Part1: Relational Model & Keys

Task 1.1: Super key and Candidate Key Analysis

Relation A: Employee

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

Tasks: 1. List at least 6 different superkeys 2. Identify all candidate keys 3. Which candidate key would you choose as primary key and why? 4. Can two employees have the same phone number? Justify your answer based on the data shown.

1. Superkeys:

- {EmpID} – unique employee identifier

- {SSN} – Social Security Number

- {Email} – Employees email unique

- {EmpID, Name} – still unique

- {SSN, Department} – still unique

- {Email, Department} – still unique

2. Candidate Keys:

-{EmpID}

-{SSN}

-{Email}

3. I would choose EmpID. Because, it is unique and not sensitive like SSN or changeable like Email.

4. Yes, they actually can. For example, some people have shared family phones or office numbers. Also there can be some mistakes in data entry, which can cause duplicate phone numbers.

Relation B: Course Registration

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

Business Rules:

- A student can take the same course in different semesters

- A student cannot register for the same course section in the same semester

- Each course section in a semester has a fixed credit value

Tasks: 1. Determine the minimum attributes needed for the primary key 2. Explain why each attribute in your primary key is necessary 3. Identify any additional candidate keys (if they exist)

1. primary key is {StudentID, CourseCode, Section, Semester, Year}

minimum attributes needed – 5

2. StudentID – Who

CourseCode – Which Course

Section – Which Section (Section A or Section B and etc)

Semester + Year – When

All of this attributes make data in a row unique

3. I think there is no exist any candidate keys

Task 1.2: Foreign Key Design

Design the foreign key relationships for this university system:

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)

Task: 1. Identify all foreign key relationships

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

   AdvisorID references Professor(ProfID)

2. Course(CourseID, Title, Credits, DepartmentCode)

   DepartmentCode references Department (DeptCode)

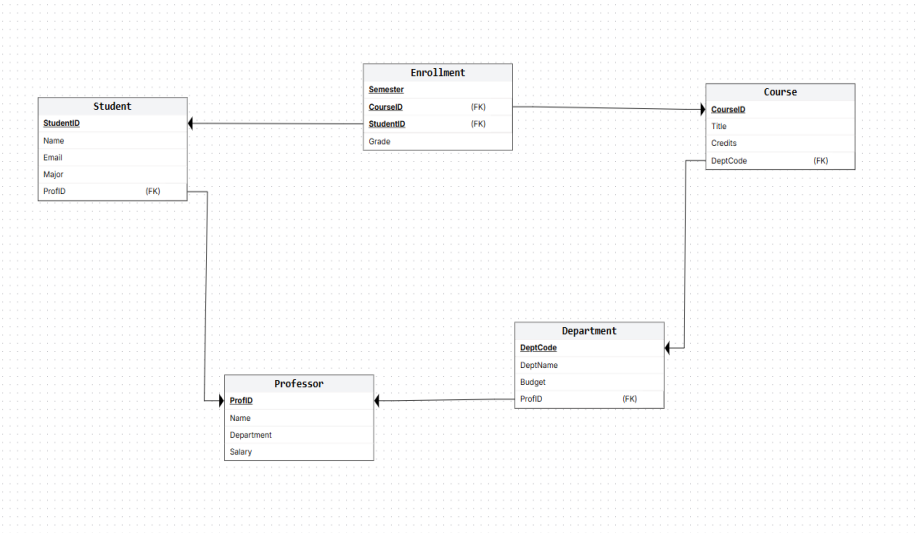
3. Department(DeptCode, DeptName, Budget, ChairID)

   ChairID references Professor(ProfID)

4. Enrollment(StudentID, CourseID, Semester, Grade) - Foreign key

   StudentID references Student(StudentID)

   CourseID references Course(CourseID)



Part 2: ER Diagram Construction

Task 2.1: Hospital Management System

Scenario: Design a database for a hospital management system.

Requirements:

• Patients have unique patient IDs, names, birthdates, addresses (street,city, state,zip), phone numbers (multiple allowed), and insurance information

• Doctors have unique doctor IDs, names, specializations (can have multiple), phone numbers, and office locations

• Departments have department codes, names, and locations

• Appointments track which patient sees which doctor at what date/time, the purpose of visit, and any notes

• Prescriptions track medications prescribed by doctors to patients, including dosage and instructions

• Hospital Rooms are numbered within departments (room 101 in Cardiology is different from room 101 in Neurology)

Tasks: 1. Identify all entities (specify which are strong and which are weak) 2. Identify all attributes for each entity (classify as simple, composite, multi-valued, or derived) 3. Identify all relationships with their cardinalities (1:1, 1:N, M:N) 4. Draw the complete ER diagram using proper notation 5. Mark primary keys

1. All Entities:

Patients, Doctors, Departments, Appointments, Prescriptions, HospitalRooms

Strong Entities:

Patients, Doctors, Departments

Weak Entities:

Appointments, Prescriptions, HospitalRooms

2. All Attributes:

Patients(PatientID, Name, Birthdate, Address, PhoneNumbers, InsuranceInfo)

Simple: PatientID, Name, Birthdate, InsuranceInfo

Composite: Address

Multi-valued: PhoneNumbers

Doctors(DoctorID, Name, Specializations, PhoneNumbers, OfficeLocations)

Simple: DoctorID, Name,

Composite: OfficeLocations

Multi-valued: Specializations, PhoneNumbers

Departments(DeptID, DeptName, DeptLocations)

Simple: DeptID, DeptName

Composite: DeptLocations

Multi-valued: --

Appointments(AppointmentID, PatientID, DoctorID, Date/Time, VisitPurpose, Note) –

PatientID references Patients(PatientID)

DoctorID references Doctors(DoctorID)

Simple: AppointmentID, PatientID, DoctorID, VisitPurpose, Note

Composite: Date/Time

Multi-valued: --

Prescriptions(PrescriptionID, DoctorID, PatientID, PrescribedMedication, Dosage, Instructions) –

PatientID references Patients(PatientID)

DoctorID references Doctors(DoctorID)

Simple: PrescriptionID, DoctorID, PatientID, PrescribedMedication, Dosage, Instructions

Composite: --

Multi-valued: --

HospitalRooms(DeptID, RoomNumber) – DeptID references Departments(DeptID)

Simple: DeptID, RoomNumber

Composite: --

Multi-valued: --

3.

Doctors ------ Departments

(N:1)

Departments ------ HospitalRooms

(1:N)

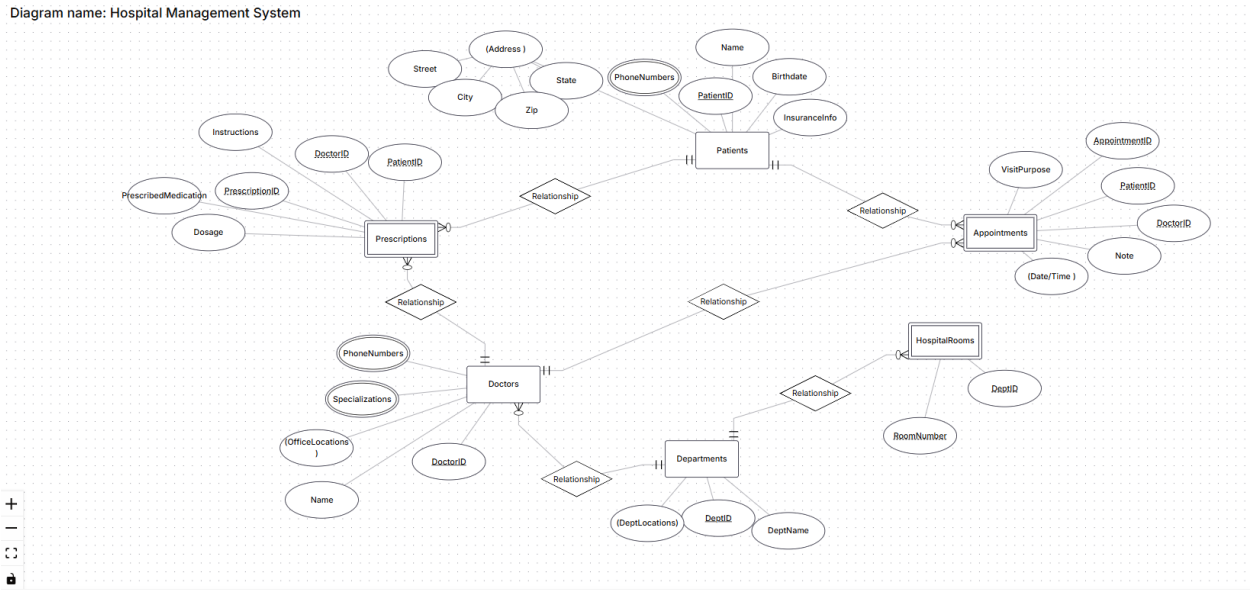
Doctors ------ Prescriptions ------ Patients

(1:N) (N:1)

Patients ------ Appointments ------ Doctors

(1:N) (N:1)

4./5.



Task 2.2: E-commerce Platform

Scenario: Design a simplified e-commerce database.

Requirements:

• Customers place Orders for Products

• Products belong to Categories and are supplied by Vendors

• Orders contain multiple Order Items (quantity and price at time of order)

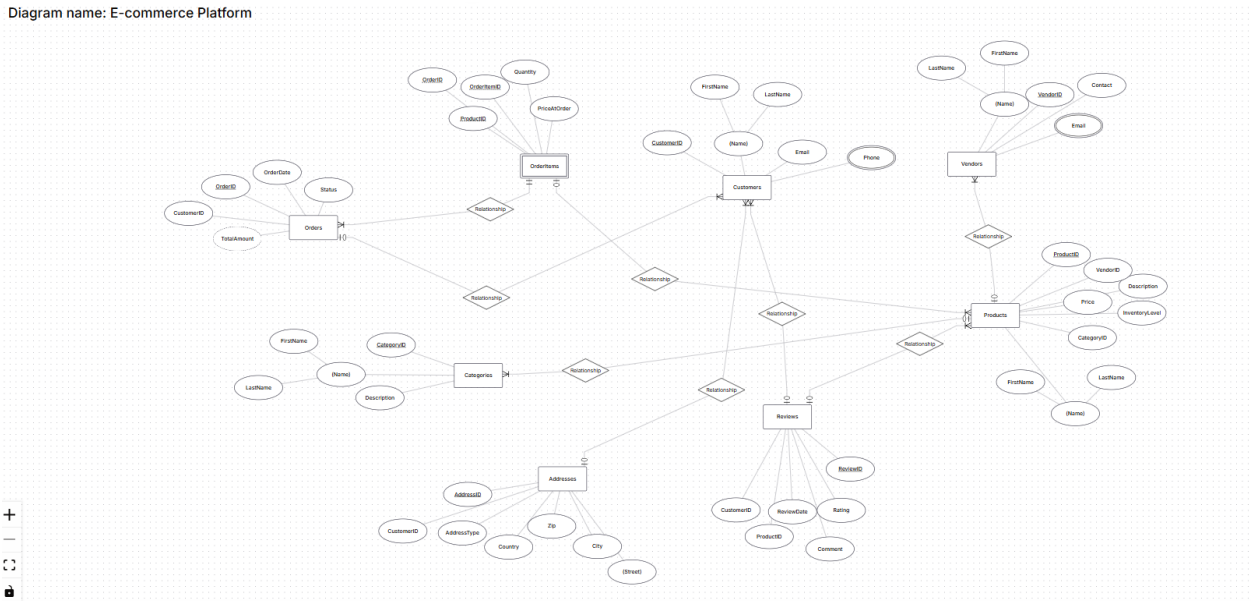
• Products have reviews and ratings from customers

• Track Inventory levels for each product

• Shipping addresses can be different from customer billing addresses

Tasks: 1.Create a complete ER diagram 2.Identify at least one weak entity and justify why it's weak 3.Identify at least one many-to-many relationship that needs attributes

1.



2. In my design, the weak entity is OrderItem. It represents individual products inside a customer's order and includes attributes such as Quantity and PriceAtOrder. OrderItem cannot

exist on its own, it only makes sense as part of a specific Order. Its primary key is based on the identifier of its parent Order together with the Product it refers to, so if the order is deleted, all its order items should be removed as well. That dependency on the owner is the reason why I treat OrderItem as a weak entity

3.

A clear many-to-many relationship in this database is between **Customers** and **Products** through **Reviews**. A customer can leave reviews for multiple products, and each product can receive reviews from many customers.
Because this relationship has its own data — like Rating, Comment, and ReviewDate it needs to be modeled as a separate entity ("Review") instead of just foreign keys in the two tables.

Part 4: Normalization Workshop

Task 4.1: Denormalized Table Analysis

Given Table:

StudentProject(StudentID, StudentName, StudentMajor, ProjectID, ProjectTitle,

ProjectType, SupervisorID, SupervisorName, SupervisorDept,

Role, HoursWorked, StartDate, EndDate)

Your Tasks:

1.Identify functional dependencies: List all FDs in the format A→B

2.Identify problems: - What redundancy exists in this table? - Give specific examples of update, insert, and delete anomalies

3.Apply 1NF: Are there any 1NF violations? How would you fix them?

4.Apply 2NF: - What is the primary key of this table? – Identify any partial dependencies – Show the 2NF decomposition

5.Apply 3NF: - Identify any transitive dependencies – Show the final 3NF decomposition with all able schemas

1.

StudentID → StudentName, StudentMajor

ProjectID → ProjectTitle, ProjectType

SupervisorID → SupervisorName, SupervisorDept

(StudentID, ProjectID, SupervisorID)→ Role, HoursWorked, StartDate, EndDate

2.

Redundancy: Students info repeated for each project

Update anomaly: Change Students major = update multiple rows

Insert anomaly: Cant add new Student without enrollment

Delete anomaly: Delete last student = lose student info

3. There is no any violation, all attributes contain atomic values

4.

PK → (StudentID, ProjectID, SupervisorID)

partial dependencies: StudentID → StudentName, StudentMajor; ProjectID → ProjectTitle, ProjectType; SupervisorID → SupervisorName, SupervisorDept.

2NF solution:

Student(StudentID, StudentName, StudentMajor)

Project(ProjectID, ProjectTitle, ProjectType)

Supervisor(SupervisorID, SupervisorName, SupervisorDept)

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

5. The resulting schema is in 3NF because every non-key attribute in each table is fully functionally dependent on its table's primary key, and there are no transitive dependencies.

Task 4.2: Advanced Normalization

Given Table:

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

Business Rules:

• Each student has exactly one major (StudentID)

• Each course has a fixed name (CourseID)

• Each instructor has exactly one name (InstructorID)

• Each time slot in a room determines the building (rooms are unique across campus) (TimeSlot, Room)

• Each course section is taught by one instructor at one time in one room (StudentID+ InstructorID+ TimeSlot, Room)

• A student can be enrolled in multiple course sections

Your Tasks: 1. Determine the primary key of this table (hint: this is tricky!) 2. List all functional dependencies 3. Check if the table is in BCNF 4. If not in BCNF, decompose it to BCNF showing your work 5. Explain any potential loss of information in your decomposition

1. PK: (StudentID, CourseID, InstructorID, TimeSlot, Room)

2. StudentID → StudentMajor

CourseID → CourseName

InstructorID → InstructorName

Room → Building

3.

StudentID → StudentMajor: StudentID only gives the student's major, not everything else, so it's not a superkey.

CourseID → CourseName: CourseID tells us the course name but not the rest of the row, so it also isn't a superkey.

InstructorID → InstructorName: this only gives the instructor's name, not the whole tuple, so again not a superkey.

(TimeSlot, Room) → Building: the pair TimeSlot and Room gives the building but doesn't identify all the data, so it's not a superkey either.

4.

Student(StudentID, StudentMajor)

Course(CourseID, CourseName)

Instructor(InstructorID, InstructorName)

RoomSchedule(TimeSlot, Room, Building)

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

5. When I split the original relation into smaller tables, I checked if any information could be lost. Because each new table keeps the right keys, I can join them back and get exactly the same rows as in the original table. So the decomposition is lossless — no data disappears, it's just stored in several tables instead of one big one.

Part 5: Design Challenge

Task 5.1: Real-World Application

Scenario: Your university wants to track student clubs and organizations with the following requirements:

System Requirements:

• Student clubs and organizations information

• Club membership (students can join multiple clubs, clubs have multiple members)

• Club events and student attendance tracking

• Club officer positions (president, treasurer, secretary, etc.)

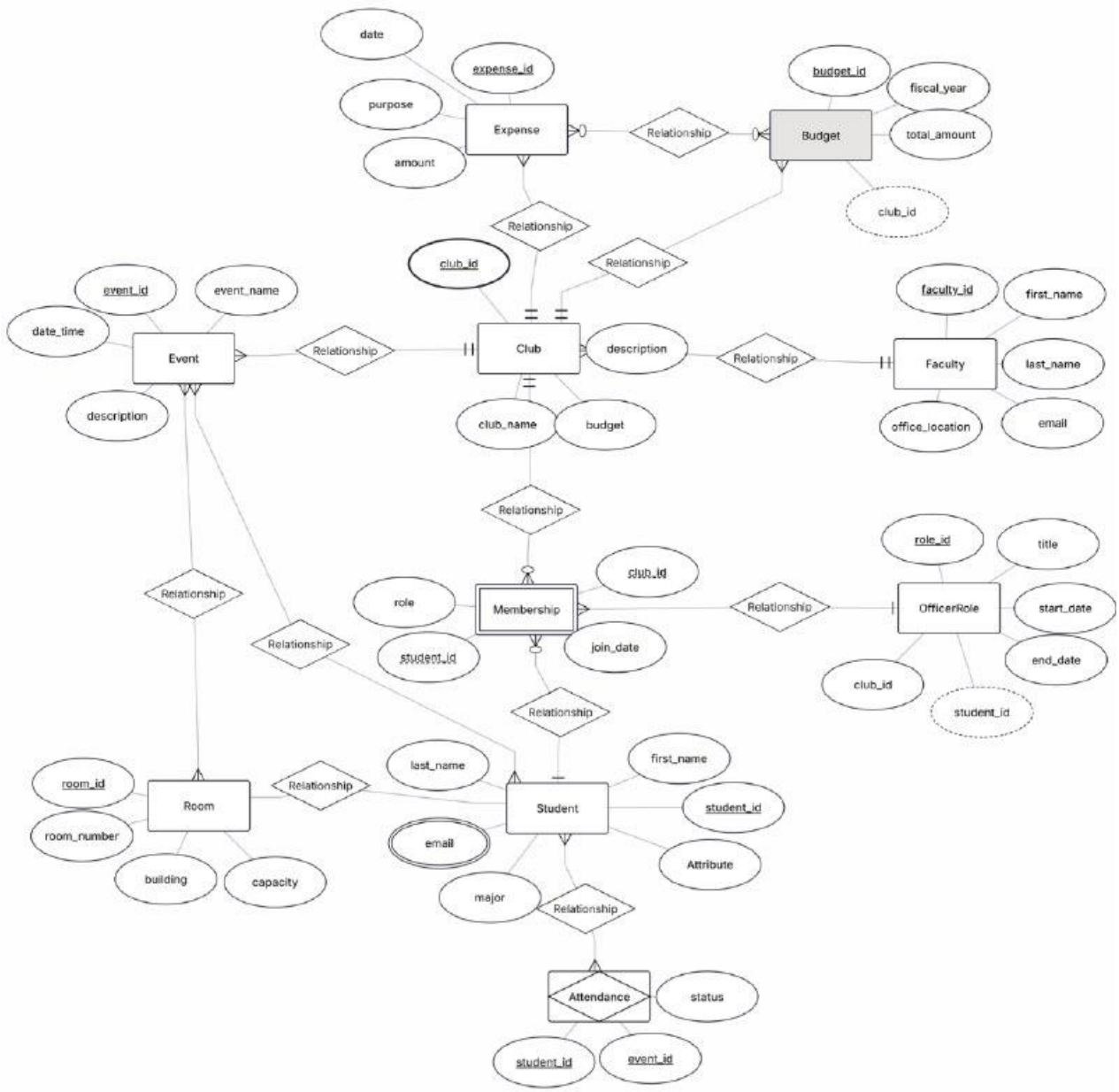
• Faculty advisors for clubs (each club has one advisor, faculty can advise multiple clubs)

• Room reservations for club events

• Club budget and expense tracking

Your Tasks: 1. Create a complete ER diagram for this system 2. Convert your ER diagram to a normalized relational schema 3. Identify at least one design decision where you had multiple valid options and explain your choice 4. Write 3 example queries that your database should support (in English, not SQL) Example Query Format: - “Find all students who are officers in the Computer Science Club” - “List all events scheduled for next week with their room reservations”

1.



2. Student(StudentID PK, Name, Major, Year)

Faculty(FacultyID PK, Name, Department)

Club(ClubID PK, ClubName, Budget, AdvisorID FK → Faculty(FacultyID))

Membership(StudentID FK → Student, ClubID FK → Club, JoinDate, PRIMARY KEY(StudentID, ClubID))

OfficerPosition(StudentID FK → Student, ClubID FK → Club, PositionName, StartDate, EndDate, PRIMARY KEY(StudentID, ClubID, PositionName))

Event(EventID PK, ClubID FK → Club, EventName, Date, Time, RoomID FK → Room)

EventAttendance(StudentID FK → Student, EventID FK → Event, PRIMARY KEY(StudentID, EventID))

Room(RoomID PK, Building, Capacity)

3. Decision: How to model Officer Positions.

Option 1: Include officer info in Membership table.

Option 2 (chosen): Create a separate OfficerPosition table.

Reason for choice:

A student may hold multiple positions over time, even in the same club.

Separating positions allows tracking historical positions without repeating membership info, reducing redundancy.

4. Example Queries

List the students holding officer positions in the Computer Science Club

Show all events happening next week along with the rooms they’re booked in

Display each club’s budget and expenses, and include the name of its advisor