Lab Work #1

Nurzhanov Amirkhan

Part 1: Key Identification Exercises

Task 1.1: Superkey and Candidate Key Analysis – Relation A: Employee

1. Superkeys: {EmpID}, {EmpID, Name}, {SSN}, {SSN, Email}, {SSN, Department}, {EmpID, Salary}.
2. Candidate keys: {EmpID}, {SSN}, {Email}.
3. Employee ID since it's a unique set of numbers that is generated systematically and implemented into the database easily without compromising the anonymity of the user (which is why SSN should be avoided since its based in the governmental database that can be exploited; email isn't reliable either since it may change over time).
4. Yes, it is possible since there are no restrictions regarding the use of the same telephone number, especially when it comes to family members working in the same company. Therefore, such possibility can't be excluded.

Relation – B: Course Registration

1. D
2. 2
3. Apart from the inclusion of {StudentID}, there are no additional Candidate keys in this table.

Task 1.2: Foreign Key Design

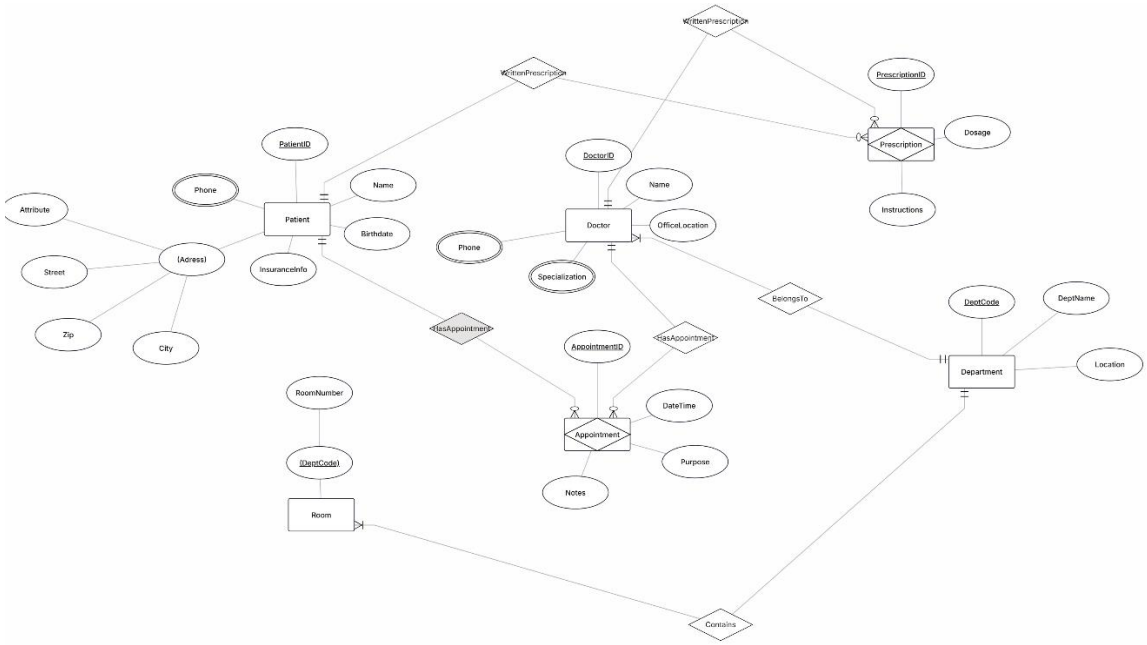
```
CREATE TABLE Professor (
    ProfID INT PRIMARY KEY,
    Name VARCHAR(100),
    Department VARCHAR(50),
    Salary DECIMAL(10,2)
);


CREATE TABLE Department (
    DeptCode VARCHAR(10) PRIMARY KEY,
    DeptName VARCHAR(100),
    Budget DECIMAL(12,2),
    ChairID INT,
    FOREIGN KEY (ChairID) REFERENCES Professor(ProfID)
```

```
);

CREATE TABLE Course (
    CourseID VARCHAR(10) PRIMARY KEY,
    Title VARCHAR(100),
    Credits INT,
    DepartmentCode VARCHAR(10),
    FOREIGN KEY (DepartmentCode) REFERENCES Department(DeptCode)
);

CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100) UNIQUE,
    Major VARCHAR(50),
    AdvisorID INT,
    FOREIGN KEY (AdvisorID) REFERENCES Professor(ProfID)
);

CREATE TABLE Enrollment (
    StudentID INT,
    CourseID VARCHAR(10),
    Semester VARCHAR(20),
    Grade CHAR(2),
    PRIMARY KEY (StudentID, CourseID, Semester),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID)
);
```

Part 2

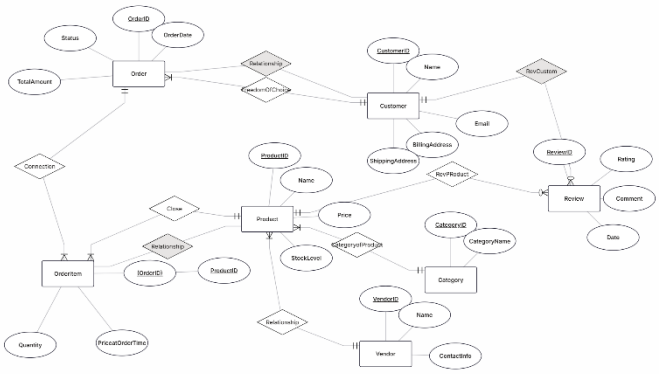
Task 2.1: Hospital Management System

1. Strong entities: Patient, Doctor, Department, Room.
   Weak entities: Specialization, Phone numbers.
2. Patient – simple: PatientID, Birthdate, InsuranceInfo; composite: Address; multi-valued: Phone numbers.
   Doctor – simple: DoctorID, Name, OfficeLocation; composite: Address; multi-valued: Phone numbers.
   Department – simple: DeptCode, DeptName, Location.
   Room – composite key: (DeptCode, RoomNumber).
3. Patient–Appointment–Doctor -> M:N.
   Doctor–Prescription–Patient -> M:N.
   Department–Doctor -> 1:N.
   Department–Room -> 1:N.



4.
5. Primary keys: PatientID, DoctorID, DeptCode, (DeptCode, RoomNumber), AppointmentID, PrescriptionID.

Task 2.2



1.
2. OrderItem is a weak entity since it depends of Product and Order entities. Primary key is based off = (OrderId, ProductID).
3. Order – Product.

Task 4.1

1. StudentID -> StudentName, StudentMajor
   ProjectID -> ProjectTitle, ProjectType, SupervisorID
   SupervisorID -> SupervisorName, SupervisorDept
   (StudentID, ProjectID) -> Role, HoursWorked, StartDate, EndDate
2. Redudancy: StudentName & StudentMajor repeated for each project the student joins. SupervisorName & Dept repeated for each project they supervise.

   Update Anomaly: If supervisor's department changes, must update many rows.

   Insert Anomaly: Cannot add a new project without assigning at least one student (since StudentID is required).

   Delete Anomaly: If the last student working on a project leaves, deleting that row also deletes project details.
3. 1NF: No violations since all values are atomic.
   StudentProject(StudentID, StudentName, StudentMajor, ProjectID, ProjectTitle, ProjectType, SupervisorID, SupervisorName, SupervisorDept, Role, HoursWorked, StartDate, EndDate)
4. 2NF: Primary key - (StudentID, ProjectID) (composite).
   Partial dependencies - StudentID -> StudentName, StudentMajor (depends only on part of our primary key).
   ProjectID -> ProjectTitle, ProjectType, SupervisorID (depends only on ProjectID).

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

5. 3NF: SupervisorID -> SupervisorName, SupervisorDept (already separated into Supervisor Table).

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

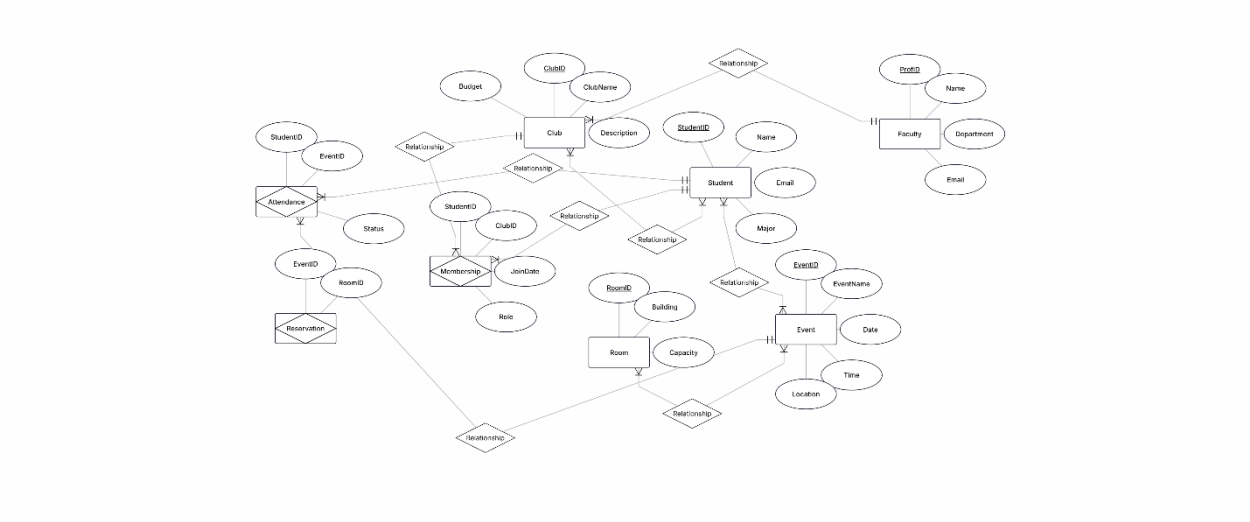
Task 4.2

1. Primary Key = (StudentID, CourseID, TimeSlot, Room).

2. FD = StudentID -> StudentMajor
   CourseID -> CourseName
   InstructorID -> InstructorName
   (TimeSlot, Room) -> Building
   (CourseID, TimeSlot, Room) - > InstructorID
   (StudentID, CourseID, TimeSlot, Room) -> all attributes (full PK dependency).
3. BCNF check: non-trivial FD X -> Y, X is a superkey.
   None of FD possess a superkey, since they only have one attribute.
4. BCNF form: Student (StudentID, Student Major)
   Course(CourseID, CourseName)
   Instructor(InstructorID, InstructorName)
   RoomAssignment(TimeSlot, Room, Building)
   Section(CourseID, TimeSlot, Room, InstructorID)
   Enrollment(StudentID, CourseID, TimeSlot, Room);
5. No loss of information, since all FDs were preserved.

Task 5.1

1)



2) Student(

StudentID PRIMARY KEY,

Name,

Email,

Major

)

Faculty(

ProfID PRIMARY KEY,

Name,

Department,

Email

)

Club(

ClubID PRIMARY KEY,

ClubName,

Description,

Budget,

AdvisorID (FK → Faculty.ProfID)

)

Membership(

StudentID (FK → Student.StudentID),

ClubID (FK → Club.ClubID),

JoinDate,

Role,

PRIMARY KEY (StudentID, ClubID)

)

Event(

EventID PRIMARY KEY,

ClubID (FK → Club.ClubID),

EventName,

Date,

Time,

Location

)

Room(

RoomID PRIMARY KEY,

Building,

Capacity

)

Reservation(

EventID (FK → Event.EventID),

RoomID (FK → Room.RoomID),

PRIMARY KEY (EventID, RoomID)

)

Attendance(

StudentID (FK → Student.StudentID),

EventID (FK → Event.EventID),

Status,

PRIMARY KEY (StudentID, EventID)

)

3) Officer(StudentID, ClubID, Position) – where Officer roles could be modelled as a separate table. But Role was included in Membership to avoid an extra join.

4) Show faculty who advise more than one club.
List all clubs a given student is a member of.
Find all students who attended a Certain event.