Enrollment(StudentID, CourseID, TimeSlot)`

Final BCNF Tables:

Students(StudentID, StudentMajor)

Courses(CourseID, CourseName)

Instructors(InstructorID, InstructorName)

Rooms(TimeSlot, Room, Building)

Sections(CourseID, TimeSlot, InstructorID, Room)

Enrollment(StudentID, CourseID, TimeSlot)

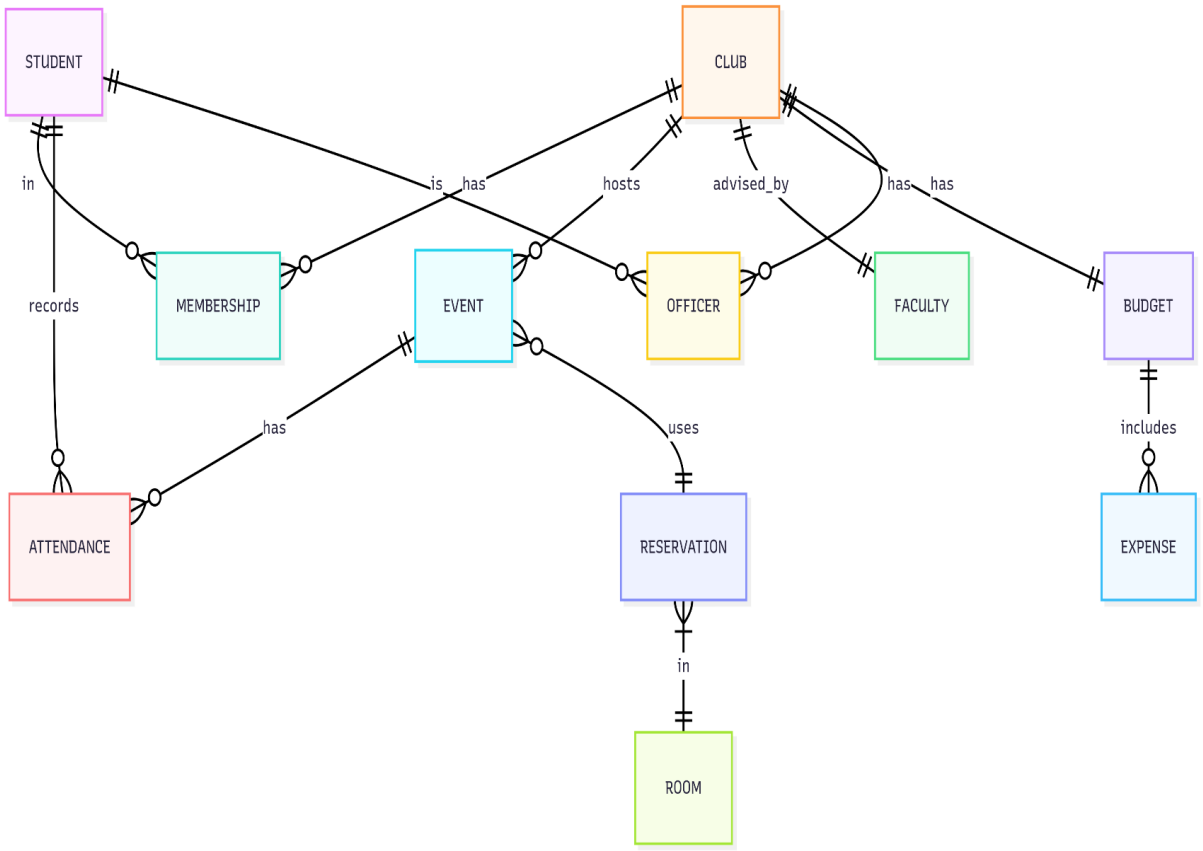
5. Loss of Information:

The decomposition is lossless because at each step, the decomposed tables share attributes that form a key in one of them. All original information can be recovered by joining the tables.

Part 5:Design challenge

Task 5.1 Real-World Application

1.



3. Normalized Relational Schema

Students(StudentID, Name, Email)

Clubs(ClubID, Name, Description, FacultyID)

Faculty(FacultyID, Name, Department)

Events(EventID, ClubID, Title, Date, Description)

Rooms(RoomID, Building, Capacity)

Memberships(StudentID, ClubID, Role, JoinDate)

Attendance(StudentID, EventID, Status)

Officers(StudentID, ClubID, Position, StartDate, EndDate)

Budgets(ClubID, TotalAmount, SpentAmount)

Expenses(ExpenseID, ClubID, Amount, Description, Date)

Reservations(EventID, RoomID, TimeSlot)

3. Key Design Decision

Choice: Separate `Officers` table instead of adding `position` to `Memberships`.

Reason:Avoids NULL values for regular members and better handles officer-specific data (start/end dates) and history.

4. Example Queries

1. "Show all events for the Chess Club next month with their locations"

2. "Find all clubs where a student is both a member and an officer"

3. "Calculate total expenses per club for current semester"