Kanshin Arseniy
ID:24B031833

Task 1.1

1. Superkeys:

{EmpID}, {SSN}, {Email}, {Phone}, {EmpID, Name}, {SSN, Phone}

2. Candidate keys:

{EmpID}, {SSN}, {Email}, {Phone}

3. Primary key:

EmpID (best choice since it's numeric)

4. Phone numbers:

In real life, two employees can share the same phone number (e.g., office phones), so the phone is not a reliable identifier, even though it appears unique in the sample data.

Task 1.1

1. **StudentID** – Necessary to identify the student.

CourseCode – Necessary to identify the course.

Section – Necessary to differentiate between different sections of the same course.

Semester – Necessary because a student can take the same course in different semesters

2. **Candidate Keys:** None beyond the primary key (only reorderings).

Task 1.2

Tables & Keys

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

   1.1.PK: StudentID

2. **Professor(ProfID, Name, Department, Salary)**

   2.1.PK: ProfID

3. **Course(CourseID, Title, Credits, DepartmentCode)**

   3.1.PK: CourseID

4. **Department(DeptCode, DeptName, Budget, ChairID)**

   4.1.PK: DeptCode

5. **Enrollment(StudentID, CourseID, Semester, Grade)**

   5.1.PK: usually composite (StudentID, CourseID, Semester)

1. Foreign Keys

2. Student → Professor

2.1.AdvisorID → Professor.ProfID
(Each student has an advisor who is a professor.)

3. Student → Department

3.1.Major → Department.DeptCode
(A student's major must be one of the existing departments.)

4.Course → Department

4.1.DepartmentCode → Department.DeptCode
(Each course belongs to a department.)

5. Department → Professor

5.1.ChairID → Professor.ProfID
(Each department has a professor as its chair.)

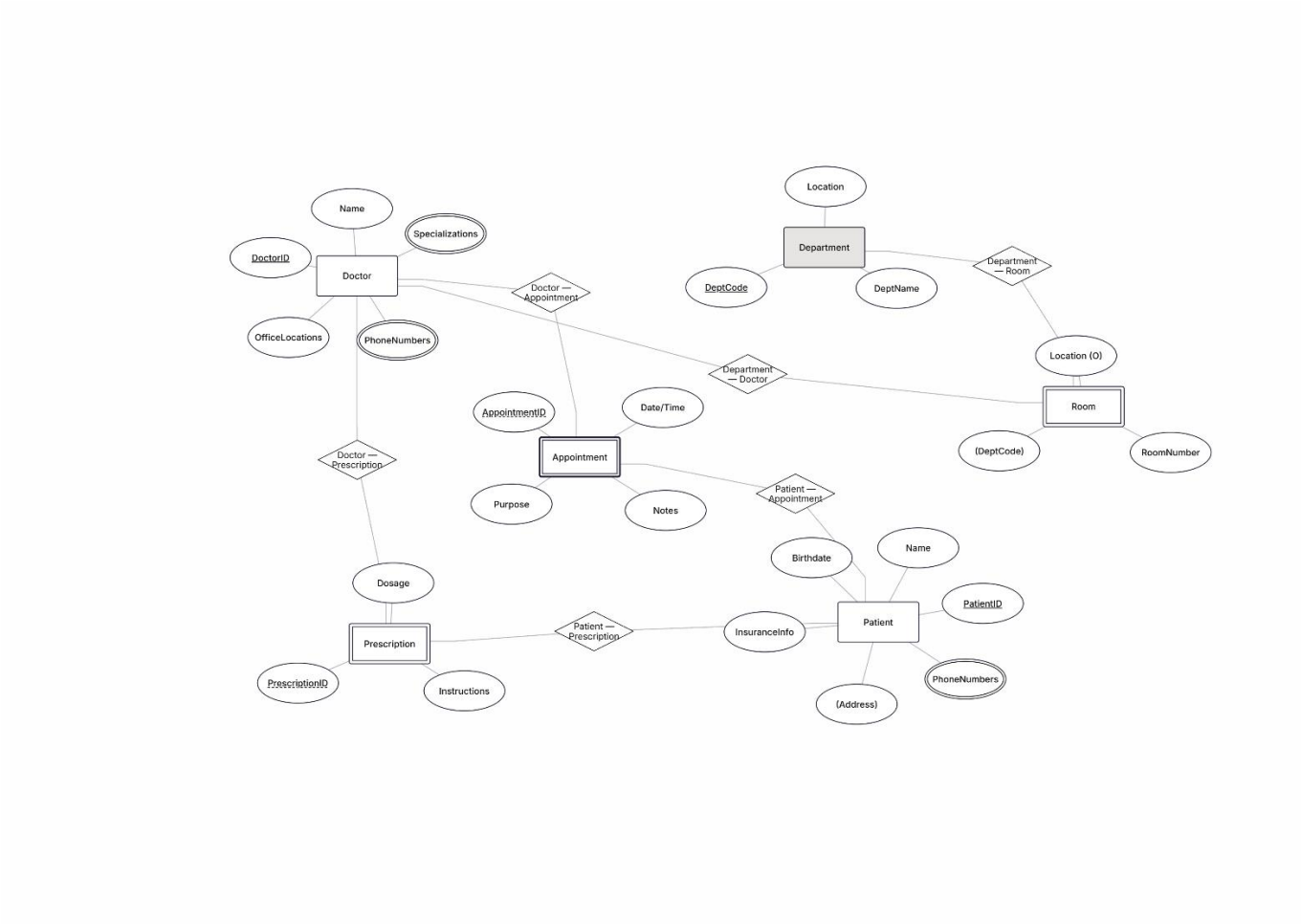
6. **Enrollment → Student**

6.1.StudentID → Student.StudentID
(Each enrollment record refers to a student.)

7. **Enrollment → Course**

7.1.CourseID → Course.CourseID
(Each enrollment record refers to a course.)

Task 2.1



Primary Keys (PK)

1. **Patient** → PatientID
2. **Doctor** → DoctorID
3. **Department** → DeptCode
4. **Appointment** → AppointmentID *(PatientID + DoctorID + DateTime)*
5. **Prescription** → PrescriptionID *(PatientID + DoctorID + Medication)*
6. **HospitalRoom** → (RoomNumber, DeptCode) *(составной PK)*

---

1. Patient — Appointment

- **Cardinality:** One-to-Many (1:M)

2.Doctor — Appointment

- **Cardinality:** One-to-Many (1:M)

3. Patient — Prescription

- **Cardinality:** One-to-Many (1:M)

4. Doctor — Prescription

- **Cardinality:** One-to-Many (1:M)

5. Department — Room

- **Cardinality:** One-to-Many (1:M)

6. Department — Doctor

- **Cardinality:** One-to-Many (1:M)

Task 4.1

Functional Dependencies (FDs):

1. StudentID → StudentName, StudentMajor
2. ProjectID → ProjectTitle, ProjectType, StartDate, EndDate
3. SupervisorID → SupervisorName, SupervisorDept
4. (StudentID, ProjectID) → Role, HoursWorked

Problems:

- Redundancy (student, project, supervisor details repeated).
- Update anomaly (e.g., supervisor name change everywhere).
- Insert anomaly (cannot add student/supervisor without project).
- Delete anomaly (losing student data when project is removed).

Decomposition to 3NF:

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

Task 4.2

Functional Dependencies (FDs):

1. StudentID → StudentMajor
2. CourseID → CourseName, InstructorID

3. InstructorID → InstructorName
4. TimeSlot, Room → Building (rooms are unique across campus)
5. (StudentID, CourseID) → (all other attributes specific to enrollment)

Primary Key: (StudentID, CourseID)

BCNF Decomposition:

1. **Student**(StudentID, StudentMajor)
2. **Course**(CourseID, CourseName, InstructorID)
3. **Instructor**(InstructorID, InstructorName)
4. **Room**(Room, Building)
5. **CourseSchedule**(StudentID, CourseID, TimeSlot, Room)

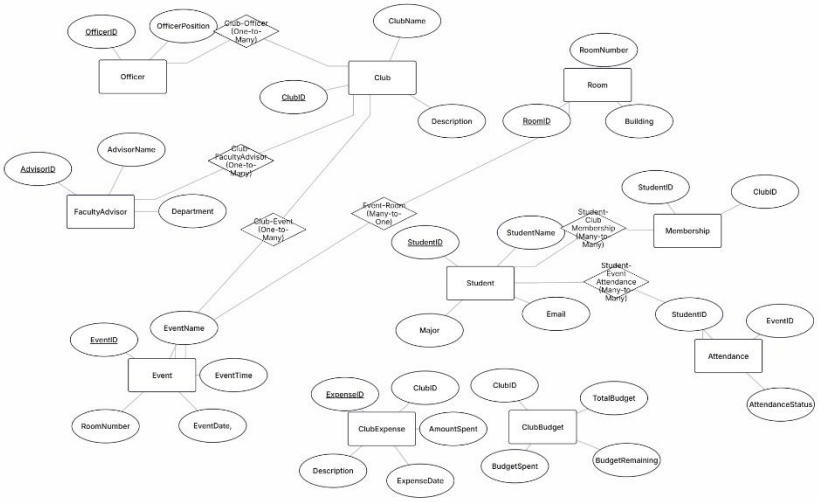
Check BCNF:

- All FDs have determinants as candidate keys → now in BCNF.

Possible Loss of Information:

- No loss of essential information, but queries require more joins after decomposition.

Task 5.1



Relationships:

1. **Student-Club Membership (Many-to-Many)**
    - A student can be a member of multiple clubs, and each club can have many students.
    - This is captured by the **Membership** entity.
2. **Club-Officer (One-to-Many)**
    - Each club has one or more officers, and officers can only hold positions in a single club at a time. However, students can be officers in multiple clubs.
    - **ClubOfficer** tracks student officer positions in clubs.
3. **Club-FacultyAdvisor (One-to-Many)**

- Each club has one faculty advisor, but one advisor can advise multiple clubs.
- **FacultyAdvisor** is linked to **Club** via the **FacultyAdvisorID**.

4. **Club-Event (One-to-Many)**
    - A club can organize many events, but each event belongs to only one club.
5. **Event-Room (Many-to-One)**
    - Each event is scheduled in a specific room, but a room can host many events.
6. **Student-Event Attendance (Many-to-Many)**
    - Students can attend multiple events, and each event can have many students attending.
7. **Club-Budget (One-to-One)**
    - Each club has one associated budget.
8. **Club-Expense (One-to-Many)**
    - A club can have multiple expenses recorded in the **ClubExpense** table