Lab 1, Database Systems

Altynbekov Yernazar

Part 1

Task 1.1 – Employee Relation, Registration Relation

Relation (A):

```
Employee(EmpID, SSN, Email, Phone, Name, Department, Salary)
```

1) Superkeys

A superkey is any set of attributes that can uniquely identify a tuple.
Here are 6 superkeys for the Employee relation:

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

2) Candidate Keys

Candidate keys are the minimal superkeys.
For this relation, the candidate keys are:

1. EmpID
2. SSN
3. Email

3) Primary Key Choice

The best primary key is **EmpID**.
Reasons:

1. It is short, simple, and system-controlled.
2. SSN is sensitive personal information and should not be used as PK.
3. Email may change over time.
4. EmpID is stable and easy to use as a reference.

4) Can two employees share the same phone number?

In the given data, all phone numbers are different. However, in real life, employees may share a phone (e.g., office phone). So, phones are not guaranteed to be unique.
If the company policy says every employee must have a personal phone, then a unique constraint can be added. Otherwise, it should not be considered a key.

Relation(B):

`Registration(StudentID, CourseCode, Section, Semester, Year, Grade, Credits)`

Primary Key Identification

- A student can take the same course in different semesters or years.
- A course can have multiple sections in the same semester.
- Therefore, the minimal set of attributes that uniquely identifies each registration is:

Primary Key = (StudentID, CourseCode, Section, Semester, Year)

Explanation:

- **StudentID** → identifies the student.
- **CourseCode** → identifies the course.
- **Section** → identifies the specific section of the course.
- **Semester + Year** → identify when the course is taken.

Credits depend only on the course offering, not on the student.

Task 1.2 – Foreign Keys in University System

Relations:

- 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)

Foreign Key Design

1. `Student.AdvisorID → Professor.ProfID`
    - Each student has an advisor who is a professor.
2. `Professor.Department → Department.DeptCode`
    - Each professor works in one department.
3. `Course.DepartmentCode → Department.DeptCode`
    - Each course belongs to a department.
4. `Department.ChairID → Professor.ProfID`
    - Each department has one chair, who is a professor.
5. `Enrollment.StudentID → Student.StudentID`
    - Each enrollment record belongs to a student.
6. `Enrollment.CourseID → Course.CourseID`
    - Each enrollment record refers to a course.

Part 2: ER Diagram Construction
Task 2.1 – Hospital Management System

1) Entities and Attributes

- **Patient**
  - Attributes: PatientID (PK), Name, DOB, Address, Phone, InsuranceInfo
- **Doctor**
  - Attributes: DoctorID (PK), Name, Specialty, Phone, Email
- **Department**
  - Attributes: DeptCode (PK), DeptName, Location
- **Room** (weak entity, because RoomNumber is unique only inside a department)
  - Attributes: RoomNumber (partial key), Capacity
  - Primary Key: (DeptCode, RoomNumber)
- **Admission** (relationship between Patient and Room)
  - Attributes: AdmissionID (PK), AdmissionDate, DischargeDate
- **Treatment** (relationship between Patient and Doctor)
  - Attributes: TreatmentID (PK), Diagnosis, Prescription, TreatmentDate

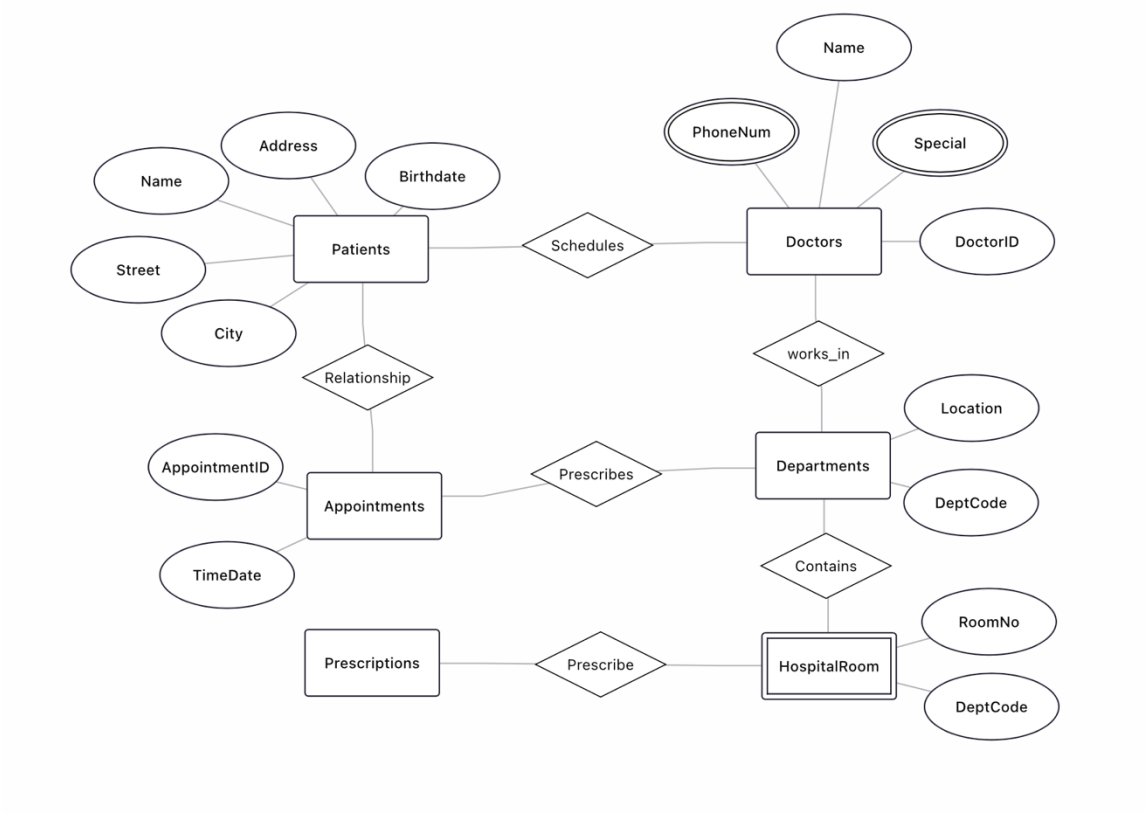
2) Relationships

1. **Patient – Admission – Room**
   - A patient is admitted to one room at a time.
   - A room can have many patients over time.
   - Cardinality: Patient(1..) ↔ *Room(0..)*
2. **Room – Department**
   - Each room belongs to one department.
   - Each department has many rooms.
   - Cardinality: Department(1) ↔ Room(0..*)
3. **Patient – Treatment – Doctor**
   - A patient can be treated by many doctors.
   - A doctor can treat many patients.
   - This is an M:N relationship → converted into **Treatment** entity.
4. **Doctor – Department**
   - Each doctor works in one department.
   - A department can have many doctors.
   - Cardinality: Department(1) ↔ Doctor(0..*)

3) Keys

- Primary Keys: PatientID, DoctorID, DeptCode, (DeptCode, RoomNumber), AdmissionID, TreatmentID.
- Foreign Keys:
  - Room.DeptCode → Department.DeptCode
  - Admission.PatientID → Patient.PatientID
  - Admission.DeptCode, RoomNumber → Room.DeptCode, Room.RoomNumber
  - Treatment.PatientID → Patient.PatientID
  - Treatment.DoctorID → Doctor.DoctorID
  - Doctor.Department → Department.DeptCode

4) ER Diagram



Part 2.2 E-commerce Platform.

Weak entity: *Review*, because it only exists when both a **Customer** and a **Product** are present.

M:N with attributes: *OrderItems*, because an order can contain many products, and products can be in many orders. This relationship has extra attributes: **Quantity** and **PriceAtOrder**.

Task 4.1: Denormalized Table Analysis

Table:

```
StudentProject(StudentID, StudentName, StudentMajor, ProjectID, ProjectTitle,
ProjectType, SupervisorID, SupervisorName, SupervisorDept, Role, HoursWorked,
StartDate, EndDate)
```

1. **Functional Dependencies:**

- StudentID → StudentName, StudentMajor
- ProjectID → ProjectTitle, ProjectType, SupervisorID
- SupervisorID → SupervisorName, SupervisorDept
- (StudentID, ProjectID) → Role, HoursWorked, StartDate, EndDate

2. **Problems:**

- Redundancy: Supervisor information is repeated.
- Update anomaly: If a supervisor changes department, we must update many rows.
- Insert anomaly: We cannot add a supervisor without a project.
- Delete anomaly: If we delete a project, we may lose supervisor info.

3. **1NF:** Already satisfied (all attributes are atomic).
4. **2NF:**
   Primary key = (StudentID, ProjectID).
   Partial dependencies:

- StudentID → StudentName, StudentMajor
- ProjectID → ProjectTitle, ProjectType, SupervisorID

Decomposition:

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

5. **3NF:**
   Removed transitive dependency SupervisorID → SupervisorName, SupervisorDept.

Final tables:

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

Task 4.2: Advanced Normalization

Table:
`CourseSchedule(StudentID, StudentMajor, CourseID, CourseName, InstructorID, InstructorName, TimeSlot, Room, Building)`

1. **Primary Key:** (StudentID, CourseID, TimeSlot)
2. **Functional Dependencies:**

- StudentID → StudentMajor
- CourseID → CourseName
- InstructorID → InstructorName
- (Room, TimeSlot) → Building
- (CourseID, TimeSlot, Room) → InstructorID
- (StudentID, CourseID, TimeSlot) → all other attributes

3. **BCNF Check:** Not in BCNF (because of partial and transitive dependencies).
4. **BCNF Decomposition:**

- Student(StudentID, StudentMajor)
- Course(CourseID, CourseName)
- Instructor(InstructorID, InstructorName)
- Room(Room, TimeSlot, Building)
- CourseSchedule(StudentID, CourseID, TimeSlot, InstructorID, Room)

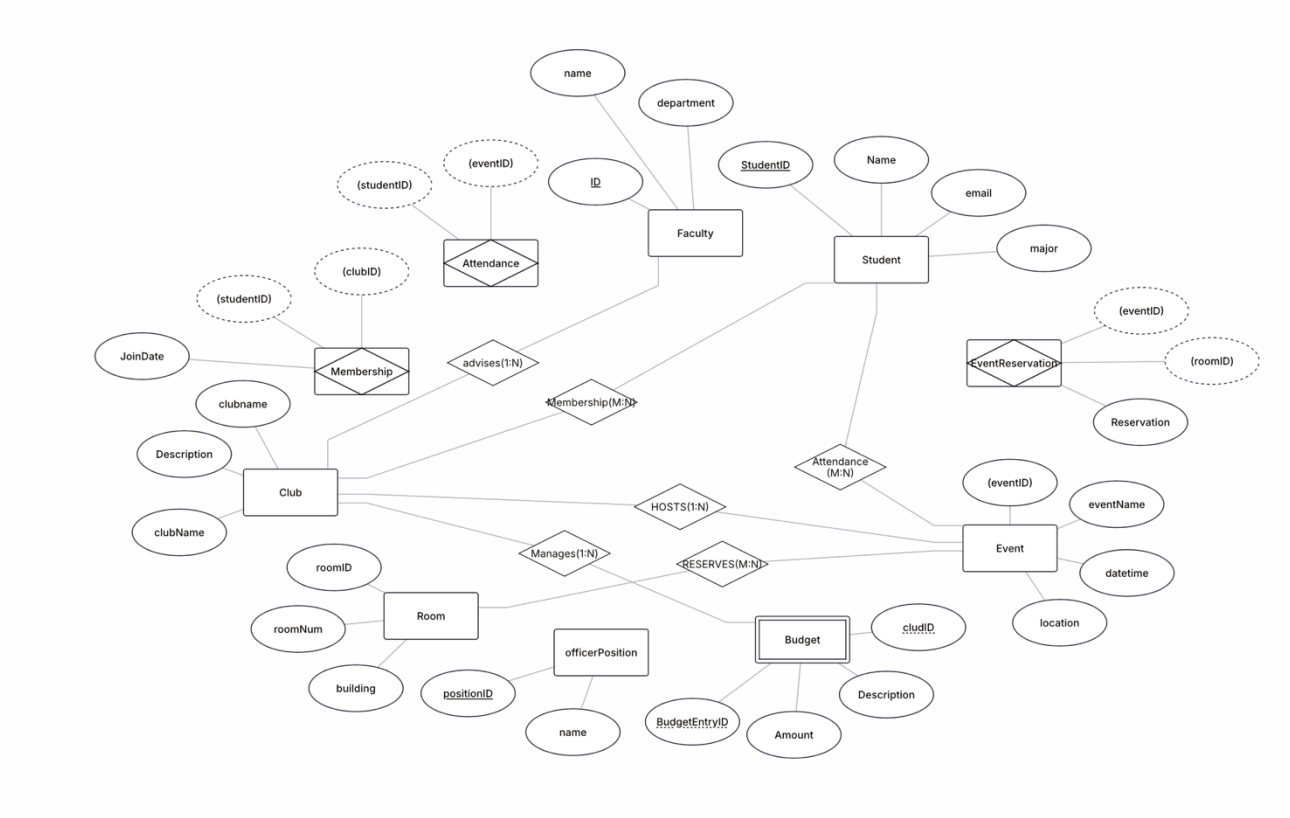
5. **Loss of Information:** No loss of information. All dependencies are preserved with proper keys.

Task 5.1: Real-World Application – Student Clubs Database

System Requirements Recap

- Store **student clubs and organizations**
- Track **club membership** (students ↔ clubs, many-to-many)
- Track **club events** and **student attendance**
- Track **club officer positions** (president, treasurer, secretary, etc.)
- Track **faculty advisors** (each club has one advisor, but a faculty member can advise multiple clubs)
- Track **room reservations** for club events
- Track **club budget and expenses**

1.



2. Relationships

- Student ↔ Club = M:N (via **Membership**)
- Student ↔ Club (officer role) = 1:N (via **ClubOfficer**)
- Club ↔ FacultyAdvisor = 1:N (one club has one advisor, but an advisor can advise many clubs)
- Club ↔ Event = 1:N
- Student ↔ Event = M:N (via **Attendance**)
- Event ↔ Room = M:N (via **Reservation**)
- Club ↔ Expense = 1:N

3. Relational Schema (in 3NF)

Student(StudentID PK, Name, Email, Major)
Club(ClubID PK, ClubName, Description, Budget)
Membership(StudentID PK, ClubID PK, JoinDate, Role)
ClubOfficer(StudentID PK, ClubID PK, Position)
FacultyAdvisor(FacultyID PK, Name, Department)
ClubAdvisor(ClubID PK, FacultyID PK)
Event(EventID PK, ClubID FK, Title, DateTime, Description)
Attendance(StudentID PK, EventID PK, Status)
Room(RoomID PK, Building, Capacity)

Reservation(EventID PK, RoomID PK, DateTime)
Expense(ExpenseID PK, ClubID FK, Amount, Purpose, Date)

4. Example Queries

- Find all students who are officers in the Computer Science Club.
- List all events scheduled for next week with their room reservations.
- Show the total budget and expenses for each club.