Laboratory work #3

Ishutin Nikolay

**3.2 Database Systems. Relational Model & Keys**

**Part 1: Key Identification Exercises**

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

**Relation A: Staff**

**A table of numbers and text

AI-generated content may be incorrect.**

**Your Tasks:**

1. List at least 4 different superkeys for the Staff relation (show the attribute sets).

Answer: {StaffID}, {NationalID}, {WorkEmail}, {StaffID, FullName}

1. Identify all candidate keys.

Answer: {StaffID}, {NationalID}, {WorkEmail}

1. Which candidate key would you choose as the primary key and why? Provide justification(uniqueness, stability, privacy, etc.).

Answer: {StaffID}, because this key is always unique, does not disclose personal data, and is stable.

1. Could two staff members share the same mobile number? Explain whether the given data supports that and what business rule you would choose.

Answer: Two staff members can have 2 identical numbers, for example, if they use the same service number. But it's better to make the rule UNIQUE for mobile.

**Relation B: CourseEnrollment**

CourseEnrollment(StudentNo, SubjectCode, GroupNo, Term, Year, Grade, CreditHours)

**Business Rules:**

* A student may take the same subject in different terms/years.
* A student cannot enroll in the same SubjectCode & GroupNo combination in the same Term & Year twice.
* Each Group in a term has a fixed CreditHours value.

**Your Tasks:**

1. Determine the minimum set of attributes required for the primary key. Explicitly list them.

Answer: {StudentNo, SubjectCode, GroupNo, Term, Year}

1. Explain why each attribute in that key is necessary (tie to the business rules).

Answer: StudentNo — identifies the student, SubjectCode — defines the subject, GroupNo — separates by groups, Term and Year are distinguished by different semesters and years.

1. Identify any other candidate keys (if any) and explain why they are or are not valid.

Answer: Other keys do not ensure the uniqueness of the data for the table.

**Task 1.2: Foreign Key Design**

Given these tables for the university:

* Student(StudentNo, FullName, Email, MajorCode, AcademicAdvisorID)
* Lecturer(LecturerID, FullName, DeptCode, Title)
* Module(ModuleID, Title, Credits, DeptCode)
* Faculty(DeptCode, DeptName, DeanID)
* Registration(StudentNo, ModuleID, Term, Year, Status)

**Your Tasks:**

1. Identify all foreign key relationships between these tables. For each FK state: referencing table, referencing attribute(s), referenced table, referenced attribute(s), and whether the FK should be ON DELETE CASCADE, SET NULL, or RESTRICT, with justification.

Answer:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Referenc**ing** table | Referenc**ed** table | Referenci**ng** attribute(s) | Referenc**ed** attribute(s) | ON DELETE |
| 1 | Registration | * Student | StudentNo | StudentNo | CASCADE |
| 2 | Module | * Faculty | DeptCode | DeptCode | RESTRICT |
| 3 | Lecturer | * Faculty | DeptCode | DeptCode | RESTRICT |
| 4 | Registration | * Module | ModuleID | ModuleID | RESTRICT |
| 5 | Student | * Lecturer | AcademicAdvisorID | LecturerID | SET NULL |
| 6 | Student | * Faculty | MajorCode | DeptCode | RESTRICT |

№1 Registration ►Student - If a student is deleted, all his registrations must be deleted.  
№2 Module ►Faculty - we can't delete a faculty while there are modules.  
№3 Lecturer ► Faculty - we can't delete a faculty while there are teachers.  
№4 Registration ► Module – we can't delete a module if there are registrations for it.  
№5 Student ► Lecturer - If the teacher is removed, the student remains, but without a supervisor.  
№6 Student ► Faculty - we can't delete a faculty while students are studying there.

**Part 2: ER Diagram Construction**

**Task 2.1: Clinic Information System**

**Scenario Requirements:**

* + Patients: patient numbers (unique), name, date of birth, contact addresses (may have multiple), multiple phone numbers, emergency contact, insurance provider and policy number.
  + Physicians: physician ID (unique), name, specialties (physician may have multiple specialties), contact numbers, office room(s), and employment start date.
  + Clinics/Departments: code, name, floor location.
  + Appointments: record patient — physician meetings with appointment datetime, reason, duration (in minutes), and visit notes.
  + Treatments: each treatment entry links to an appointment and may include procedure codes, costs, and follow-up instructions.
  + Rooms: rooms are identified by (ClinicCode, RoomNumber) — room numbers are reused across clinics.

**Your Tasks:**

1. List all entities and indicate which are strong and which (if any) are weak.

Answer: Patients: STRONG, because has its own unique patientID, can exist independently  
Physicians: STRONG, because has a physicianID, does not depend on other tables  
Clinics/Departments: STRONG because has a clinicID, exists separately  
Rooms:WEAK because It is identified by a combination (clinicID, roomNumber) that is, it depends on the clinic where it is located  
Appointments: STRONG because It has its own AppointmentID, but contains FKs for the patient and the doctor.  
Treatments: WEAK because there is no appointment (AppointmentID) because the treatment is associated with a specific patient visit

1. For each entity list attributes and classify them as simple, composite, multi-valued, or derived.

Answer: Patients: **PK:** patientID, fullName *(simple)*, dateOfBirth *(simple)*, contactAddresses *(multi-valued)*, phoneNumbers *(multi-valued)*, emergencyContact *(simple)*, insuranceProvider *(simple)*, policyNumber *(simple)*.

Physicians: **PK:** physicianID, name *(simple)*, specialties *(multi-valued)* contactNumbers *(multi-valued),* officeRoom *(simple)*, employmentStartDate *(simple)*, **FK:** clinicID references ClinicsDepartments(clinicID)

ClinicsDepartments: **PK:** clinicID, clinicName *(simple)*, floorLocation *(simple)*

Rooms: **PK:** (clinicID, roomNumber) *(composite key)* **FK:** clinicID references ClinicsDepartments(clinicID), capacity *(simple)*

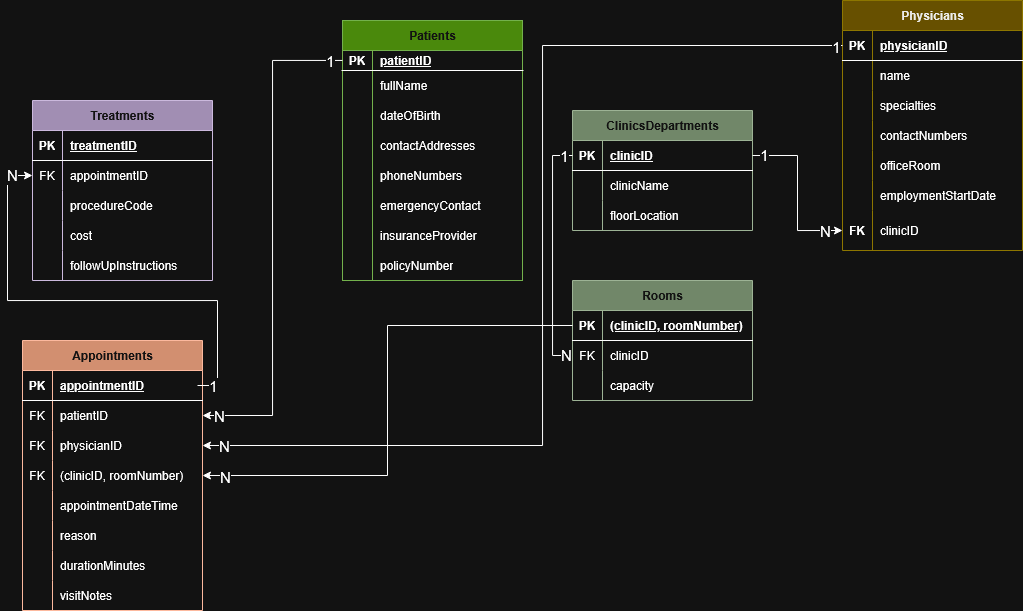
Appointments: **PK:** appointmentID,

**FK:** patientID references Patients(patientID),  
physicianID references Physicians(physicianID),   
(clinicID, roomNumber) references Rooms(clinicID, roomNumber) *(composite key)*,  
appointmentDateTime *(simple)*, reason *(simple)*, durationMinutes *(simple)*, visitNotes *(simple)*

Treatments: **PK:** treatmentID, **FK:** appointmentID references Appointments(appointmentID), procedureCode *(simple)*, cost *(simple)*, followUpInstructions *(simple)*

1. Identify relationships and state cardinalities (1:1, 1:N, M:N); note any associative entities required.

Answer: ClinicDepartments ► Physicians: 1:N  
ClinicDepartments ► Rooms: 1 : N  
Patient ► Appointments: 1 : N  
Physician ► Appointments: 1 : N  
Room ► Appointments: 1 : N  
Appointment ► Treatments: 1 : N

1. Draw a complete ER diagram using Chen or Crow’s Foot notation; mark primary keys clearly. 
2. Suggest how to handle physicians with multiple specialties (one-to-many vs. M:N with a junction table) and justify your choice.

Answer: A doctor may have several specialties, and the same specialty may belong to several doctors.Therefore, there is a many-to-many (M:N) relationship between Physicians and Specialties.For correct modeling, an intermediate PhysicianSpecialties table is created, containing pairs (physicianID, specialtyID).

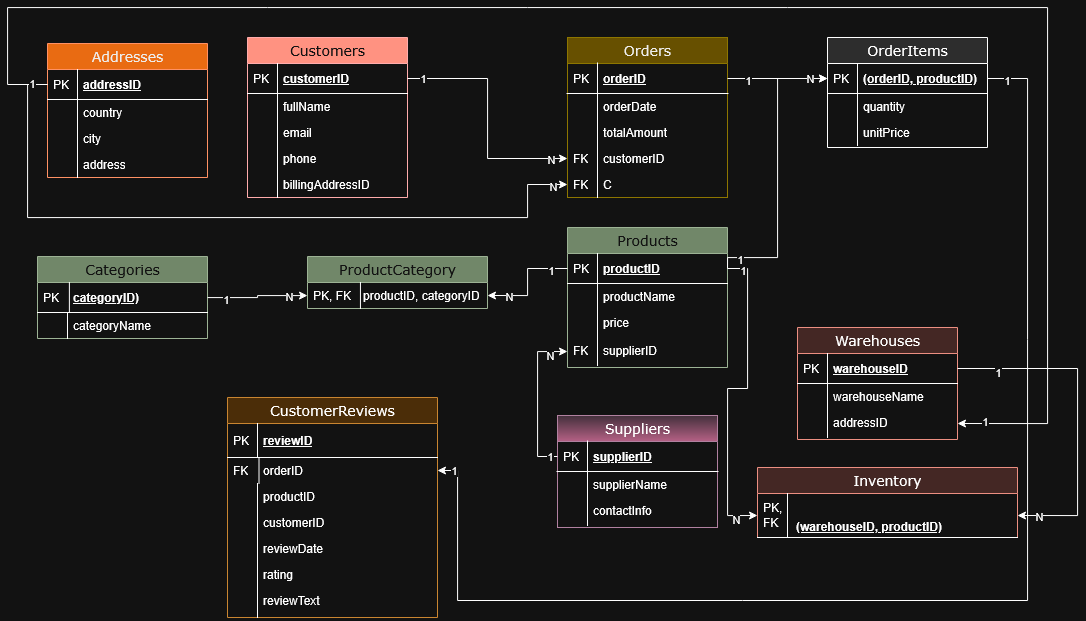
**Task 2.2: Online Marketplace**

**Scenario Requirements:**

* + Customers place Orders. Orders contain OrderItems (each item records quantity and price at purchase time).
  + Products belong to Categories (a product can belong to multiple categories) and are provided by Suppliers.
  + Customers can have multiple shipping addresses distinct from billing address.
  + Products have CustomerReviews (rating, review text, review date) — a customer may review a product only once per order of that product.
  + Inventory tracks stock by Warehouse and SKU; warehouses are physical locations with address info.

**Your Tasks:**

1. Provide a full ER diagram capturing Customers, Orders, OrderItems, Products, Categories, Suppliers, Warehouses, Inventory, and Reviews.



2. Identify a weak entity in this design (if any) and explain why it is weak.

Answer: OrderItems – weak, because OrderItems depends on Orders.

1. Identify at least one many-to-many relationship that requires attributes (e.g., Product–Category or Product–Warehouse inventory snapshot), and show how you model it (associative entity with attributes).  
   Answer: Product ↔ Category. M:N relationship, since one product can belong to multiple categories, and one category can include multiple products. Table: ProductCategory.  
   Product ↔ Warehouse. This is also an M:N connection, but with an additional stockQuantity attribute. Table:Inventary.

**Part 4: Normalization Workshop**

**Task 4.1: Denormalized Table Analysis**

**Given Table:**

ResearchParticipation(StudentNo, StudentName, Major, ProjectCode, ProjectTitle,

ProjectDomain, SupervisorID, SupervisorName, SupervisorDept, RoleInProject,

HoursContributed, StartDate, EndDate)

**Your Tasks:**

1. List all functional dependencies (FDs) you can infer from the attributes (use the format A → B).

Answer: 1. StudentNo → StudentName, Major

2. ProjectCode → ProjectTitle, ProjectDomain, SupervisorID

3. SupervisorID → SupervisorName, SupervisorDept

4. (StudentNo, ProjectCode) → RoleInProject, HoursContributed, StartDate, EndDate

2. Identify redundancy: describe what information repeats and provide concrete examples.

Answer: ProjectTitle, ProjectDomain, SupervisorID, SupervisorName, SupervisorDept are repeated for each student participating in the same project.

studentName, Major are repeated when one student participates in several projects.

1. Anomalies: give examples of update, insert, and delete anomalies that would arise.

Answer:

* Update anomaly: Example: If the SupervisorDept for Dr. Lee has changed, you need to update each row where SupervisorID = S001 occurs. There may be inconsistencies if we forget to update somewhere.
* Insert anomaly: Example: You cannot add a new Supervisor without immediately creating a ResearchParticipation record. Loss of data about the supervisor.
* Delete anomaly: Example If we delete the last student participating in Project P10, we will lose all data about the project and its supervisor.Loss of related information.

1. 1NF: Are there 1NF violations? If yes, fix them and show the corrected schema.   
   1NF is violated if the cells contain multiple values (lists, arrays, etc.).

Answer:

Here each field contains one value.

The table is already in 1NF.

1. 2NF: Propose the primary key for the denormalized table; detect any partial dependencies; provide a 2NF decomposition.

Answer:

Each record is defined by a unique combination (StudentNo, ProjectCode) - this is a composite key .2NF is violated if the attribute depends only on a part of the composite key.

StudentNo → studentName, Major - partial dependency.

ProjectCode → ProjectTitle, ProjectDomain, SupervisorID - partial dependency So 2NF is broken.

Breaking it up

1. Students(StudentNo, StudentName, Major)
2. Projects(ProjectCode, ProjectTitle, ProjectDomain, SupervisorID)
3. Supervisors(SupervisorID, SupervisorName, SupervisorDept)
4. ResearchParticipation(StudentNo, ProjectCode, RoleInProject, HoursContributed, StartDate, EndDate)
5. 3NF: Identify transitive dependencies and provide a final 3NF decomposition with table schemas and keys.

Answer: Transitive dependencies are when an attribute does not depend on the key directly, but through another attribute.

SupervisorID → SupervisorName, SupervisorDept - Transitive dependency (via SupervisorID).

1. Students(StudentNo, StudentName, Major) Key: StudentNo

1. Supervisors(SupervisorID, SupervisorName, SupervisorDept) Key: SupervisorID
2. Projects(ProjectCode, ProjectTitle, ProjectDomain, SupervisorID) Key: ProjectCode
3. ResearchParticipation(StudentNo, ProjectCode, RoleInProject, HoursContributed, StartDate, EndDate) Key: (StudentNo, ProjectCode)

**Task 4.2: Advanced Normalization**

**Given Table:**

ClassTimetable(StudentNo, StudentMajor, CourseCode, CourseTitle, InstructorID,

InstructorName, Slot, RoomNumber, Campus)

**Business Rules:**

* + Each student has one declared major.
  + Each course code maps to exactly one course title.
  + Instructors are identified by InstructorID and have a fixed name.
  + Slot + RoomNumber together imply Campus (rooms are unique per campus but may repeat numbers across campuses).
  + Each course section (CourseCode taught at a Slot in a Room) has exactly one instructor.
  + Students can enroll in many course sections.

**Your Tasks:**

1. Determine the correct primary key for the table (explain your reasoning).

Answer: "Each course section (CourseCode located at a Slot in a Room) has exactly one instructor." → the course section is defined by a triple (CourseCode, Slot, roomNumber).  
Therefore, the natural composite primary key for the source table is a combination of: (StudentNo, CourseCode, Slot, roomNumber) - this uniquely identifies the entry "student X in the section (CourseCode, Slot, roomNumber)".

1. List all functional dependencies. Answer:

Answer:   
StudentNo → StudentMajor: each student has one declared major

CourseCode → CourseTitle: each course code has exactly one name

InstructorID → InstructorName: the instructor is identified by an ID and has a fixed name

(Slot, RoomNumber) → Campus: by convention: Slot + roomNumber together define the Campus

(CourseCode, Slot, RoomNumber) → InstructorID: each section is a combination of CourseCode+Slot+roomNumber has exactly one teacher

the full key:

(StudentNo, CourseCode, Slot, RoomNumber) → (StudentMajor, CourseTitle, InstructorID, InstructorName, Campus)

1. Is the table in BCNF? If not, decompose it to BCNF, showing every step and the schemas that result.

Answer: BCNF rule: for every nonzero functional dependency X ► Y, X must be a superkey. But since we have several functional dependencies, this is not a BCNF. For example StudentNo → StudentMajor is a violation, because StudentNo is not a superkey.  
Step 1 Separating the students:  
FD StudentNo → StudentMajor indicates that StudentMajor depends only on StudentNo. We'll take it out:

Students(StudentNo PK, StudentMajor)

Removing the StudentMajor from the source table.

Step 2 separate the courses:

FD CourseCode → CourseTitle takes out:

Courses(CourseCode PK, CourseTitle)

Removing CourseTitle from the source table.

Step 3 Separating the instructors:

FD InstructorID → InstructorName takes out

Instructors(InstructorID PK, InstructorName)

Removing the InstructorName from the main table.

Step 4 Processing (Slot, roomNumber) → Campus:

The important point here is: FD (Slot, roomNumber) → Campus indicates that the combination (Slot, roomNumber) defines Campus by itself. This means that we should have a table where the key will be (Slot,roomNumber) and in which the Campus is stored. Let's call it RoomSlots or RoomAssignments.:

RoomSlots:

RoomSlots(Slot, RoomNumber, Campus)

PRIMARY KEY (Slot, RoomNumber) (we do PK = (Slot,roomNumber) because the left part of FD is the determinant)  
Removing the Campus from the main table.

Step 5 select the sections:

FD (CourseCode, Slot, RoomNumber) → InstructorID says that the section is identified by a triple (CourseCode,Slot,roomNumber) and identifies the teacher. highlights the essence of Sections.:

Table: Sections

Attributes:

* CourseCode — code of the course.
* Slot — time slot during which the course is taught.
* RoomNumber — classroom where the course takes place.
* InstructorID — the instructor assigned to this section.

Primary Key:

(CourseCode, Slot, RoomNumber)

Foreign Keys:

* CourseCode → Courses(CourseCode)
* InstructorID → Instructors(InstructorID)
* (Slot, RoomNumber) → RoomSlots(Slot, RoomNumber)  
    
  This table represents individual course sections.

Here Sections stores the "section → teacher" relationship, and a link to RoomSlots makes it possible to get a Campus.

Step 6 — the remaining table is the enrollment records:

Now the original flat table is reduced to the "student is enrolled in the section" entity:

Table: Enrollments

Attributes:

* StudentNo — unique ID of the student.
* CourseCode — code of the course the student is enrolled in.
* Slot — time slot of the section.
* RoomNumber — classroom where the section takes place.

Primary Key:

(StudentNo, CourseCode, Slot, RoomNumber)

Foreign Keys:

* StudentNo → Students(StudentNo)
* (CourseCode, Slot, RoomNumber) → Sections(CourseCode, Slot, RoomNumber)  
  This table records which students are enrolled in which course sections.  
  Each record uniquely identifies the enrollment of a specific student in a particular course section.

This table stores the fact "student — section".

1. Explain any potential loss of information or need for joins after decomposition.  
   No information is lost (with lossless decomposition), but to get the original "flat" representation, you need to connect the tables (JOINS): Enrollments → Students, Sections → Courses, Instructors and RoomSlots.

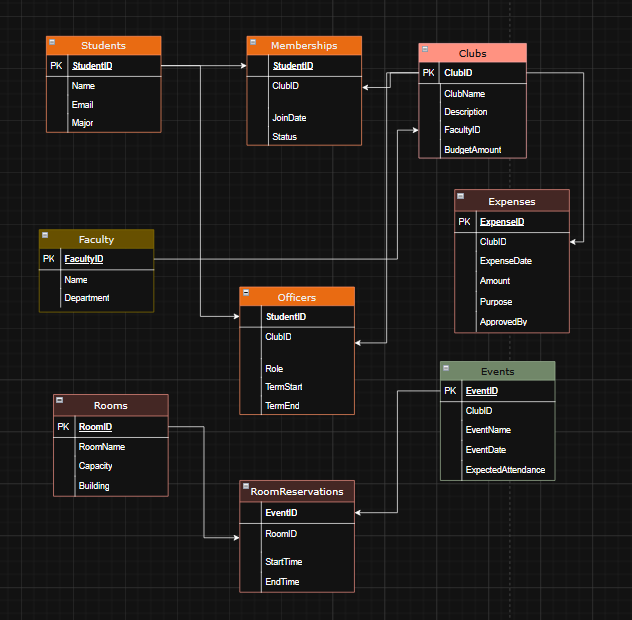
**Part 5: Design Challenge — Campus Organizations**

**Scenario Requirements:**

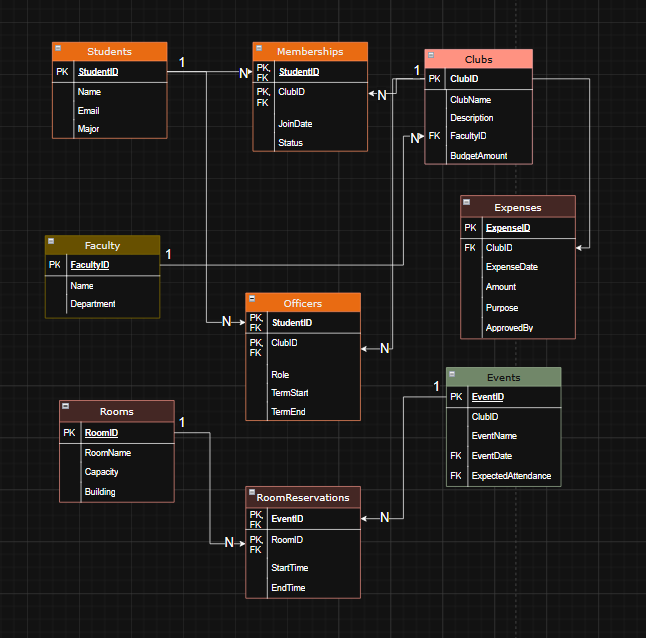
* + Track student clubs and societies, memberships, event scheduling, officer positions, faculty advisors, room bookings, and budgets.
  + Students may belong to multiple clubs; clubs have many members.
  + Clubs run events; events have attendees and may reserve rooms (room reservations include start/end times and expected attendance).
  + Clubs have officers (a student holding a role for a given term), and each club has exactly one faculty advisor (faculty may advise multiple clubs).
  + Budget tracking: clubs have budgets and record expenses (expense date, amount, purpose, approved-by).

**Your Tasks:**

* 1. Create a full ER diagram for the system (entities, relationships, cardinalities, and keys).



* 1. Convert the ER model to a normalized relational schema (at least to 3NF). Include primary keys, foreign keys, and suggested data types.



* 1. Identify one design decision with multiple valid alternatives (e.g., modeling officers as attributes vs. as a relationship table) and justify your chosen approach.

Answer: Separate table of Officers(StudentID, ClubId, Role, TermStart, TermEnd). Flexible, allows to store any number of officers and different roles, suitable for M:N (many students, many clubs).

* 1. Write three example queries the system must support (in plain English), for example:

“Find all students who are officers in the Engineering Society” – “List all upcoming club events with room reservations and expected attendance”

Answer:   
Query 1: Find all clubs along with the names of their faculty advisors and total number of active members.

Query 2: List of all officers who have served in more than one club.  
Query 3: Show total expenses per club for the current semester.