# Database Systems Laboratory Work #1

KBTU

Alexandr Chernov
9-12-2025

## Part 1: Key Identification Exercises

### Task 1.1: Superkey and Candidate Key Analysis

#### *Relation A: Employee*

1) Superkeys: (EmpID, SSN), (EmpID, SSN, Email), (EmpID, SSN, Phone), (EmpID, SSN, Name), (EmpID, Name, Department), (EmpID, Phone, Name, Salary).
2) Candidate keys: (EmpID), (SSN).
3) I'd choose (EmpID) candidate key as primary key, because it's more convenient and reliably identifies an employe in this database (there're cannot be two identical EmpIDs).
4) It's depending on company's politics. If they're allow their employees to have use the same phone number, then yes. But, I think, it's very rare and therefore, in most cases, two employees cannot have the same phone number.

#### *Relation B: Course Registration*

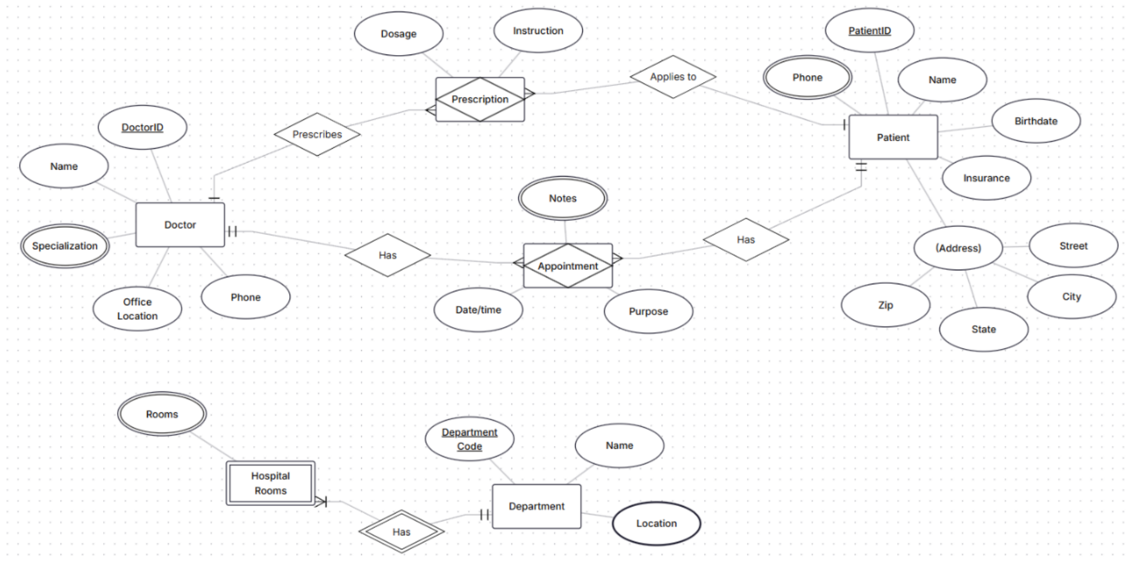
1) The minimum attributes needed for the primary key is: (StudentID, CourseCode, Section, Semester).
2) "StudentID" is essential to identify each student; attributes "Section" and "Semester" are needed, because student cannot register for the same course section in the same semester, and it's will be used to handle this; "CourseCode" is needed to distinguish sections. "Year" is not needed because there're no restrictions on taking the same course in different semesters or years. Yes, the same student can take the same course and the same section across the time, but to uniquely identify it we just need a "Semester". "Year" attribute is not needed as long as semesters do not repeats.
3) I think, there're no additional candidate keys.

### Task 1.2: Foreign Key Design

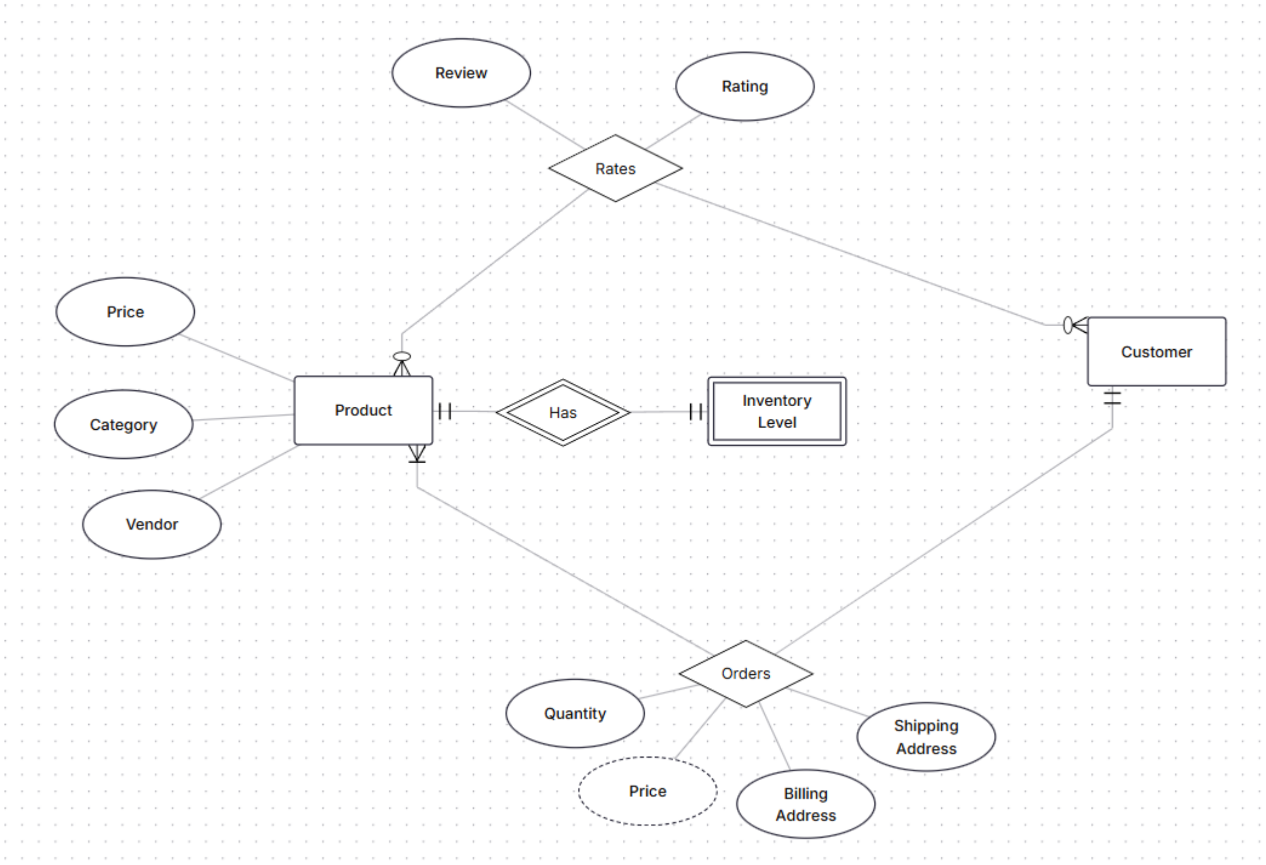
1) Foreign keys in relationships:
    a. Enrollment.StudentID → Student.StudentID;
    b. Enrollment.CourseID → Course.CourseID;
    c. Professor.Department → Department.DeptName;
    d. Course.DepartmentCode → Department.DeptCode

## Part 2: ER Diagram Construction

### Task 2.1: Hospital Management System



### Task 2.2: E-commerce Platform



2) Inventory Level is weak entity, because it's fully depended on product: if there're no product, then there's no inventory level for it.

3) For example, rating relationship is needs “Review” and “Rating” attributes by task’s requirement. It’s many-to-many, because one user can rate multiple products, and one product can have reviews from multiple users.

## Part 4: Normalization Workshop

### Task 4.1: Denormalized Table Analysis

1)
    a. StudentID → StudentName, StudentMajor;
    b. ProjectID → ProjectTitle, ProjectType, SupervisorID, StartDate, EndDate;
    c. SupervisorID → SupervisorName, SupervisorDept;
    d. (StudentID, ProjectID) → Role, HoursWorked.
2) Redundancy in naming attributes: [Project]ID, [Project]Title, [Project]Type; multiple students can be associated with one particular project, so all these project attributes will be repeated (the same will be with supervisor's).
   Delete anomaly: if there's only one student assigned with project, when this student dropped, all information about project will be lost. Update anomaly: when, for example, name of the project changes, we need to change it in multiple rows which leads us to the risk of inconsistency. Insert anomaly: can't add a new project unless at least one student is assigned.
3) There are no 1NF violations in this design: there is no chance to duplicate rows, each cell contains a single value, and each is atomic (can't be split down further).
4)
    a. The primary key of the StudentProject table is a composite key consisting of (StudentID, ProjectID). This is because a single student can work on multiple projects and a single project can be worked on by multiple students.
    b. "StudentName" and "StudentMajor" depend only on "StudentID", not on the full composite key (StudentID, ProjectID).
    c. "ProjectTitle", "ProjectType", "SupervisorID", "SupervisorName", and "SupervisorDept" depend only on "ProjectID".
    d. So, there're following modifications:
        i. Student(StudentID PK, StudentName, StudentMajor);
        ii. Project(ProjectID PK, ProjectTitle, ProjectType, SupervisorID, SupervisorName, SupervisorDept, StartDate, EndDate);
        iii. StudentProjectAssignment(StudentID FK, ProjectID FK, Role, HoursWorked).
    e. "SupervisorName" and "SupervisorDept" depend on "SupervisorID", and "SupervisorID" depends on "ProjectID". Final 3NF decomposition:
        i. Student(StudentID PK, StudentName, StudentMajor);
        ii. Project(ProjectID PK, ProjectTitle, ProjectType, SupervisorID FK, StartDate, EndDate);
        iii. Supervisor(SupervisorID PK, SupervisorName, SupervisorDept);

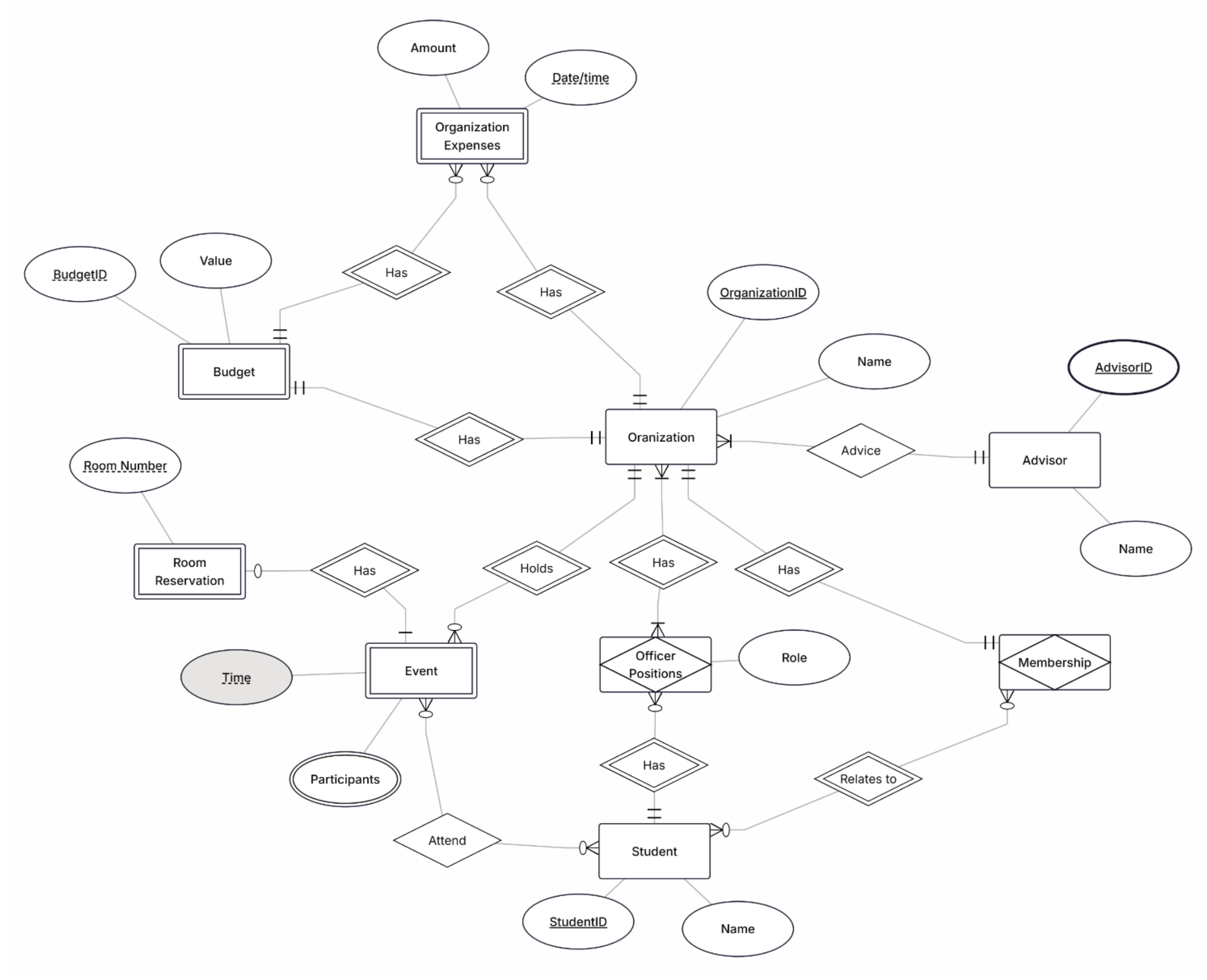
iv. StudentProjectAssignment(StudentID FK, ProjectID FK, Role, HoursWorked).

## Task 4.2: Advanced Normalization

1) Primary key is: (CourseID, TimeSlot, Room).
2) Functional dependencies:
    a. StudentID → StudentMajor;
    b. CourseID → CourseName;
    c. InstructorID → InstructorName.
3) This table is certainly not in BCNF.
4) Decomposition to BCNF:
    a. Student(StudentID PK, StudentMajor);
    b. Course(CourseID PK, CourseName);
    c. Instructor(InstructorID PK, InstructorName);
    d. CourseSection(CourseSectionID PK, CourseID FK, TimeSlot, Room, InstructorID FK);
    e. Enrollment(StudentID PK FK, CourseSectionID PK FK).
5) There's potential loss of the building numbering: each room in  unique across campus, therefore there's no need to keep track of it.

## Part 5: Design Challenge

### Task 5.1: Real-World Application



3) In relationship between student and organization in terms of officer positions I had choices to do this through just relation with additional to this attribute called "Role" or do it through entity called "Officer Positions" which is associative (by StudentID and OrganizationID).
4) Examples queries:
    a. List all organizations with more than 50 members;
    b. List all members of the "BookJourney Club" organization;
    c. List all organizations that have had expenses in the last week;
    d. Find all students who has some role in "RoboClub" organization and participated in club's last three events.