Database Systems Laboratory Work

Part 1: Key Identification Exercises

Изображение выглядит как текст, Шрифт, снимок экрана, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Relation A: Employee

Task:

1. List of Superkeys (at least 6):

* {EmpID}
* {SSN}
* {Email}
* {EmpID, Name}
* {SSN, Phone}
* {Email, Department}

2. Candidate Keys:

* {EmpID}
* {SSN}
* {Email}

3. I would choose EmpID as the primary key because:

* it is short and numeric (better for indexing),
* it is guaranteed to be unique and stable,
* SSN and Email are sensitive or can change.

4. Yes, they can. For example, two employees in the same department may share an office phone.

Relation B: Course Registration

Tasks:

1. Minimum attributes for the primary key:

* {StudentID, CourseCode, Section, Semester, Year}

2. Why each attribute is necessary:

* StudentID → identifies the student.
* CourseCode + Section → identifies the specific course section.
* Semester + Year → distinguishes between different academic terms.

3. Additional Candidate Keys:

* The only candidate key is {StudentID, CourseCode, Section, Semester, Year}.

Task 1.2: Foreign Key Design

Given Tables:

Student(StudentID, Name, Email, Major, AdvisorID)

Professor(ProfID, Name, Department, Salary)

Course(CourseID, Title, Credits, DepartmentCode)

Department(DeptCode, DeptName, Budget, ChairID)

Enrollment(StudentID, CourseID, Semester, Grade)

1. Foreign Key Relationships:

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

Part 2: ER Diagram Construction

1. Entities

* Patient (strong) → PatientID (PK), Name, BirthDate, Address (Street, City, State, Zip), Insurance
* Doctor (strong) → DoctorID (PK), Name, Specialization (multivalued), Phone, OfficeLocation
* Department (strong) → DeptCode (PK), DeptName, Location
* Appointment (weak) → AppointmentID (PK), PatientID (FK), DoctorID (FK), DateTime, Purpose, Notes
* Prescription (weak) → PrescriptionID (PK), PatientID (FK), DoctorID (FK), Medication, Dosage, Instructions
* Room (strong, but composite key) → (DeptCode + RoomNumber), Capacity

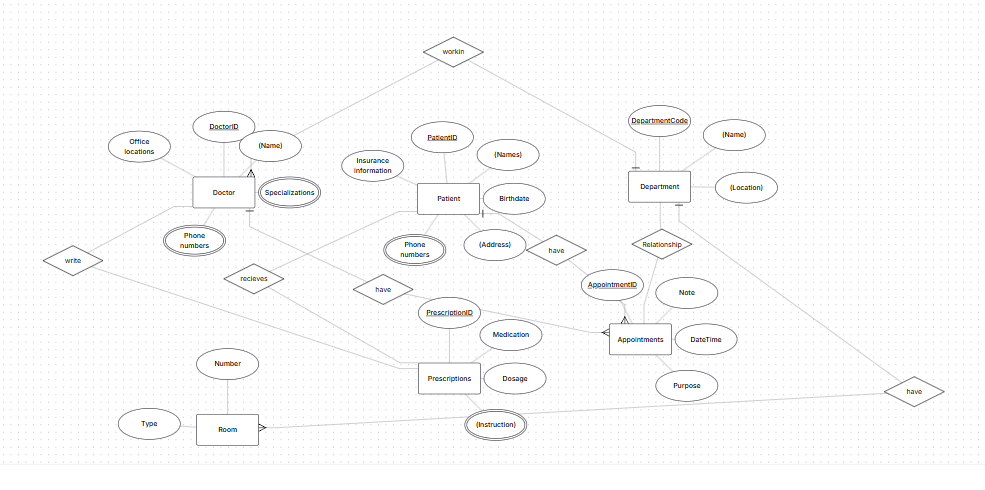
2. Attributes Classification

* Simple attributes: Name, BirthDate, Insurance, DeptName, Salary
* Composite attributes: Address (Street, City, State, Zip)
* Multivalued attributes: PatientPhone, DoctorSpecialization
* Derived attributes (optional): PatientAge (from BirthDate)

3. Relationships & Cardinalities

* Patient — Appointment — Doctor: M:N (a patient can have many appointments with many doctors)
* Doctor — Department: M:1 (each doctor belongs to one department; a department has many doctors)
* Patient — Prescription — Doctor: M:N (a doctor prescribes to many patients; a patient can have prescriptions from many doctors)
* Department — Room: 1:N (a department has many rooms; a room belongs to one department)

4

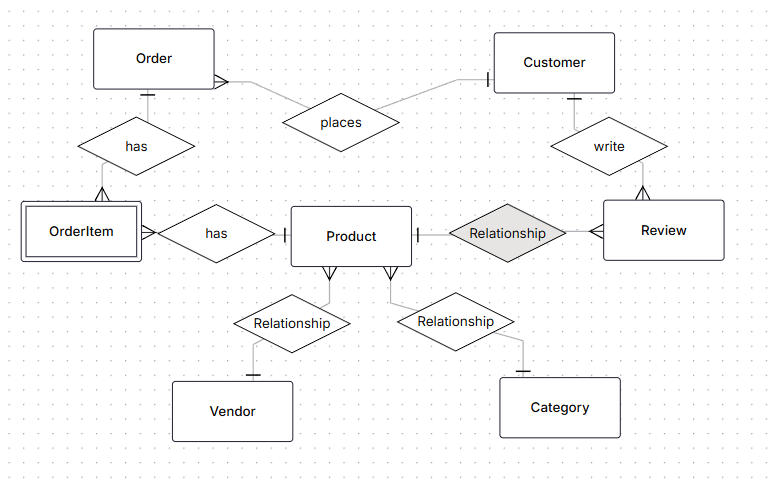


5.PK

* Patient → PatientID (PK)
* Doctor → DoctorID (PK)
* Department → DeptCode (PK)
* Room → (DeptCode, RoomNumber) (composite PK, since room numbers repeat in different departments)
* Appointment → AppointmentID (PK)
* Prescription → PrescriptionID (PK)

Task 2.2: E-commerce Platform

1.



2. Weak Entity

OrderItem is a weak entity:

* + It has no independent existence without Order and Product.

3. Many-to-Many Relationship with Attributes

* The Order ↔ Product relationship is M:N.
* It cannot be modeled directly, so we introduce OrderItem.
* This weak entity stores attributes of the relationship (Quantity, PriceAtOrder).

Task 4.1: Denormalized Table Analysis

1. Functional Dependencies (FDs)

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

Redundancy:

* StudentName and StudentMajor are repeated for every project the student participates in.
* ProjectTitle and ProjectType are repeated for each student working on the same project.
* SupervisorName and SupervisorDept are repeated for each student with the same supervisor.

Update Anomaly:

* If a supervisor changes their department, the update must be made in multiple rows.

Insert Anomaly:

* We cannot add a new student without assigning them to a project.

Delete Anomaly:

* If we delete the last student working on a project, we lose information about the project and its supervisor.

3. First Normal Form (1NF)

* The table is already in 1NF: all attributes are atomic, and there are no repeating groups.

4. Second Normal Form (2NF)

* Primary Key: (StudentID, ProjectID)
* Partial Dependencies:
  + StudentID → StudentName, StudentMajor
  + ProjectID → ProjectTitle, ProjectType
  + SupervisorID → SupervisorName, SupervisorDept

Decomposition into 2NF:

* Student(StudentID PK, StudentName, StudentMajor)
* Project(ProjectID PK, ProjectTitle, ProjectType, SupervisorID FK)
* Supervisor(SupervisorID PK, SupervisorName, SupervisorDept)
* StudentProject(StudentID FK, ProjectID FK, Role, HoursWorked, StartDate, EndDate, PK = (StudentID, ProjectID))

5. Third Normal Form (3NF)

* In Supervisor, SupervisorDept depends only on SupervisorID → no issue.
* In Project, SupervisorID is a foreign key, so no transitive dependency exists.

Final 3NF Schema:

* Student(StudentID PK, StudentName, StudentMajor)
* Project(ProjectID PK, ProjectTitle, ProjectType, SupervisorID FK)
* Supervisor(SupervisorID PK, SupervisorName, SupervisorDept)
* StudentProject(StudentID FK, ProjectID FK, Role, HoursWorked, StartDate, EndDate, PK = (StudentID, ProjectID))

Task 4.2: Advanced Normalization  
1. Primary Key

* The primary key is (StudentID, CourseID, TimeSlot).
  + StudentID is needed to identify the student enrolled.
  + CourseID + TimeSlot is needed to identify the exact course section.

2. Functional Dependencies (FDs)

1. StudentID → StudentMajor
2. CourseID → CourseName
3. InstructorID → InstructorName
4. Room → Building (because rooms are unique across campus)
5. (CourseID, TimeSlot) → InstructorID, Room
6. (StudentID, CourseID, TimeSlot) → (StudentMajor, CourseName, InstructorID, InstructorName, Room, Building)

3. Check for BCNF

* In BCNF, for every non-trivial FD X → Y, X must be a superkey.

Let’s check:

* StudentID → StudentMajor (StudentID alone is not a superkey).
* CourseID → CourseName (CourseID is not a superkey in the whole table).
* InstructorID → InstructorName (InstructorID is not a superkey).
* Room → Building (Room is not a superkey in the table).
* (CourseID, TimeSlot) → InstructorID, Room (this determines a section, but still not a superkey).

Thus, the table is not in BCNF.

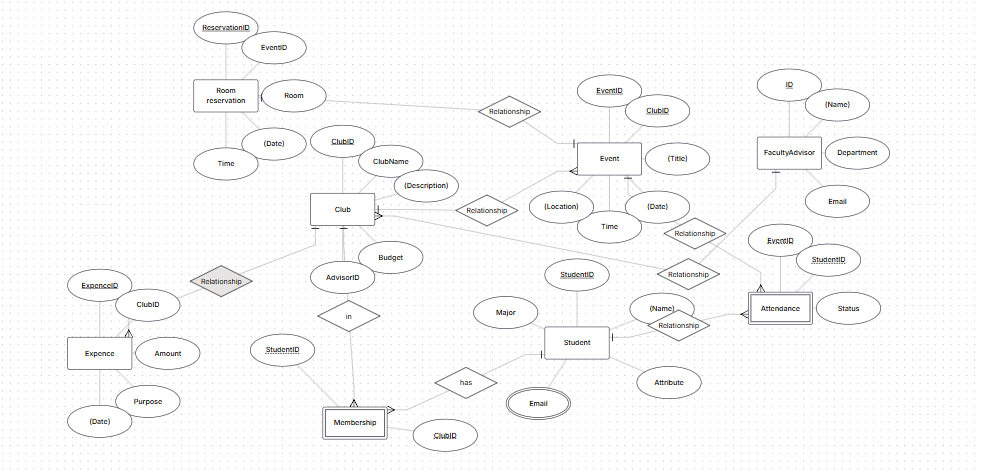
4. Decomposition into BCNF

* From StudentID → StudentMajor:
  + Student(StudentID PK, StudentMajor)
* From CourseID → CourseName:
  + Course(CourseID PK, CourseName)
* From InstructorID → InstructorName:
  + Instructor(InstructorID PK, InstructorName)
* From Room → Building:
  + Room(Room PK, Building)
* From (CourseID, TimeSlot) → InstructorID, Room:
  + CourseSection(CourseID FK, TimeSlot, InstructorID FK, Room FK,  
    PK = (CourseID, TimeSlot))
* Relationship between students and course sections:
  + Enrollment(StudentID FK, CourseID FK, TimeSlot,  
    PK = (StudentID, CourseID, TimeSlot))

5. Information Loss

* This decomposition is lossless, because:
  + We preserved all functional dependencies.
  + Every original attribute appears in at least one of the decomposed tables.
* However, queries now require joins across multiple tables (e.g., to see the full course schedule for one student).

Part 5: Design Challenge



2. Design Decision

A critical design choice was whether to model Membership as a simple many-to-many relationship or as a separate entity.

* If we only needed to track which students belong to which clubs, a plain M:N relationship would suffice.
* However, since we must also record roles (president, treasurer, secretary, etc.), Membership is designed as a weak entity with Role as an attribute.

3. Example Queries (in English)

1. “Find all students who are officers in the Computer Science Club.”
2. “List all events scheduled for next week with their room reservations.”
3. “Show the total expenses of each club for the last semester.”