**Database Systems Laboratory Work — Complete Solutions**

Generated on: 2025-09-15 03:17

This document contains detailed, step-by-step solutions to every task from LabWork1 (Week 2: Relational Model & Keys, ER Modeling, and Normalization). Each task includes explanations, tables, and schematic ER‑diagram suggestions.

**Part 1: Key Identification Exercises**

**Task 1.1: Superkey and Candidate Key Analysis**

**Relation A: Employee(EmpID, SSN, Email, Phone, Name, Department, Salary)**

Assumptions based on common practice: EmpID, SSN, and Email are unique attributes; Phone may or may not be unique across employees in real life.

**1) Examples of Superkeys (≥6)**

* {EmpID}
* {SSN}
* {Email}
* {EmpID, SSN}
* {EmpID, Email}
* {SSN, Email}
* {EmpID, SSN, Email}
* {EmpID, Phone} (superset of a key, hence a superkey)

**2) Candidate Keys**

Likely candidate keys are single-attribute: {EmpID}, {SSN}, {Email}.

**3) Recommended Primary Key and Rationale**

Choose {EmpID} as the primary key. It is system-assigned, stable, and avoids exposing sensitive personal data (SSN) or mutable contact info (Email).

**4) Can two employees share the same phone number?**

Yes, it is possible (shared family/business lines or temporary numbers). The sample data shows unique phones, but nothing enforces uniqueness unless explicitly constrained.

**Relation B: Registration(StudentID, CourseCode, Section, Semester, Year, Grade, Credits)**

Business Rules recap: A student can retake a course in different semesters; cannot register for the same course section in the same semester; each course section in a semester has a fixed credit value.

**1) Minimal Primary Key**

Minimal composite key: {StudentID, CourseCode, Section, Semester, Year}. This uniquely identifies a student's registration to a specific section in a specific term.

**2) Why each attribute is necessary**

* StudentID — identifies the student.
* CourseCode — identifies the course.
* Section — distinguishes parallel offerings of the same course that term.
* Semester + Year — distinguishes the same course/section across different academic terms.

**3) Additional Candidate Keys (if any)**

Without a surrogate RegistrationID, there are no alternative minimal keys. (Order of attributes does not create a new candidate key.)

**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).

**Foreign Key Relationships**

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

Optional: If a separate Term table exists, Enrollment.Semester (and/or Year) would reference it.

**Part 2: ER Diagram Construction**

**Task 2.1: Hospital Management System — ER Design**

**Entities (Strong/Weak)**

* Patient (Strong): PatientID (PK), Name, Birthdate, Address, Insurance
* Doctor (Strong): DoctorID (PK), Name, Phone, OfficeLocation
* Department (Strong): DeptCode (PK), DeptName, Location
* Room (Weak): (DeptCode, RoomNumber) as composite PK; depends on Department
* Appointment (Associative): AppointmentID (PK) or (PatientID, DoctorID, DateTime) as identifying composite; Purpose, Notes
* Prescription (Associative): PrescriptionID (PK) or composite identifying (PatientID, DoctorID, DatePrescribed); Medication, Dosage, Instructions

**Attributes Classification**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity** | **Simple** | **Composite** | **Multi-valued** | **Derived** |
| Patient | PatientID, Name, Birthdate, Insurance | Address(street, city, state, zip) | PhoneNumbers | Age (from Birthdate) |
| Doctor | DoctorID, Name, Phone |  | Specializations |  |
| Department | DeptCode, DeptName, Location |  |  |  |
| Room | RoomNumber |  |  |  |
| Appointment | Purpose, Notes, DateTime |  |  |  |
| Prescription | Medication, Dosage, Instructions, DatePrescribed |  |  |  |

**Relationships & Cardinalities**

* Department 1 — N Room (weak entity Room identified by DeptCode + RoomNumber)
* Doctor N — 1 Department (each doctor belongs to one department; department has many doctors)
* Patient M — N Doctor via Appointment (Appointment has attributes: DateTime, Purpose, Notes)
* Patient M — N Doctor via Prescription (Prescription has attributes: Medication, Dosage, Instructions, DatePrescribed)

**ER Diagram (schematic suggestions)**

* Use double rectangle for Room (weak) with identifying relationship to Department.
* Model Specializations and Patient.PhoneNumbers as multivalued (double ovals).
* For Appointment/Prescription, use associative entities (diamonds turned into rectangles with own attributes).
* Underline primary keys: PatientID, DoctorID, DeptCode; composite for Room: (DeptCode, RoomNumber).

**Task 2.2: E‑commerce Platform — ER Design**

**Entities & Relationships**

* Customer(CustomerID, Name, Email, BillingAddressID, ... )
* Address(AddressID, Street, City, State, Zip, Country) — customers can have multiple addresses (billing/shipping).
* Order(OrderID, CustomerID, OrderDate, Status, BillingAddressID, ShippingAddressID)
* OrderItem( OrderID, ProductID, Quantity, PriceAtOrder ) — associative (weak) entity for M:N between Order and Product.
* Product(ProductID, Name, Description, Price, CategoryID, VendorID, ... )
* Category(CategoryID, CategoryName)
* Vendor(VendorID, VendorName, ContactInfo, ... )
* Inventory(ProductID, StockQty, ReorderLevel, WarehouseLocation)
* Review(ReviewID, ProductID, CustomerID, Rating, Comment, ReviewDate)

**Weak Entity Justification**

OrderItem is weak because its identity depends on Order and Product (composite PK: {OrderID, ProductID} or with LineNo).

**M:N Relationship Needing Attributes**

Order–Product is M:N; the intersection needs Quantity and PriceAtOrder, so it must be modeled as OrderItem with its own attributes.

**ER Diagram (schematic suggestions)**

* Customer 1—N Order; Order 1—N OrderItem; Product 1—N OrderItem.
* Product N—1 Category; Product N—1 Vendor.
* Product 1—N Review (each review by one customer; optionally enforce (CustomerID, ProductID) uniqueness to prevent duplicates).
* Product 1—1 Inventory (or 1—N if multi‑warehouse).
* Order → Address (billing/shipping) via two FKs.

**Part 4: Normalization Workshop**

**Task 4.1: Denormalized Table Analysis — StudentProject**

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

**1) Functional Dependencies (FDs)**

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

**2) Problems & Anomalies**

* Redundancy: StudentName/Major repeated per project; ProjectTitle/Type and Supervisor data repeated per student assigned.
* Update anomaly: Changing SupervisorName requires updating many rows for the same SupervisorID.
* Insert anomaly: Cannot insert a new project with supervisor info until at least one student is assigned.
* Delete anomaly: Deleting the last student on a project loses the project and supervisor info.

**3) 1NF**

All attributes appear atomic; no repeating groups. Ensure dates are single-valued and types are consistent.

**4) 2NF**

Assuming the natural key is (StudentID, ProjectID) for an assignment row:

* Partial dependencies exist: StudentID → StudentName, StudentMajor; ProjectID → ProjectTitle, ProjectType, SupervisorID.
* Decompose to eliminate partial dependencies:

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

**5) 3NF**

Transitive dependency: ProjectID → SupervisorID and SupervisorID → SupervisorName, SupervisorDept. Our decomposition already isolates Supervisor, removing the transitive dependency from Project.

Final 3NF schema (keys underlined):

|  |  |
| --- | --- |
| **Relation** | **Schema** |
| Student | ⟂StudentID⟂, StudentName, StudentMajor |
| Supervisor | ⟂SupervisorID⟂, SupervisorName, SupervisorDept |
| Project | ⟂ProjectID⟂, ProjectTitle, ProjectType, SupervisorID (FK→Supervisor) |
| StudentProject | ⟂StudentID⟂, ⟂ProjectID⟂, Role, HoursWorked, StartDate, EndDate (FKs→Student, Project) |

**Task 4.2: Advanced Normalization — CourseSchedule**

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

**Business Rule Implications**

* Student has exactly one major ⇒ StudentID → StudentMajor.
* Course has a fixed name ⇒ CourseID → CourseName.
* Instructor has exactly one name ⇒ InstructorID → InstructorName.
* Room determines building (rooms are unique campus‑wide) ⇒ Room → Building.
* Each course section is taught by one instructor at one time in one room ⇒ (CourseID, TimeSlot, Room) → InstructorID.

**1) Primary Key (row semantics)**

A row represents a student's enrollment in a specific course section (identified by CourseID, TimeSlot, Room).

Thus a minimal key is: {StudentID, CourseID, TimeSlot, Room}.

**2) Functional Dependencies**

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

**3) BCNF Check**

BCNF requires every determinant to be a superkey. Violations: StudentID→StudentMajor, CourseID→CourseName, InstructorID→InstructorName, Room→Building.

**4) Decomposition to BCNF**

|  |  |  |
| --- | --- | --- |
| **Table** | **Attributes** | **Key / Notes** |
| Student | StudentID, StudentMajor | Key: StudentID |
| Course | CourseID, CourseName | Key: CourseID |
| Instructor | InstructorID, InstructorName | Key: InstructorID |
| Room | Room, Building | Key: Room |
| Section | CourseID, TimeSlot, Room, InstructorID | Key: (CourseID, TimeSlot, Room); FKs→Course, Room, Instructor |
| Enrollment | StudentID, CourseID, TimeSlot, Room | Key: (StudentID, CourseID, TimeSlot, Room); FKs→Student, Section |

**5) Lossless Join & Dependency Preservation**

The decomposition is lossless because Enrollment references Section via the section key, and Section references Course/Room/Instructor via their keys; natural joins reconstruct original rows.

Most dependencies are preserved in their respective relations (Student, Course, Instructor, Room).

**Part 5: Design Challenge — Student Clubs System**

**Task 5.1: ER & Relational Design**

**ER Scope & Entities**

* Student(StudentID, Name, Email, Major, Year)
* Club(ClubID, ClubName, Description, Budget)
* Membership(StudentID, ClubID, JoinDate, Status, DuesPaid) — associative (weak) entity
* OfficerPosition(PositionID, PositionName) (reference of possible roles)
* ClubOfficer(ClubID, StudentID, PositionID, StartDate, EndDate) — who holds which position when
* Event(EventID, ClubID, Title, StartDateTime, EndDateTime, Purpose)
* Attendance(EventID, StudentID, CheckedInAt, Notes) — associative entity with attributes
* Faculty(ProfID, Name, Department, Email)
* Advisor(ClubID, ProfID, StartDate, EndDate) — club ↔ faculty relationship
* Room(RoomID, Building, Capacity)
* Reservation(ReservationID, EventID, RoomID, ReservedStart, ReservedEnd, ApprovedBy)
* Expense(ExpenseID, ClubID, Amount, Category, SpentOn, Notes)

**Key Relationships & Cardinalities**

* Student M—N Club via Membership.
* Club 1—N Event; Event M—N Student via Attendance.
* Club 1—N ClubOfficer; constrained by (ClubID, PositionID, date ranges) for exclusivity per period.
* Club 1—1 Advisor (each club has exactly one active advisor at a time); Faculty 1—N Advisor across clubs.
* Event 1—1 Reservation (or 1—N if multiple rooms/time blocks).
* Reservation N—1 Room.

**Normalized Relational Schema (PKs underlined, FKs noted)**

|  |  |
| --- | --- |
| **Relation** | **Attributes** |
| Student | ⟂StudentID⟂, Name, Email, Major, Year |
| Club | ⟂ClubID⟂, ClubName, Description, Budget |
| Membership | ⟂StudentID⟂, ⟂ClubID⟂, JoinDate, Status, DuesPaid (FKs→Student, Club) |
| OfficerPosition | ⟂PositionID⟂, PositionName |
| ClubOfficer | ⟂ClubID⟂, ⟂StudentID⟂, ⟂PositionID⟂, StartDate, EndDate (FKs→Club, Student, OfficerPosition) |
| Faculty | ⟂ProfID⟂, Name, Department, Email |
| Advisor | ⟂ClubID⟂, ⟂ProfID⟂, StartDate, EndDate (FKs→Club, Faculty) |
| Event | ⟂EventID⟂, ClubID(FK→Club), Title, StartDateTime, EndDateTime, Purpose |
| Attendance | ⟂EventID⟂, ⟂StudentID⟂, CheckedInAt, Notes (FKs→Event, Student) |
| Room | ⟂RoomID⟂, Building, Capacity |
| Reservation | ⟂ReservationID⟂, EventID(FK→Event), RoomID(FK→Room), ReservedStart, ReservedEnd, ApprovedBy |
| Expense | ⟂ExpenseID⟂, ClubID(FK→Club), Amount, Category, SpentOn, Notes |

**Design Choice with Alternatives**

Officer modeling: Either (A) embed OfficerRole as an attribute inside Membership (simpler, but loses history and multi‑role support), or (B) create ClubOfficer as a separate relation with date ranges (chosen), enabling role history, multiple roles per student, and integrity constraints per period.

**Example English Queries**

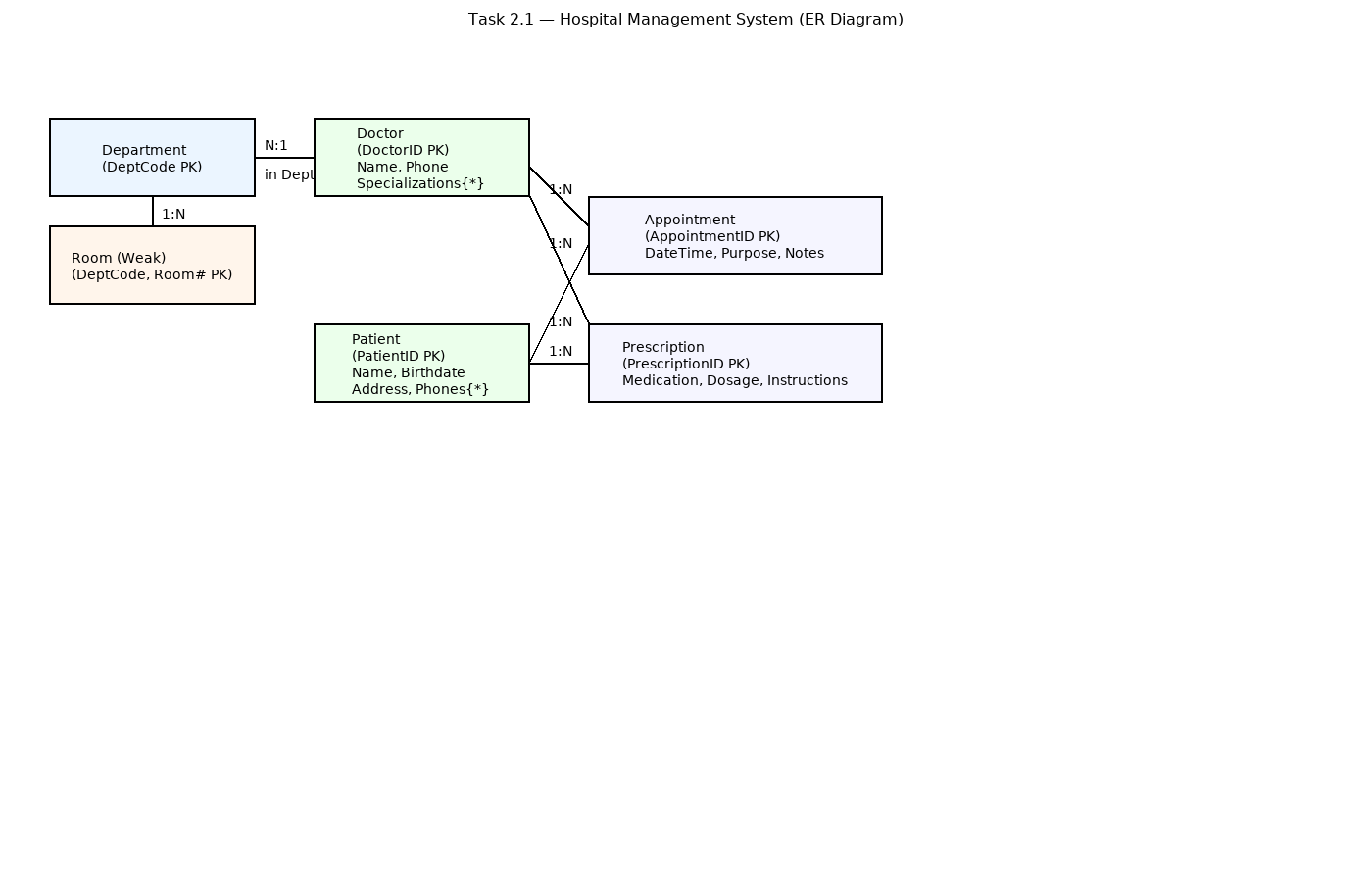
1. Find all students who are officers in the Computer Science Club this semester.
2. List all events scheduled for next week with their room reservations (room and building).
3. Show clubs where total expenses this month exceed their monthly budget allocation.

**ER‑Diagramming Tips (Schematic)**

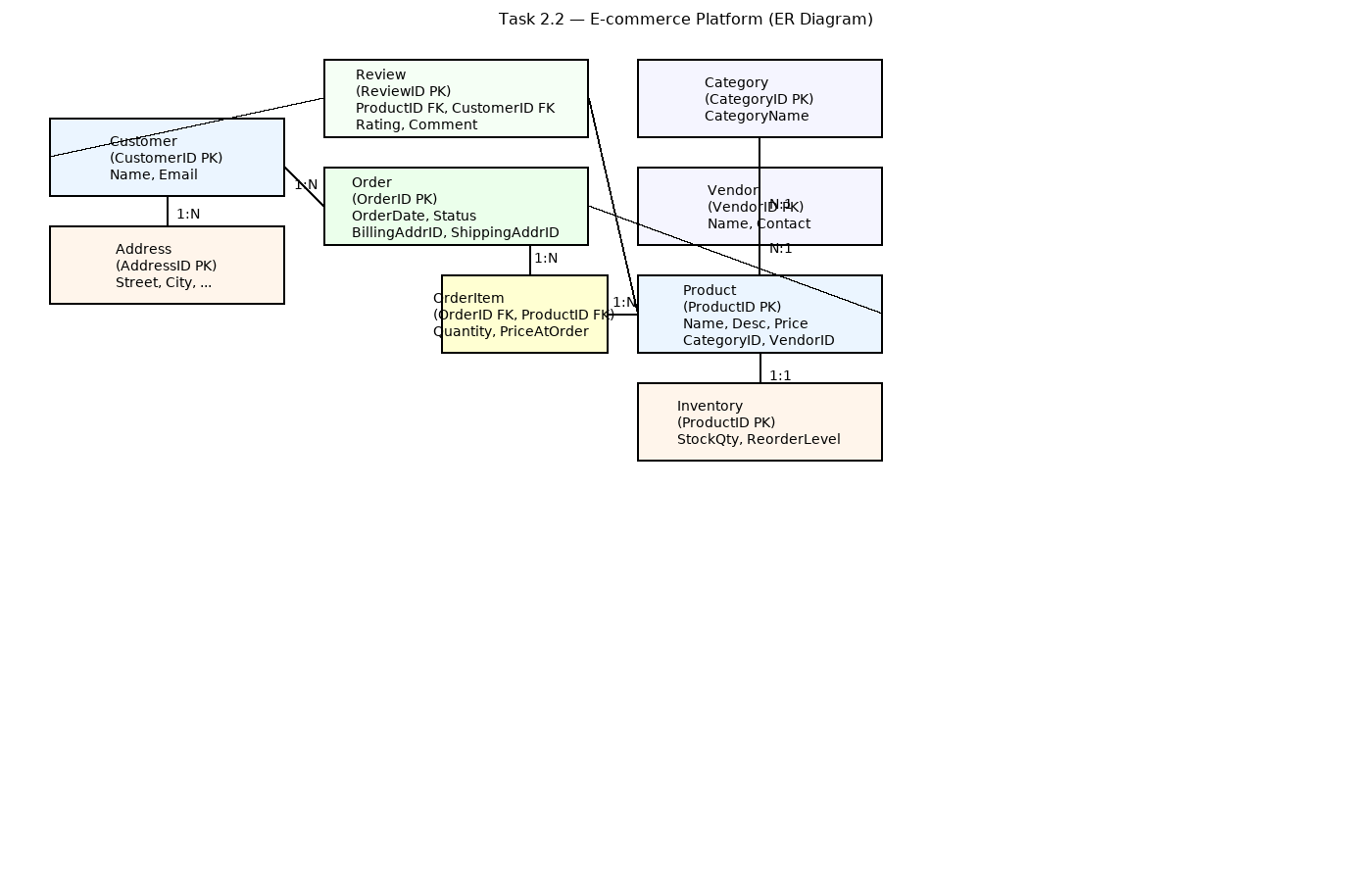
* Underline primary keys; mark weak entities with double rectangles and identifying relationships with double diamonds.
* Use crow’s foot or (1, N, M) markers for cardinalities; annotate participation (total/partial) if your notation supports it.
* Promote multivalued attributes to separate entities when they require additional properties or querying.
* When many‑to‑many relationships need attributes, convert them into associative entities with their own keys.

End of solutions.

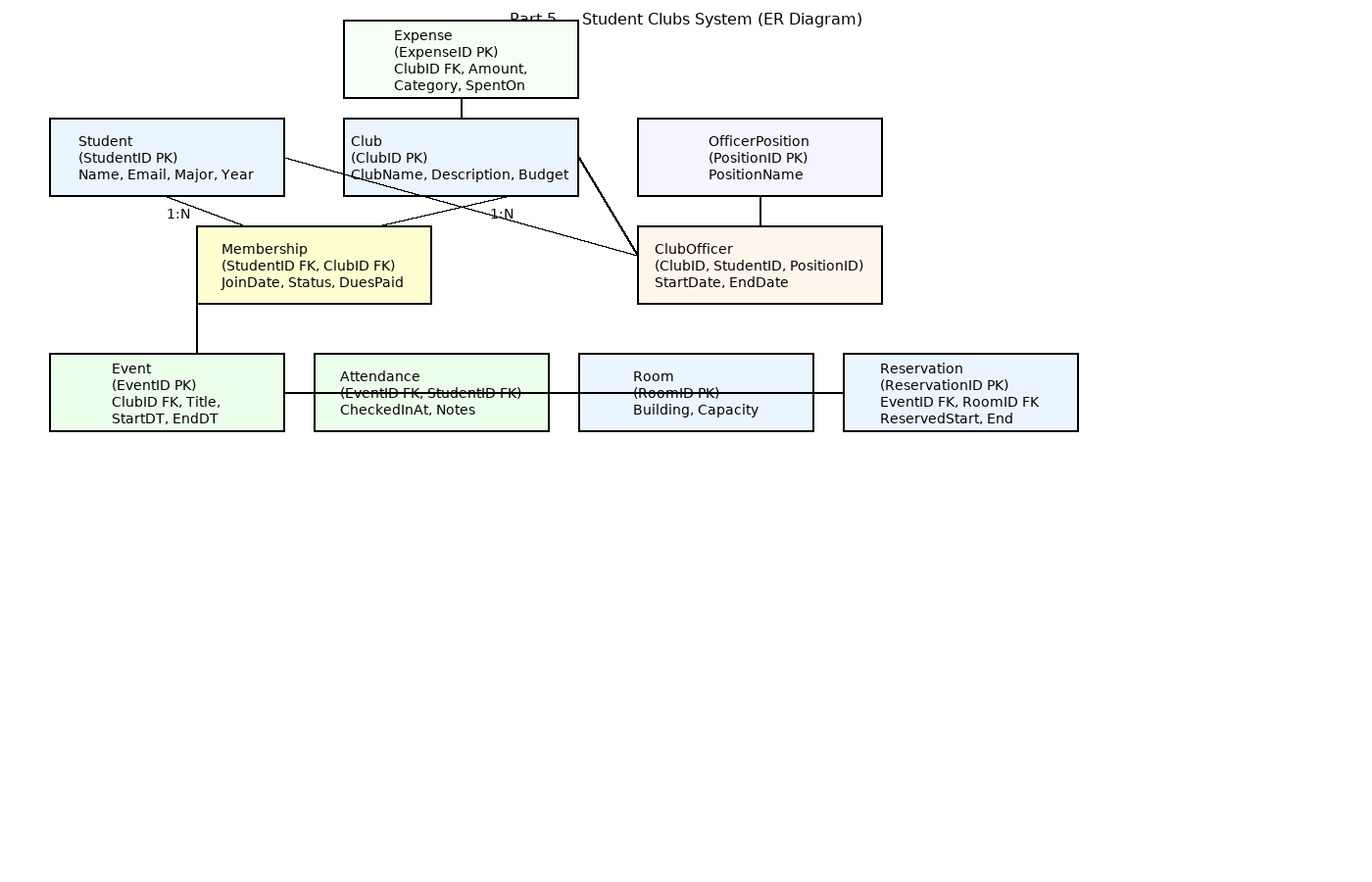
**ER Diagrams — Visual Figures**



*Figure 1: Hospital Management System — ER Diagram (Task 2.1)*



*Figure 2: E‑commerce Platform — ER Diagram (Task 2.2)*



*Figure 3: Student Clubs System — ER Diagram (Part 5.1)*