**Databases Lab1;**

# Part 1: Key Identification Exercises

# Task 1.1: Superkey and Candidate Key Analysis

# Relation A: Employee

**1) Six example superkeys**

1. **{EmpID}**
2. **{SSN}**
3. **{Email}**
4. **{Phone} *(unique in the sample — may or may not be globally unique depending on business rules)***
5. **{EmpID, Email}**
6. **{SSN, Phone}**

**2)Candidate Keys**

**EmpID**

**SSN**

**Email**

**Phone *(conditionally — only if phone numbers are guaranteed unique in your business rules)***

**3)** **Recommended primary key**

**Recommended primary key: EmpID (a surrogate/system identifier).**

**Reasons:**

* **Stability: EmpID is usually generated by the system and does not change over time.**
* **Non-sensitive: Unlike SSN, EmpID does not expose sensitive personal data.**
* **Compact & efficient: Typically an integer — smaller index size and faster joins/searches.**
* **Control: The organization controls its generation (no external collisions or formatting issues).**
* **Flexibility: Works for all employees (including contractors, foreign staff, or people without SSNs/emails).**

**4)Yes — in real life it’s possible for two employees to share the same phone number.**

Relation B: Course Registration

**1. Minimum attributes for primary key**

(StudentID, CourseCode, Section, Semester, Year)

**2)Why each attribute is necessary**

* **StudentID** – identifies which student
* **CourseCode** – identifies which course
* **Section** – distinguishes different sections of the same course
* **Semester** – same course/section can be taken in different semesters
* **Year** – same semester name (Fall) can occur in different years

3)

 **Natural candidate key**:  
(StudentID, CourseCode, Section, Semester, Year)  
(this is the only natural combination guaranteed to be unique under the given rules)

 **Possible surrogate key** (if you add it):  
RegistrationID (a system-generated unique ID)

# Task 1.2:Foreign Key Design

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

* **Primary Key (PK):** StudentID
* **Foreign Keys (FK):**
  + Major → Department.DeptCode  
    (Major — это код специальности, которая соответствует коду отдела/факультета)
  + AdvisorID → Professor.ProfID  
    (каждому студенту назначен научный руководитель — профессор)

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

* **PK:** ProfID
* **FK:**
  + Department → Department.DeptCode  
    (каждый профессор работает в каком-то отделении)

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

* **PK:** CourseID
* **FK:**
  + DepartmentCode → Department.DeptCode  
    (каждый курс относится к какому-то отделению/факультету)

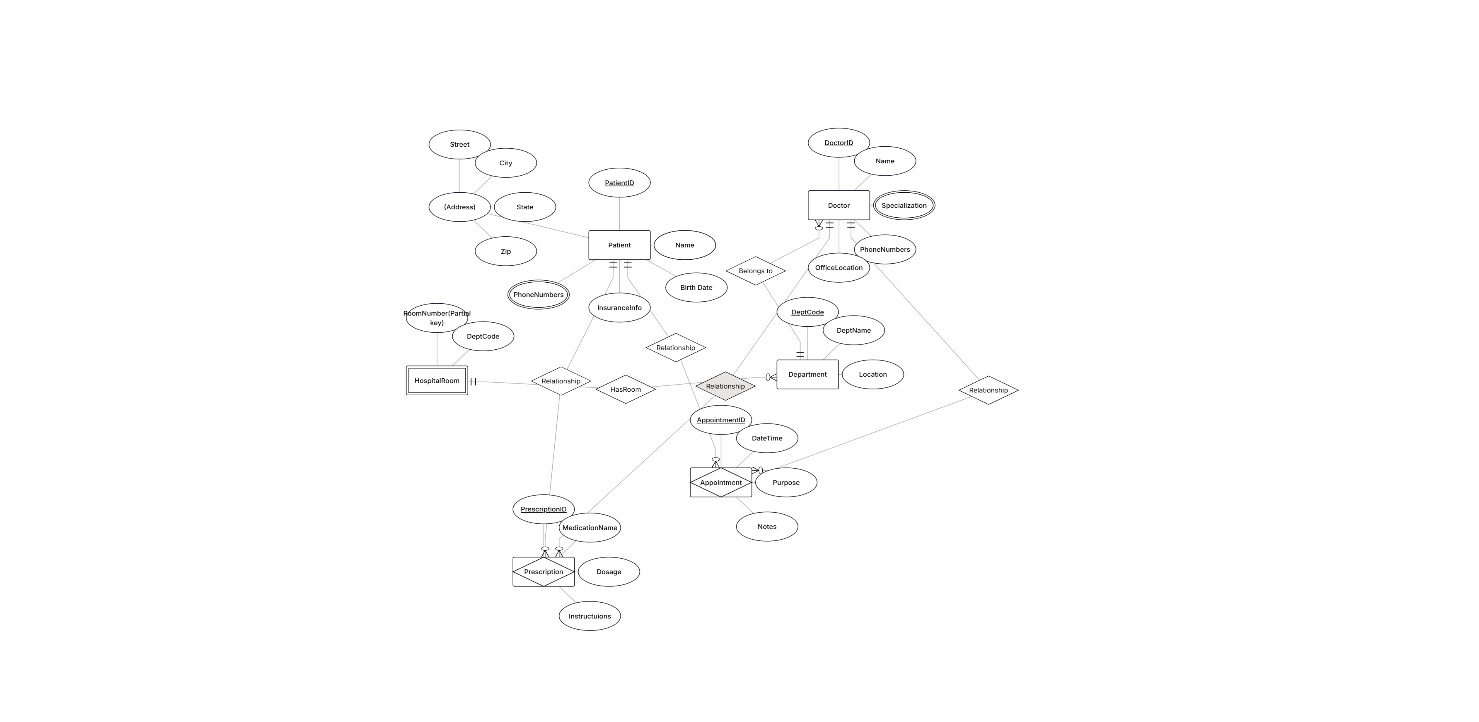
**4. Department(DeptCode, DeptName, Budget, ChairID)**

* **PK:** DeptCode
* **FK:**
  + ChairID → Professor.ProfID  
    (каждое отделение имеет заведующего, который является профессором)

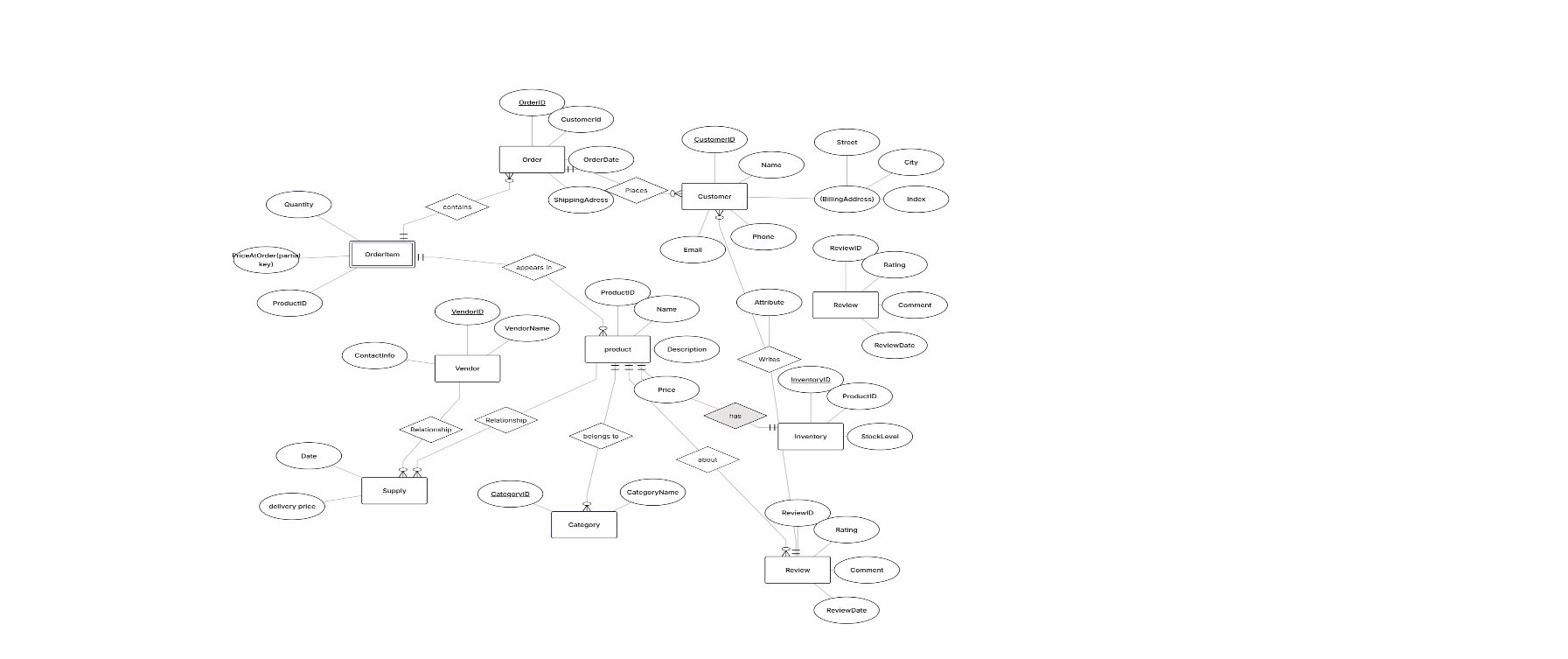
**5. Enrollment(StudentID, CourseID, Semester, Grade)**

* **PK (составной):** (StudentID, CourseID, Semester)
* **FK:**
  + StudentID → Student.StudentID
  + CourseID → Course.CourseID

Part 2: ER Diagram Construction

Task 2.1: Hospital Management Systemart

Task 2.2: E-commerce Platform



Part 4: Normalization Workshop Task

4.1: Denormalized Table Analysis

**Functional Dependencies (FDs)**

Based on the table:

* **StudentID → StudentName, StudentMajor**  
  (Each student has one name and one major)
* **ProjectID → ProjectTitle, ProjectType, SupervisorID**  
  (Each project has one title, type, and supervisor)
* **SupervisorID → SupervisorName, SupervisorDept**  
  (Each supervisor has one name and department)
* **(StudentID, ProjectID) → Role, HoursWorked, StartDate, EndDate**  
  (For each student-project pair, we store role, hours, and dates)

Additionally, we have a transitive dependency:

* **ProjectID → SupervisorID → SupervisorName, SupervisorDept**

**2)Problems (Redundancy and Anomalies)**

**Redundancy:**

* Student name and major are repeated for each project the student participates in.
* Project info and supervisor info are repeated for each student in the project.

**Examples of anomalies:**

* **Update anomaly:** Changing a supervisor’s department requires updating multiple rows.
* **Insert anomaly:** Cannot insert a new supervisor without a project and a student.
* **Delete anomaly:** Deleting the last student in a project deletes information about the project and supervisor.

**3)1NF Check**

All attributes are atomic (no lists or multiple values in one cell).  
**→ No 1NF violations.**

(If, for example, Role contained multiple roles in one cell, we would need to separate rows.)

**4)Applying 2NF**

**Primary Key:** (StudentID, ProjectID)

* Each row is uniquely identified by the combination of student and project.

**Partial dependencies:**

* StudentID → StudentName, StudentMajor
* ProjectID → ProjectTitle, ProjectType, SupervisorID

**2NF Decomposition:**

* **STUDENT(StudentID, StudentName, StudentMajor)**
* **PROJECT(ProjectID, ProjectTitle, ProjectType, SupervisorID)**
* **SUPERVISOR(SupervisorID, SupervisorName, SupervisorDept)**
* **STUDENT\_PROJECT(StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate)**

(STUDENT\_PROJECT is the many-to-many relationship with attributes.)

**5)** **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, decompoApplying 3NF**

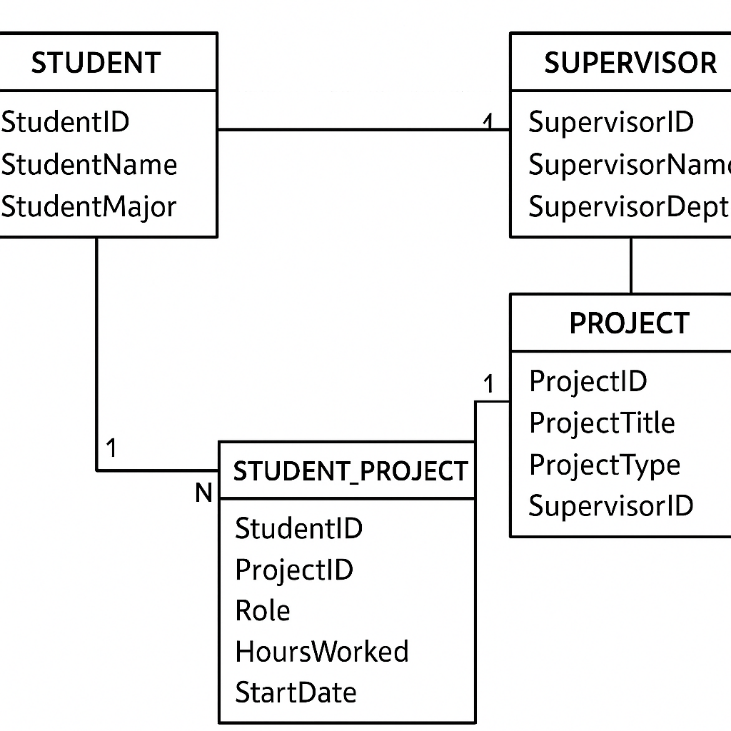
Check for transitive dependencies:

* In PROJECT: ProjectID → SupervisorID → SupervisorName, SupervisorDept  
  (transitive dependency via SupervisorID)

**Solution:** Supervisor info is in its own table. Now PROJECT only contains SupervisorID as a foreign key.

**Final 3NF schema:**

* **STUDENT(StudentID PK, StudentName, StudentMajor)**
* **SUPERVISOR(SupervisorID PK, SupervisorName, SupervisorDept)**
* **PROJECT(ProjectID PK, ProjectTitle, ProjectType, SupervisorID FK)**
* **STUDENT\_PROJECT(StudentID FK, ProjectID FK, Role, HoursWorked, StartDate, EndDate,  
  PRIMARY KEY(StudentID, ProjectID))**



**Task 4.2: Advanced Normalization**

**1. Primary Key**

Each record represents a student enrolled in a course section. A section is uniquely identified by CourseID + TimeSlot + Room.  
 **Primary Key:** (StudentID, CourseID, TimeSlot, Room)

**2. Functional Dependencies**

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

**3. BCNF Check**

BCNF requires every determinant to be a superkey.

* StudentID → StudentMajor -
* CourseID → CourseName -
* InstructorID → InstructorName -
* (TimeSlot, Room) → Building -
* (CourseID, TimeSlot, Room) → InstructorID -  
  + Table is **not in BCNF**.

**4. BCNF Decomposition**

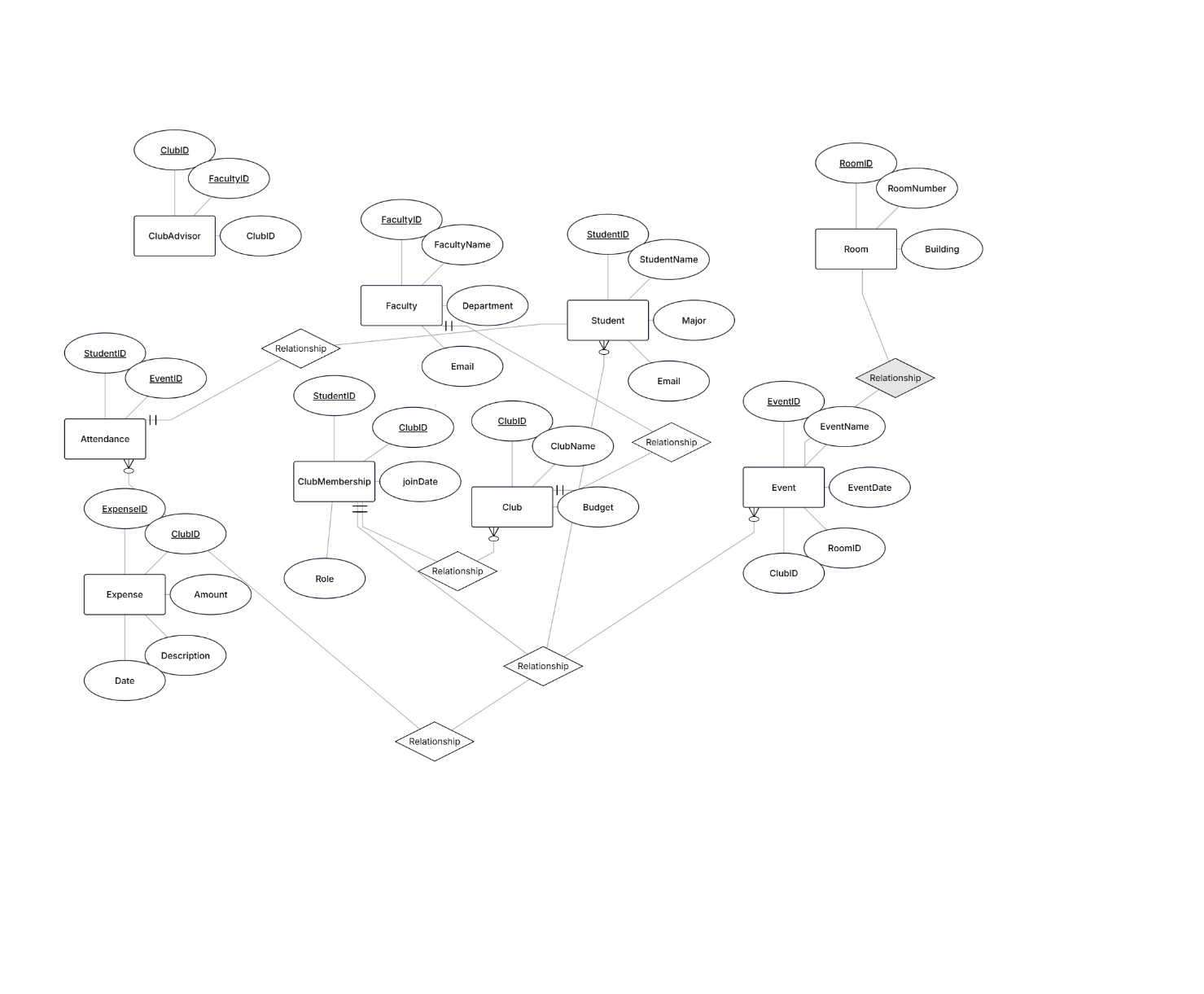
1. **Students:** Student(StudentID, StudentMajor)
2. **Courses:** Course(CourseID, CourseName)
3. **Instructors:** Instructor(InstructorID, InstructorName)
4. **Rooms:** RoomSchedule(TimeSlot, Room, Building)
5. **Sections:** Section(CourseID, TimeSlot, Room, InstructorID)
6. **Enrollments:** Enrollment(StudentID, CourseID, TimeSlot, Room)

**5. Loss of Information**

+ No information is lost; all original data can be reconstructed via **joins**.

Part 5: Design Challenge

Task 5.1: Real-World Application



**Step 2: Normalized Relational Schema**

1. **Student(StudentID PK, StudentName, Major, Email)**
2. **Faculty(FacultyID PK, FacultyName, Department, Email)**
3. **Club(ClubID PK, ClubName, Budget, FacultyID FK)** ← Advisor included as FK
4. **ClubMembership(StudentID FK, ClubID FK, JoinDate, Role, PK(StudentID, ClubID))**
5. **Event(EventID PK, EventName, EventDate, ClubID FK, RoomID FK)**
6. **Room(RoomID PK, RoomNumber, Building)**
7. **Attendance(StudentID FK, EventID FK, PK(StudentID, EventID))**
8. **Expense(ExpenseID PK, ClubID FK, Amount, Description, Date)**

All tables are in **BCNF**: all FDs have superkeys as determinants.

**Step 3: Design Decisions**

**Decision Example:**

* **Where to store officer positions**
  + Option 1: Add Role in ClubMembership (a student can be an officer)
  + Option 2: Create a separate ClubOfficer table for officers only
* **Choice:** I included Role in ClubMembership because it reduces the number of tables and captures the officer role naturally for students who are members.

**Step 4: Example Queries (in English)**

1. **Find all members of a specific club and their roles.**
2. **List all events a student attended in the last semester.**
3. **Calculate the total expenses of each club and compare with their budget.**