Part 1

Task 1.1

1. {EmpID}, {SSN}, {Email}, {Phone}, {EmpID, Name}, {Email, Name}
2. EmpID, SSN, Email, Phone
3. EmpID, because it's acting as a unique order number of each employee
4. No, because each employee possesses its own phone number.

1. StudentID, CourseCode, Section, Semester, Year
2. **StudentID** is necessary for identifying the exact student
CourseCode shows which course the student took
Section - different sections in the same course
Semester is the exact part of the year of taking course
Year - because student can take courses in every year
3. There is no additional candidate key that identify all attributes in one.

Task 1.2

Student.Major → Department.DeptCode

Student.AdvisorID → Professor.ProfID

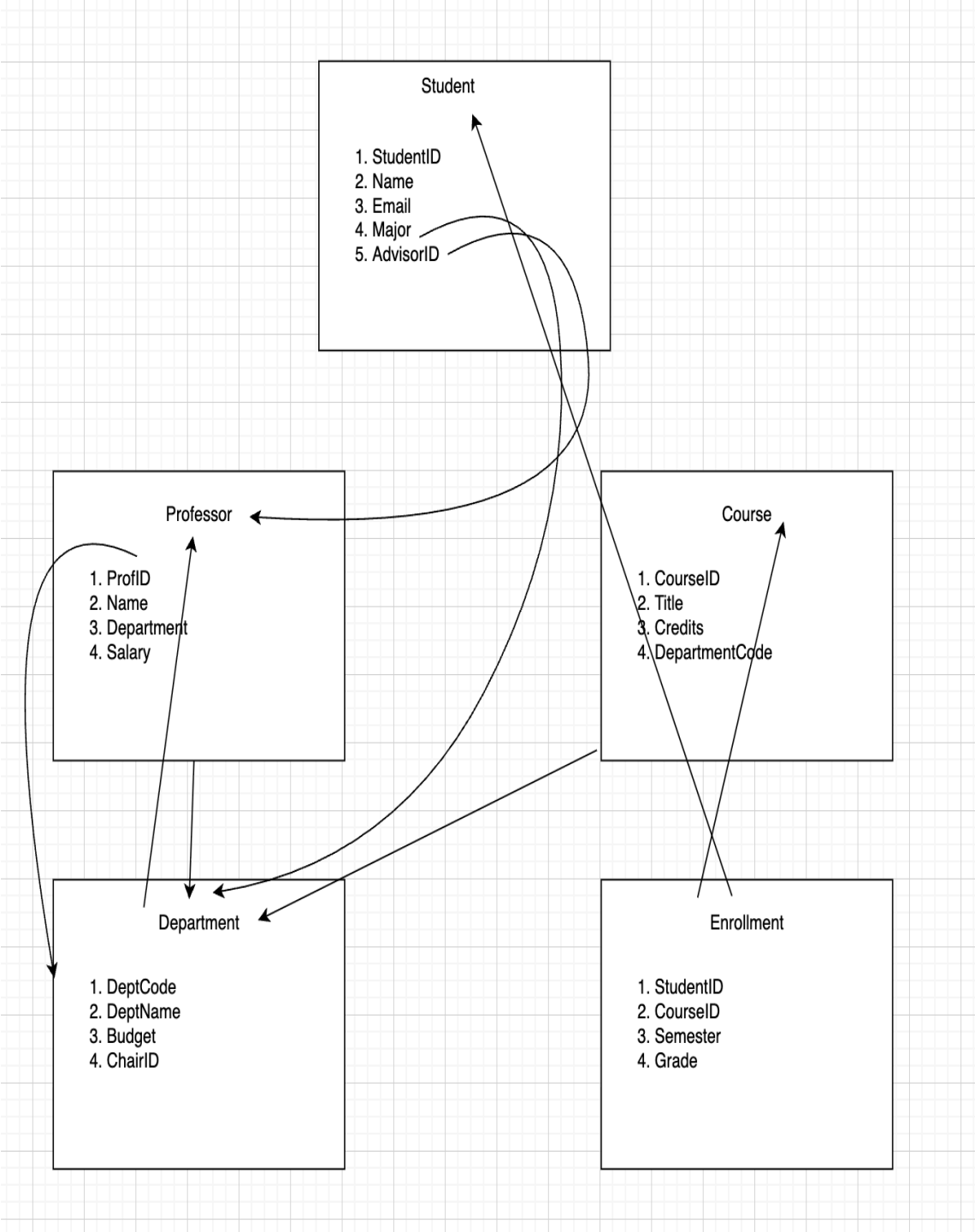
Professor.Department → Department.DeptCode

Course.DepartmentCode → Department.DeptCode

Department.ChairID → Professor.ProfID

Enrollment.StudentID → Student.StudentID

Enrollment.CourseID → Course.CourseID



Task 2.1

1. Entities

Patient (strong)

Doctor (strong)

Department (strong)

Appointment (weak**)**

Prescription (weak)

HospitalRoom (weak)

PatientPhone(weak)

DoctorPhone(weak)

2. Attributes

Patient
PatientID (PK) — simple
Name — composite
Birthdate — composite
Address — composite (Street, City, State, Zip)
PhoneNumbers — multi-valued
InsuranceInfo — simple

Doctor
DoctorID (PK) — simple
Name — composite:
Specializations — multi-valued
PhoneNumbers — multi-valued
OfficeLocation — composite

Department
DepartmentCode (PK) — simple
Name — composite
Location — simple

Appointment
AppointmentID (PK) — simple
DateTime — composite
Purpose — simple
Notes — simple
PatientID (FK), simple
DoctorID (FK), simple

Prescription
PrescriptionID (PK) — simple
Medication — simple
Dosage — simple
Instructions — simple
AppointmentID (FK), simple

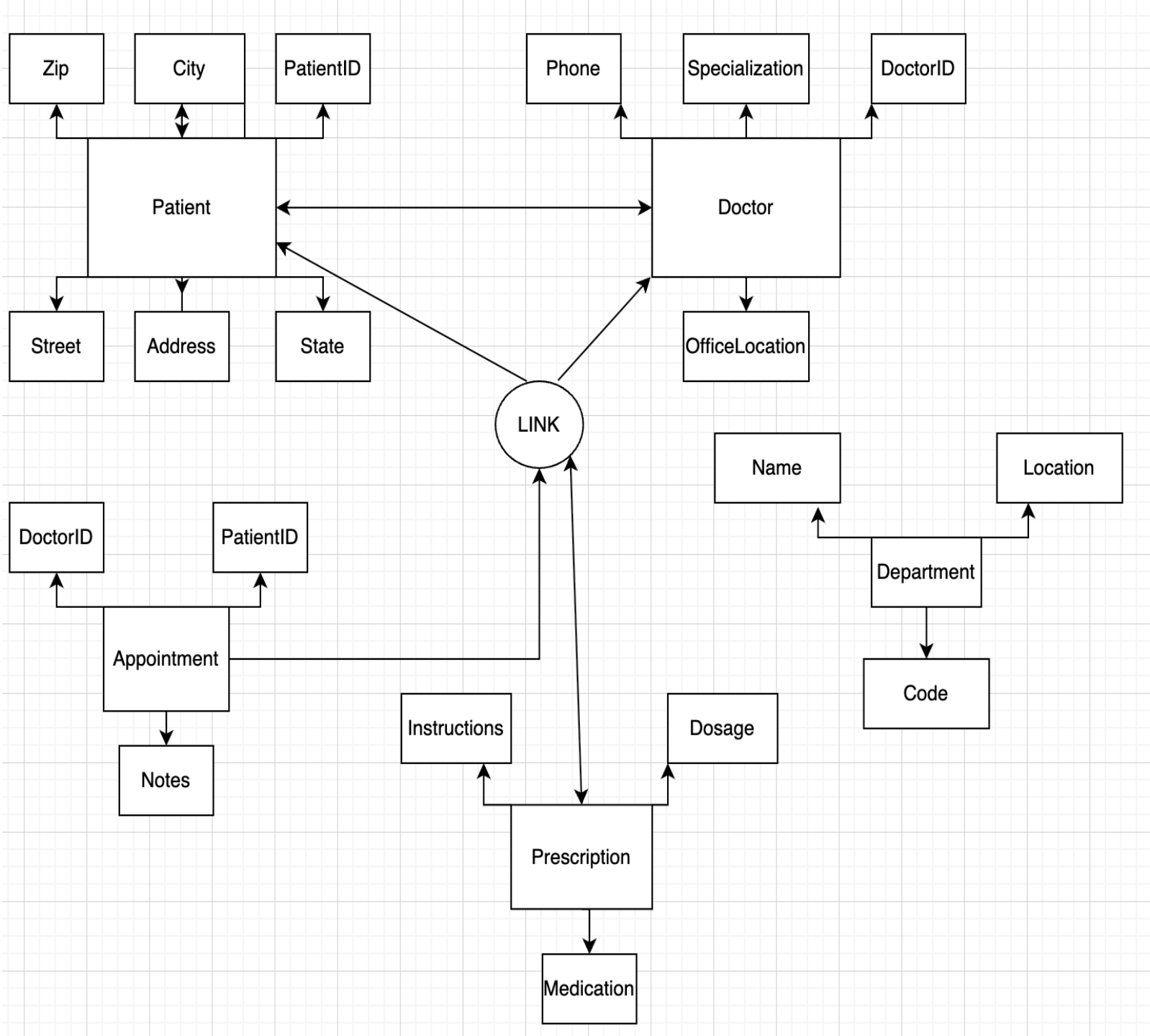
HospitalRoom
RoomNumber (simple)
DepartmentCode (simple)
RoomType (simple)

3. Relationships & Cardinalities

One-to-Many	Many-to-Many
Patient —> Appointment – a patient can have several appointment, there is no limit for this case, when each appointment is connected with exactly one patient	Doctor —> Specialization - a Doctor can have multiple Specializations, and a specialization can belong to many doctors
Doctor —> Appointment – the same as with patient	Prescription —> Medication - a prescription can contain multiple medications but a medication can appear on many prescriptions
Department —> Doctor – each doctor has his own department and each department contain many doctors	
Department —> Room – each room belongs to exactly one department, when each department have many rooms	
Patient —> Prescription - each patient can have many prescription	
Doctor —> Prescription – a doctor can write many prescriptions but each prescription is written by one doctor	
Patient — >PatientPhone - one patient can have several phone numbers, ex: mobile number, work number, home number and etc.	
Doctor — > DoctorPhone – the same as with PatientPhone	

4. ER diagram

In the ER diagram, labels like **‘link’** are just *relationship names* — they describe how two entities are connected, but they’re not special ER notation themselves.



Task 2.2

1.

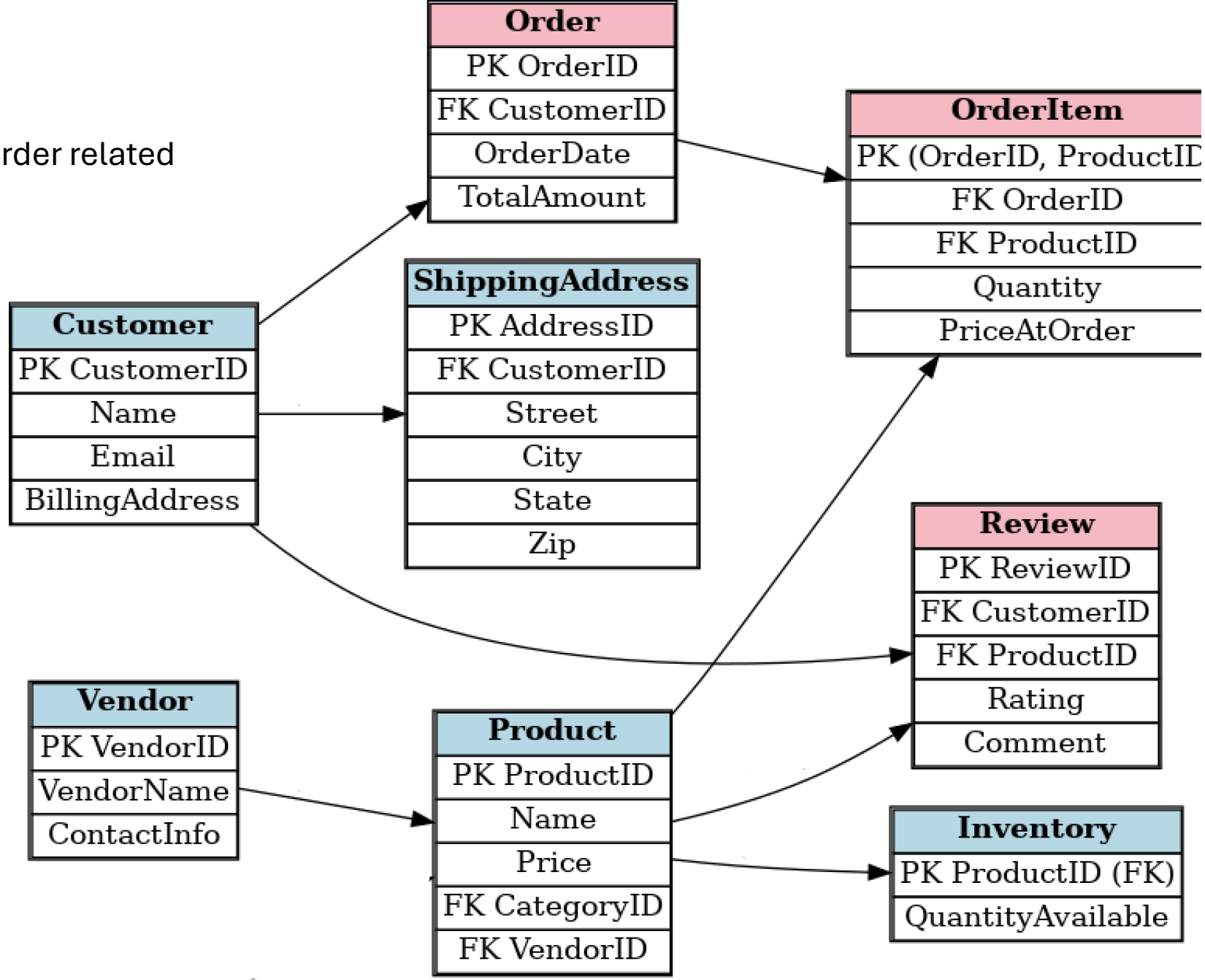
Customer (strong)
Order (weak)
Product (strong)
Category (strong)
Vendor (strong)
Inventory (strong)
ShippingAddress (strong)
OrderItem (weak)
Review (weak)

M:M
Order – Product

There can be many order related to many products

M:M
Customer – Product

Same as ‘Order-Product’



Task 4.1

1.

Functional Dependencies (FDs)

StudentID → StudentName, StudentMajor

ProjectID → ProjectTitle, ProjectType, SupervisorID

SupervisorID → SupervisorName, SupervisorDept

{StudentID, ProjectID} → Role, HoursWorked, StartDate, EndDate

2.

Problems

1) The student's name and major are repeated for each entry if the student is involved in multiple projects.

2) The supervisor information is repeated for all students working on the project.

Anomalies

1) If SupervisorName has changed, it must be changed in all rows.

2) Can't add new student without project or new project without student

3.

There is no violations for 1NF, so no point to fix anything

4**. 2NF**

Primary Key: (StudentID, ProjectID)

Partial Dependencies:

StudentID → StudentName, StudentMajor

ProjectID → ProjectTitle, ProjectType, SupervisorID

Decomposition (2NF):

Student(StudentID, StudentName, StudentMajor)

Project(ProjectID, ProjectTitle, ProjectType, SupervisorID)

Supervisor(SupervisorID, SupervisorName, SupervisorDept)

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

5**. 3NF**

Transitive dependencies:

SupervisorID → SupervisorDept through ProjectID.

Final 3NF decomposition:

Student(StudentID, StudentName, StudentMajor)

Supervisor(SupervisorID, SupervisorName, SupervisorDept)

Project(ProjectID, ProjectTitle, ProjectType, SupervisorID)

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

Task 4.2

1. **Primary Key:** (StudentID, CourseID)

2. Functional Dependencies

StudentID → StudentMajor
CourseID → CourseName
InstructorID → InstructorName
TimeSlot, Room → Building
CourseID → InstructorID, TimeSlot, Room

3. Check BCNF

There is violations:

CourseID → InstructorID, TimeSlot, Room (CourseID is not a superkey).
TimeSlot, Room → Building (also not a superkey).

4. BCNF Decomposition

Student(StudentID, StudentMajor)

Instructor(InstructorID, InstructorName)

Course(CourseID, CourseName, InstructorID, TimeSlot, Room)

RoomSchedule(TimeSlot, Room, Building)

5. Loss of Information

There is no loss of information because all relationships are joined by keys.

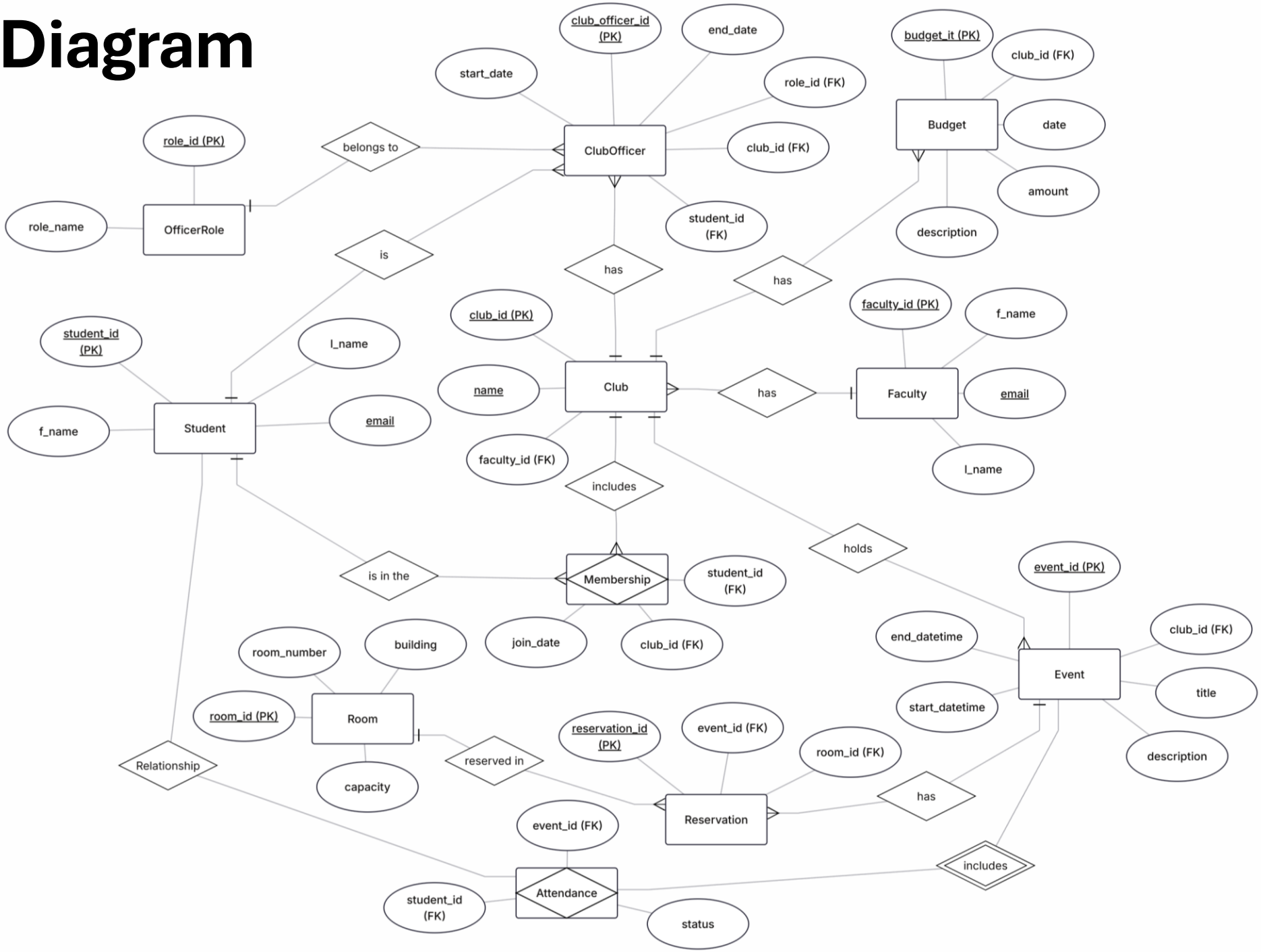
Task 5.1. Design decision and rationale

Reservations — model reservation as attribute of Event vs separate Reservation entity

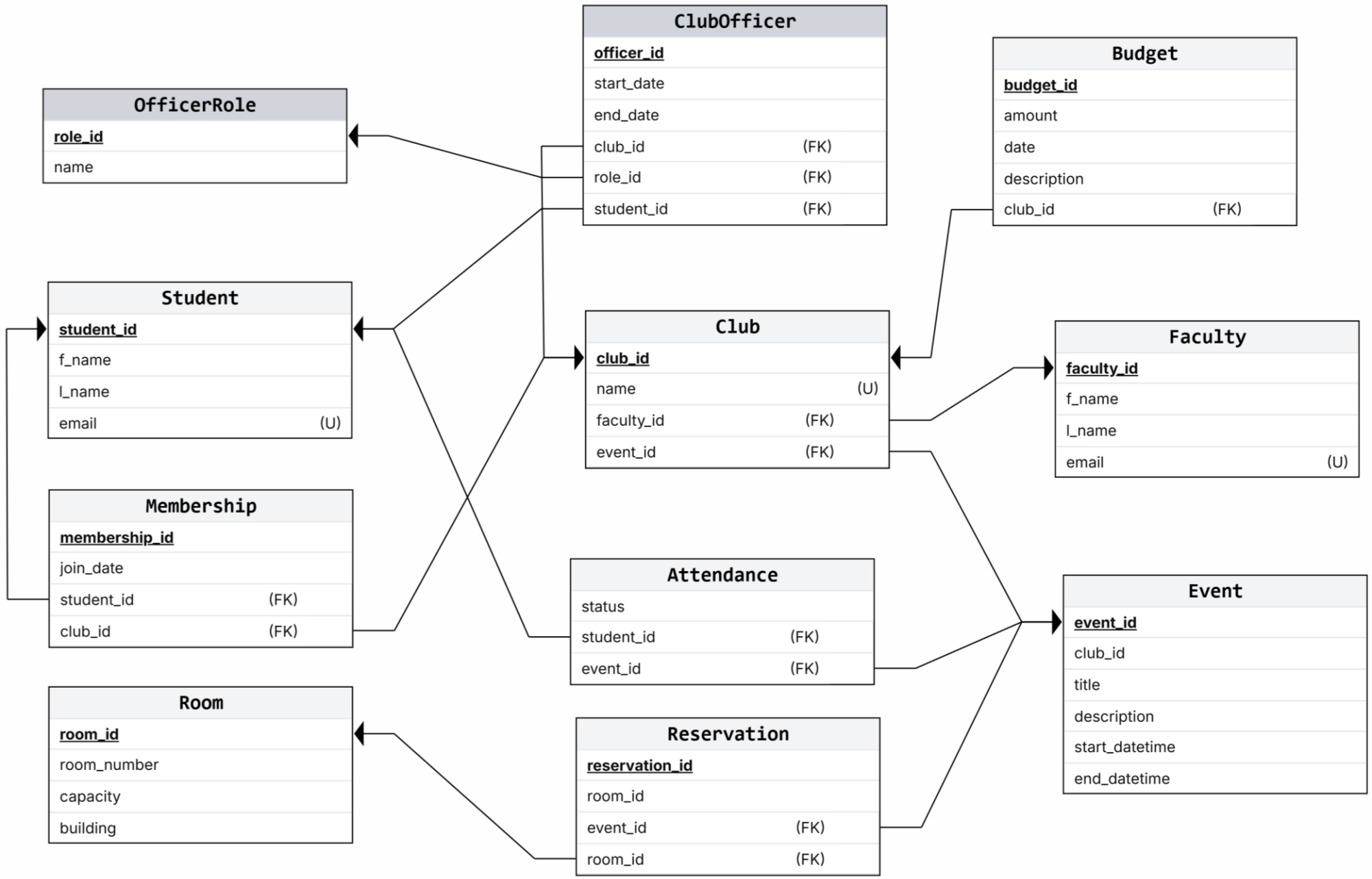
Option: keep room fields in Event (simple) or separate Reservation for complex scheduling.

Choice: separate *Reservation* table to support events that use multiple rooms, changes in reservation status, audit/history of room bookings, and to allow a room reservation without an event (if needed). This is safer for real-world scheduling.

Task 5.1. ER Diagram



Task 5.1. Normalization



Task 5.1. Proofs

All relations in the database schema satisfy the requirements of **First Normal Form (1NF)** (no repeating groups, atomic attributes)

Second Normal Form (2NF) (all non-key attributes fully depend on the whole key for composite PKs)

Third Normal Form (3NF) (no transitive dependencies; non-key attributes depend only on the primary key).

Therefore, the database schema is normalized up to **Third Normal Form (3NF)**.

Task 5.1. Three example queries

- “Retrieve the names of clubs that currently do not have any assigned faculty advisor”
- “List the students who have attended at least three different events organized by the Math Club during the past semester”
- “Find all rooms that are reserved more than once at overlapping times, to detect scheduling conflicts”