Database Systems Laboratory Work

Week 2: Relational Model & Keys

Part 1: Key Identification Exercises

Task 1.1: Superkey and Candidate Key Analysis

Given the following relations, identify all superkeys and candidate keys:

Relation A: Employee

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

Sample Data:
```

EmpID	SSN	Email	Phone	Name	Department	Salary
101	123-45-6789	john@company.com	555-0101	John	IT	75000
102	987-65-4321	mary@company.com	555-0102	Mary	HR	68000
103	456-78-9123	bob@company.com	555-0103	Bob	IT	72000

Your 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.

ANSWER:

1. Superkeys:

   {EmpID}

   {SSN}

   {Email}

   {Phone}

   {EmpID, Name}

   {SSN, Name}

   {Email, Name}

   {Phone, Name}

2. {EmpID}, {SSN}, {Email}, {Phone}

3. EmpID. This key is unique, short and independent of external factors

4. No. Phone is a candidate key

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
```

Your 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)

ANSWER:

1. {StudentID, CourseCode, Section, Semester, Year}
2. StudentID – determines who is the student, without it only one student could register in one course

   CouseCode – defines the subject

   Section – defines which group

   Semester – because the same course can be chosen in different semesters

   Year - because the same course section and semester can repeat in different academic years

   Each attribute is necessary to ensure no duplicate registrations for the same course section in the same semester and year
3. They don't exist. There can be the same Grades and Credits

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)
```

Your Tasks: 1. Identify all foreign key relationships

ANSWER:

1. Student(AdviserID) and Professor(ProfID)
2. Student(StudentID) and Enrollment(StudentID)
3. Enrollment(CourseID) and Course(CourseID)
4. Professor(ProfID) and Department(ChairID)
5. Professor(Department) and Department(DeptCode)
6. Course(DepartmentCode) and Department(DeptCode)

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)

Your 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

Patients (patient_id, name, birthdate, address, phone_numbers, insurance_info)

Doctors (doctor_id, name, specializations, phone_numbers, location)

Departments (department_code, dept_name, location)

Appointments (appointment_id, patient_id, doctor_id, date_time, purpose, notes)

Prescriptions (prescription_id, doctor_id, patient_id, medication_name, dosage, instructions)

HospitalRooms (department_code, room_number)

ANSWER:

1. Patients strong

   Doctors strong

   Departments strong

   Appointments weak

   Prescriptions weak

   Hospital Rooms weak

2. Patients:

   - patient_id(simple, primary key)
   - name (simple)

- birthdate (composite)d
- address (composite)
- phone_numbers (milti-valued)
- insurance_info (simple)

Doctors:

- doctor_id (simple, primary key)
- name (simple)
- specializations (multi-valued)
- phone_numbers (multi-valued)
- location (composite)

Departments:

- department_code (simple, primary key)
- dept_name (simple)
- location (composite)

Appointments:

- appointment_id (simple, primary key)
- patient_id (simple)
- doctor_id (simple)
- date_time (composite)
- purpose (simple)
- notes (simple)

Prescriptions:

- prescription_id (simple, primary key)
- doctor_id (simple)
- patient_id (simple)
- medication_name (simple)
- dosage (simple)
- instructions (simple)

Hospital Rooms:

- department_code (simple)
- room_number (simple)

3. Patients(1):Appointments(M)

   Doctors(1):Appointments(M)

   Patients(1):Prescriptions(M)

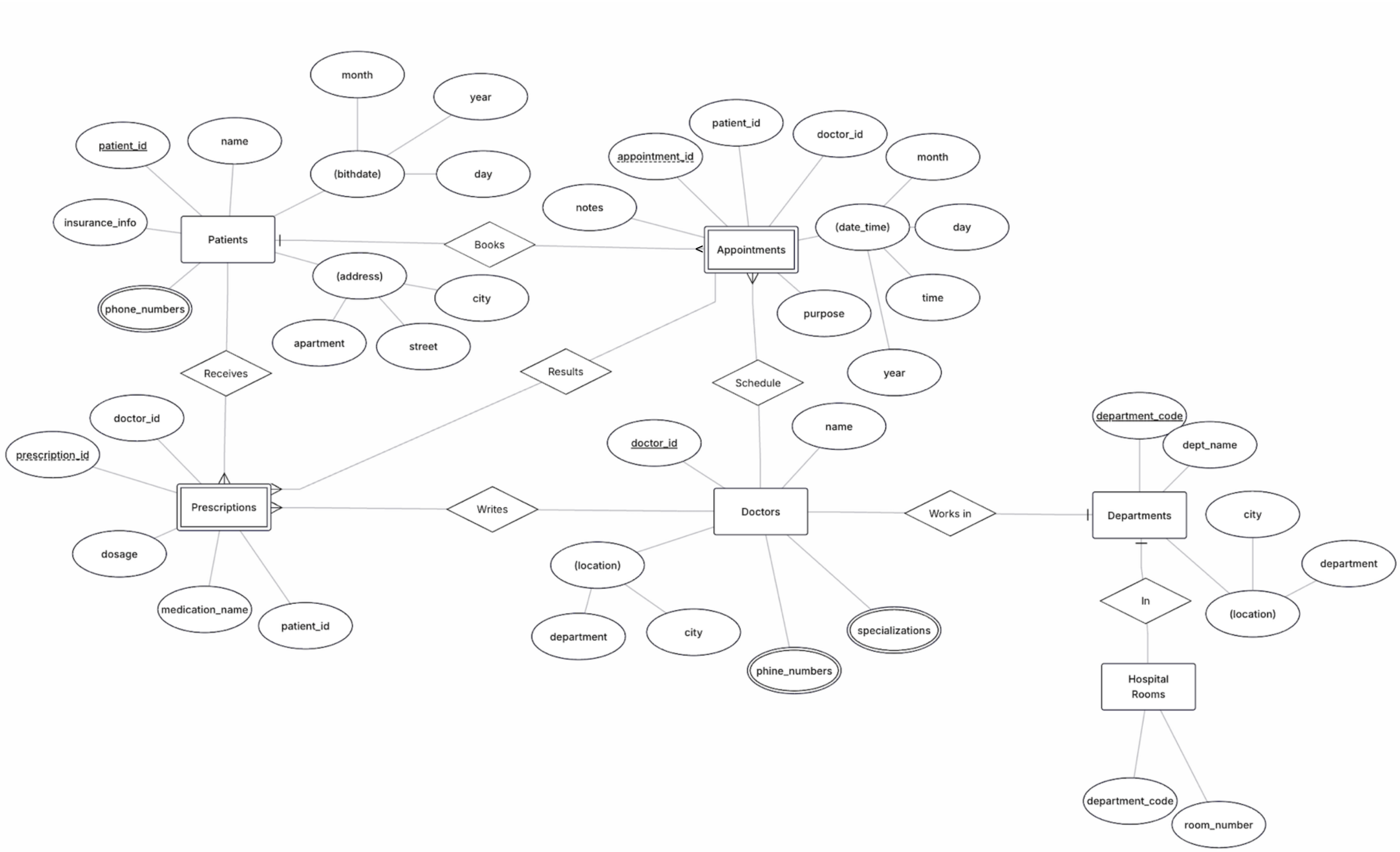
   Doctors(1):Prescriptions(M)

   Departments(1):Doctors(M)

   Appointments(1):Prescriptions(M)

   Hospital Rooms(M):Departments(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

Your 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

Customers (customer_id, address, email)

Address (address_id, customer_id, city, street, apartment)

Orders (order_id, customer_id, address, date)

Order_item (order_item_id, order_id, product_id, quantity, price)

Products (product_id, name, price, category_id, vendor_id, inventory_leve)

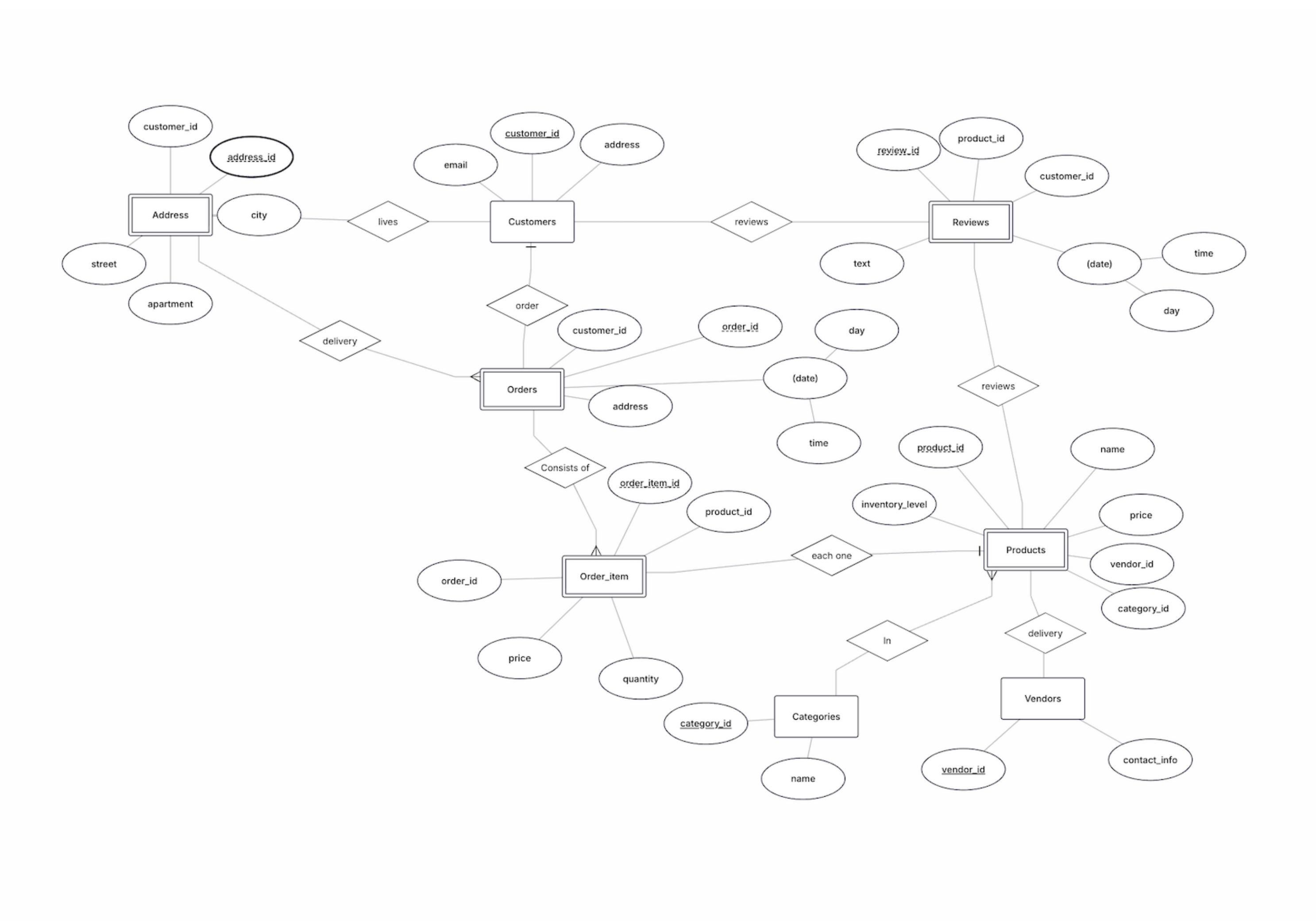
Categories (category_id, name)

Vendors (vendor_id, contact_info)

Reviews (review_id, product_id, customer_id, date, text)

ANSWER:

1.



2. Address. Because it relies on Customer (customer_id) for unique identification and existence; cannot be uniquely identified by address_id alone without customer context.

3. the many-to-many relationship between Orders and Products needs attributes and is represented by the Order_item entity with its own attributes like quantity and price

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 table schemas

ANSWER:

1. StudentID → StudentName, StudentMajor

   ProjectID → ProjectTitle, ProjectType, SupervisorID, StartDate, EndDate

   SupervisorID → SupervisorName, SupervisorDept

   (StudentID, ProjectID) → Role, HoursWorked

2. Redundancy exists

   StudentName and StudentMajor repeat for each project student does

   ProjectTitle, ProjectType, SupervisorID, SupervisorName, and SupervisorDept repeat for each student involved in a project

   Update anomaly – if the supervisor`s name changes, each row need to be updated

   Insert anomaly – cannot add a new student unless project is assigned (same vise versa)

   Delete anomaly – deleting a student's only project deletes the student information as well or if we delete a student project may be deleted too

3. The table is already in 1NF as all attributes are atomic, no repeating groups or arrays present

4. Primary keys: StudentID, ProjectID

   Partial dependencies exist: StudentID functionally determines StudentName, StudentMajor; ProjectID determines ProjectTitle, ProjectType, SupervisorID

   Tables:

   Student (<u>`StudentID`</u>`, StudentName, StudentMajor`)

   Project (<u>`ProjectID`</u>`, ProjectTitle, ProjectType, SupervisorID`)

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

5. Transitive Dependency: ProjectID → SupervisorID → SupervisorName, SupervisorDept

   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

Given Table:

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

Business Rules:

- Each student has exactly one major
- Each course has a fixed name
- Each instructor has exactly one name
- Each time slot in a room determines the building (rooms are unique across campus)
- Each course section is taught by one instructor at one time in one 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

ANSWER:

1. Primary key is composite (StudentID, TimeSlot, Room). Because each student can attend different courses and each course is identified by timeslot and room
2. StudentID → StudentMajor
   CourseID → CourseName
   InstructorID → InstructorName
   (TimeSlot, Room) → Building
   (CourseID, TimeSlot, Room) → InstuctorID
3. No, the dependency StudentID → StudentMajor violates BCNF because StudentID alone is not a superkey. Similarly, dependencies CourseID → CourseName, InstructorID → InstructorName, and (TimeSlot, Room) → Building violate BCNF for the same reason.
4. Tables:
   Student(StudentID, StudentMajor)
   Course(CourseID, CourseName)
   Instructor(InstructorID, InstructorName)
   Room(Room, Building)
   Section(Room, TimeSlot, CourseID, InstructorID)
   Enrollment(StudentID, Room, TimeSlot)
5. No information loss

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

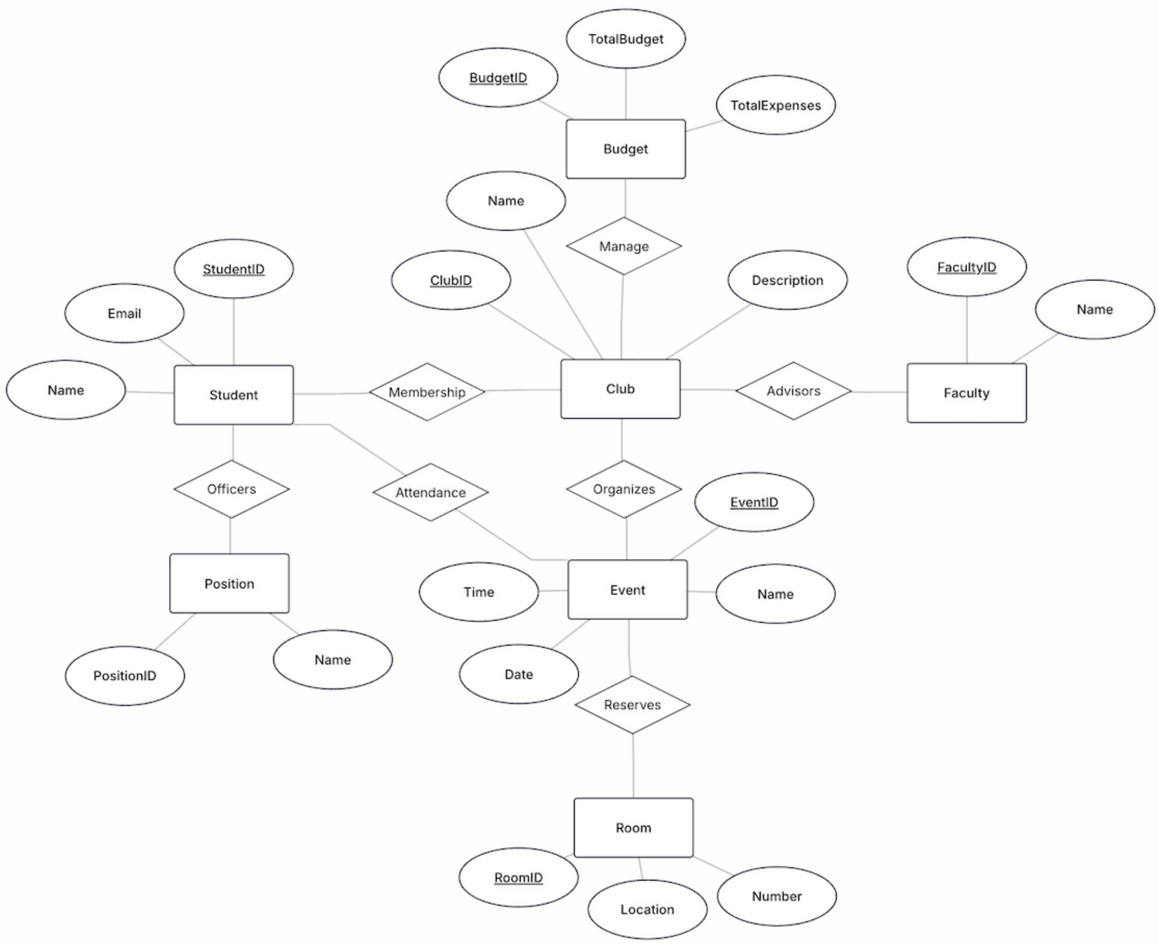
Student (StudentID, Name, Email)
Club (ClubID, Name, Description)
Event (EventID, Name, Date, Time)
Faculty (FacultyID, Name)
Room (RoomID, Number, Location)
Position (PositionID, Name)
Budget (BudgetID, TotalBudget, TotalExpenses)

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"

ANSWER:

1



2

Student(StudentID, Name, Email)

Club(ClubID, Name, Description, FacultyID)

Faculty(FacultyID, Name)

Room(RoomID, Number, Location)

Position(PositionID, Name)

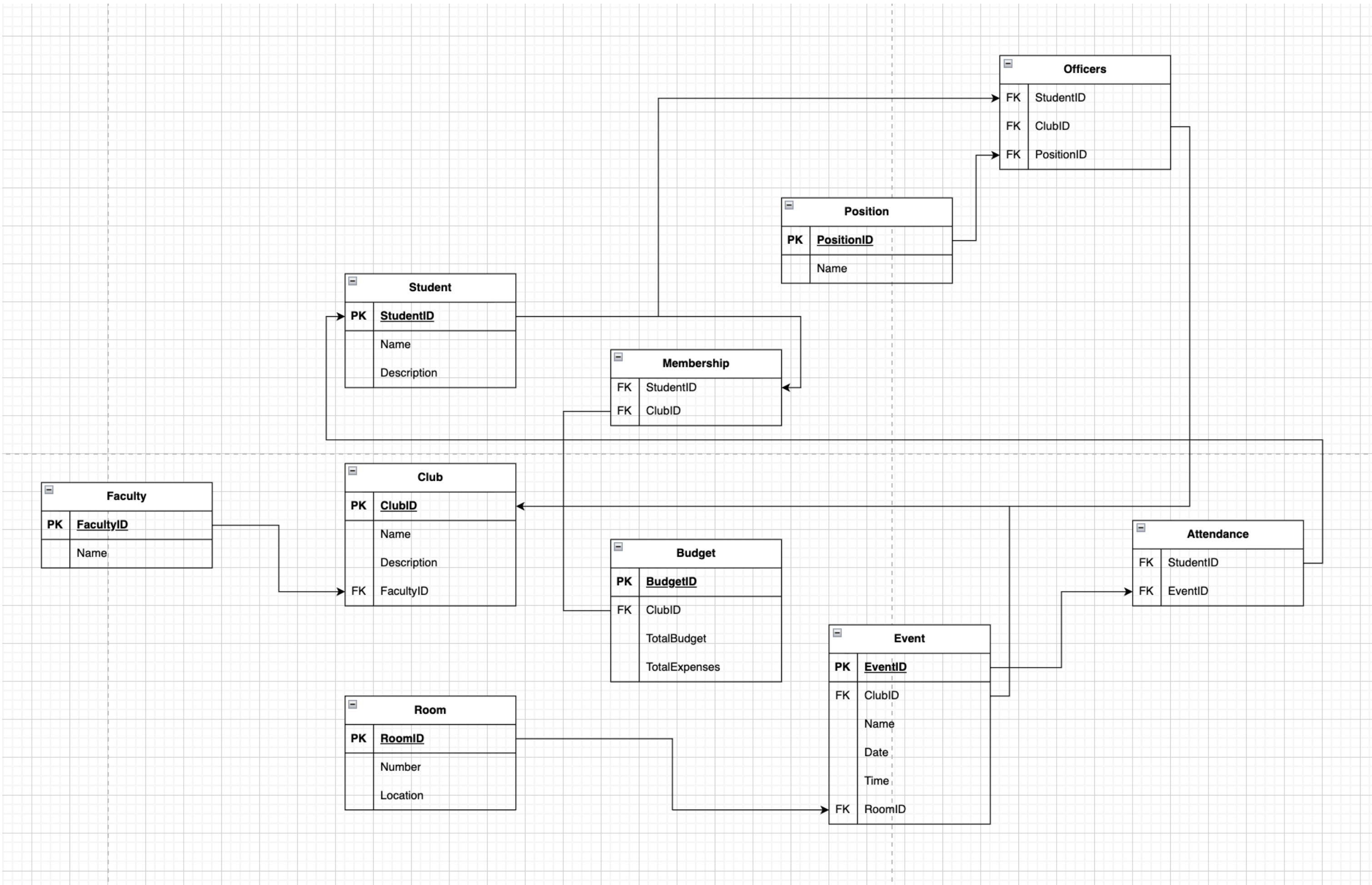
Budget(BudgetID, ClubID, TotalBudget, TotalExpenses)

Event(EventID, ClubID, Name, Date, Time, RoomID)

Membership(StudentID, ClubID)

Attendance(StudentID, EventID)

Officers(StudentID, ClubID, PositionID)



3 The main design decision was to model the Officers relationship as a ternary relation (Student-Club-Position) rather than separate entities or multiple binary relationships. This choice cleanly represents that a student can hold a specific position in a specific club, avoiding redundancy and allowing easy querying of officer roles. Alternative approaches like adding a position attribute inside Membership would be less flexible if a student could hold multiple officer roles or change roles independently.

4

- Find all students who are officers in the Computer Science Club.
- List all events scheduled for 12.10 along with their room reservations.
- Show the total budget and expenses for each club and identify clubs that are overspending.

Lab Deliverables

What to Submit:

1. **Complete Solutions:** Detailed answers to all tasks with clear explanations
2. **ER Diagrams:** Hand-drawn or digital diagrams for Tasks 2.1, 2.2, and 5.1
3. **Normalization Work:** Step-by-step decomposition showing all intermediate steps
4. **Relational Schemas:** Complete table definitions with primary keys, foreign keys, and data types

Submission Format:

- **File Format:** PDF document with clear headings for each task
- **Diagrams:** Include all ER diagrams (scanned hand-drawings or digital files)
- **Tables:** Use clear formatting for relational schemas
- **Page Limit:** No strict limit, but aim for clarity and conciseness

Common Mistakes to Avoid:

- Forgetting to underline primary keys in relational schemas
- Missing foreign key relationships between tables
- Stopping normalization at 2NF instead of continuing to 3NF
- ER diagrams missing cardinality constraints or participation indicators
- Weak entities without proper composite keys including owner's key
- Confusing superkeys with candidate keys

Additional Resources:

- Database textbook chapters on ER modeling and normalization
- Online ER diagram tools: Draw.io, Lucidchart, ERDPlus
- Course slides and lecture notes

Good luck! This laboratory work will provide you with hands-on experience in the fundamental skills every database designer needs. Take your time to understand each concept thoroughly, as these form the foundation for all advanced database topics