Lab 1

Part 1: Key Identification Exercises

Task 1.1: Superkey and Candidate Key Analysis

Your Tasks: 1. List at least 6 different superkeys 2. Identify all candidate keys 3. Which candidate key

would you choose as primary key and why? 4. Can two employees have the same phone number? Justify

your answer based on the data shown.

1. Superkeys:

[EmpID]

[SSN]

[Email]

[EmpID, phone]

[SSN, email]

[name, department, phone]

1. Candidate keys:

[empID]

[SSN]

[Email]

1. It’s better to choose [empID] because it can be primary key. It is especially created to be unique, and 2 things can’t be the same basically like SSN email Salary etc.
2. Yes, the phone number can be match between 2 workers in real life so it’s better to not be chosen as primary key.

Relation B: Course Registration

Registration(StudentID, CourseCode, Section, Semester, Year, Grade, Credits)

Business Rules:

- A student can take the same course in different semesters

- A student cannot register for the same course section in the same semester

- Each course section in a semester has a fixed credit value

Your Tasks: 1. Determine the minimum attributes needed for the primary key 2. Explain why each

attribute in your primary key is necessary 3. Identify any additional candidate keys (if they exist)

1. Minimum atributes: [StudentID, CourseCode, Section, Semester, Year]
2. So the StudentID for identification, CourseCode is for identification of the course, Section is for the distinguishing different section on the same course, Semester and Year is just for specifying.
3. You actually can add any surrogate keys for the candidate key or it can be if you combine Year and semester.

Task 1.2: Foreign Key Design

Design the foreign key relationships for this university system:Given Tables:

Student(StudentID, Name, Email, Major, AdvisorID)

Professor(ProfID, Name, Department, Salary)

Course(CourseID, Title, Credits, DepartmentCode)

Department(DeptCode, DeptName, Budget, ChairID)

Enrollment(StudentID, CourseID, Semester, Grade)

Your Tasks: 1. Identify all foreign key relationships

Student.AdvID – Professsor.ProfID

Course.DepartmentCode – Department.DeptCode

Department.ChairID – Professor.ProfID

Enrollment.StudentID – Student.StudentID

Enrollment.CourseID – Course.CourseID

Part 2: ER Diagram Construction

Task 2.1: Hospital Management System

Scenario: Design a database for a hospital management system.

Requirements:

• Patients have unique patient IDs, names, birthdates, addresses (street, city, state, zip), phone

numbers (multiple allowed), and insurance information

• Doctors have unique doctor IDs, names, specializations (can have multiple), phone numbers, and

office locations

• Departments have department codes, names, and locations

• Appointments track which patient sees which doctor at what date/time, the purpose of visit, and

any notes

• Prescriptions track medications prescribed by doctors to patients, including dosage and

instructions

• Hospital Rooms are numbered within departments (room 101 in Cardiology is different from

room 101 in Neurology)

Your Tasks: 1. Identify all entities (specify which are strong and which are weak) 2. Identify all attributes

for each entity (classify as simple, composite, multi-valued, or derived) 3. Identify all relationships with

their cardinalities (1:1, 1:N, M:N) 4. Draw the complete ER diagram using proper notation 5. Mark

primary keys

1. Patients, Doctors, Departments is strong, Appointments Prescriptions is partly strong, Hospital Rooms is weak

**Patient:**

* PatientID (PK)
* Name
* Birthdate
* Address (композитный: street, city, state, zip)
* Phone (мультизначный)
* InsuranceInfo

**Doctor:**

* DoctorID (PK)
* Name
* Specialization (мультизначный)
* Phone (мультизначный)
* OfficeLocation

**Department:**

* DeptCode (PK)
* DeptName
* Location

**Appointment:**

* AppointmentID (PK)
* DateTime
* Purpose
* Notes

**Prescription:**

* PrescriptionID (PK)
* MedicationName
* Dosage
* Instructions

**Room (weak entity):**

* RoomNumber (partial key)
* DeptCode (FK, часть составного PK)

 **Patient — Appointment — Doctor**:

* Patient (1) - (N) Appointment
* Doctor (1) - (N) Appointment

 **Doctor — Prescription — Patient**:

* Doctor (1) - (N) Prescription
* Patient (1) - (N) Prescription

 **Department — Doctor**:

* Department (1) - (N) Doctor

 **Department — Room**:

* Department (1) - (N) Room (Room — слабая сущность, PK = (DeptCode, RoomNumber))

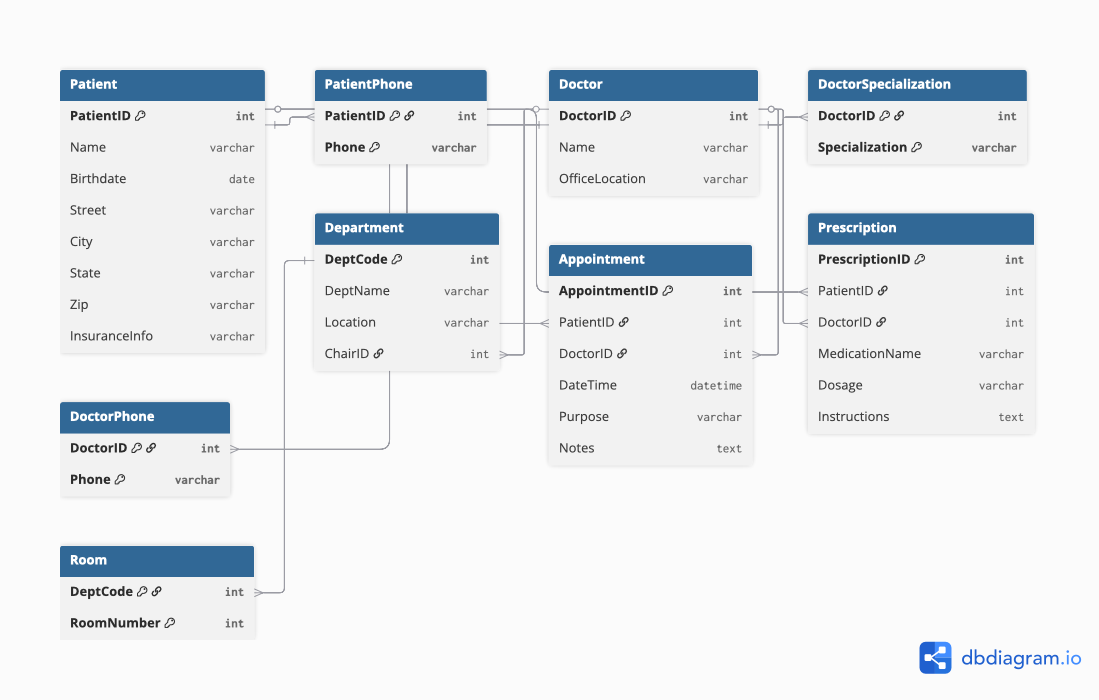


Table Patient {

PatientID int [pk]

Name varchar

Birthdate date

Street varchar

City varchar

State varchar

Zip varchar

InsuranceInfo varchar

}

Table PatientPhone {

PatientID int [pk, ref: > Patient.PatientID]

Phone varchar [pk]

}

Table Doctor {

DoctorID int [pk]

Name varchar

OfficeLocation varchar

}

Table DoctorSpecialization {

DoctorID int [pk, ref: > Doctor.DoctorID]

Specialization varchar [pk]

}

Table DoctorPhone {

DoctorID int [pk, ref: > Doctor.DoctorID]

Phone varchar [pk]

}

Table Department {

DeptCode int [pk]

DeptName varchar

Location varchar

ChairID int [ref: > Doctor.DoctorID]

}

Table Appointment {

AppointmentID int [pk]

PatientID int [ref: > Patient.PatientID]

DoctorID int [ref: > Doctor.DoctorID]

DateTime datetime

Purpose varchar

Notes text

}

Table Prescription {

PrescriptionID int [pk]

PatientID int [ref: > Patient.PatientID]

DoctorID int [ref: > Doctor.DoctorID]

MedicationName varchar

Dosage varchar

Instructions text

}

Table Room {

DeptCode int [pk, ref: > Department.DeptCode]

RoomNumber int [pk]

}

Task 2.2: E-commerce Platform

Scenario: Design a simplified e-commerce database.

Requirements:

• Customers place Orders for Products

• Products belong to Categories and are supplied by Vendors

• Orders contain multiple Order Items (quantity and price at time of order)

• Products have reviews and ratings from customers

• Track Inventory levels for each product

• Shipping addresses can be different from customer billing addresses

1 task

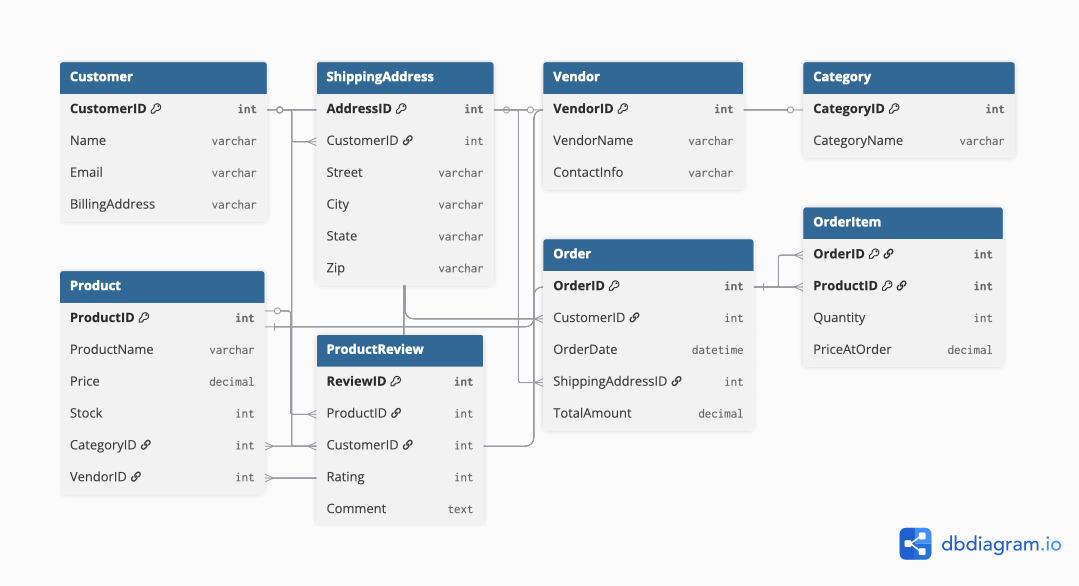


Table Customer {

CustomerID int [pk]

Name varchar

Email varchar

BillingAddress varchar

}

Table ShippingAddress {

AddressID int [pk]

CustomerID int [ref: > Customer.CustomerID]

Street varchar

City varchar

State varchar

Zip varchar

}

Table Vendor {

VendorID int [pk]

VendorName varchar

ContactInfo varchar

}

Table Category {

CategoryID int [pk]

CategoryName varchar

}

Table Product {

ProductID int [pk]

ProductName varchar

Price decimal

Stock int

CategoryID int [ref: > Category.CategoryID]

VendorID int [ref: > Vendor.VendorID]

}

Table ProductReview {

ReviewID int [pk]

ProductID int [ref: > Product.ProductID]

CustomerID int [ref: > Customer.CustomerID]

Rating int

Comment text

}

Table Order {

OrderID int [pk]

CustomerID int [ref: > Customer.CustomerID]

OrderDate datetime

ShippingAddressID int [ref: > ShippingAddress.AddressID]

TotalAmount decimal

}

Table OrderItem {

OrderID int [pk, ref: > Order.OrderID]

ProductID int [pk, ref: > Product.ProductID]

Quantity int

PriceAtOrder decimal

}

Task 2

OrderItem is weak because it depends on will there be OrderID and ProductID

Task 3

Order – Product many to many bc there can be one product that ordered 1000 times

Product – Customer

Part 4: Normalization Workshop

Task 4.1: Denormalized Table Analysis

Given Table:

StudentProject(StudentID, StudentName, StudentMajor, ProjectID, ProjectTitle,

ProjectType, SupervisorID, SupervisorName, SupervisorDept,

Role, HoursWorked, StartDate, EndDate)

Your Tasks: 1. Identify functional dependencies: List all FDs in the format A → B 2. Identify

problems: - What redundancy exists in this table? - Give specific examples of update, insert, and delete

anomalies 3. Apply 1NF: Are there any 1NF violations? How would you fix them? 4. Apply 2NF: - What is

the primary key of this table? - Identify any partial dependencies - Show the 2NF decomposition 5. Apply

3NF: - Identify any transitive dependencies - Show the final 3NF decomposition with all table schemas

1 task

StudentID → StudentName, StudentMajor

ProjectID → ProjectTitle, ProjectType

SupervisorID → SupervisorName, SupervisorDept

(StudentID, ProjectID) → Role, HoursWorked, StartDate, EndDate

2 task

**Update:** if a student changes their major, all rows must be updated.

**Insert:** it is not possible to add a new student without a project, or a new project without a student.

**Delete:** if the last student participating in a project is deleted, the data about the project and the supervisor is also lost.

3 task

Table doesnt have 1NF violations there is no matching groups

4 task

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

5 task

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

Task 4.2: Advanced Normalization

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, decompose it to BCNF showing your work

5. Explain any potential loss of information in your decomposition

1 task

Primary key is StudentID

2 task

StudentID → StudentMajor

CourseID → CourseName, InstructorID, TimeSlot, Room

InstructorID → InstructorName

Room → Building

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

3 task

No BCNF keys depend to each other

4 task

**Student(StudentID [PK], StudentMajor)**

**Instructor(InstructorID [PK], InstructorName)**

**Room(Room [PK], Building)**

**Course(CourseID [PK], CourseName, InstructorID [FK], TimeSlot, Room [FK])**

**Enrollment(StudentID [PK, FK], CourseID [PK, FK])**

**5 task**

**Actually no risks because we just splitted and created multiple tables BCNF but all thing are the same**

**This recomposition of table imporves inserting deleting and updating table protects from actual data loss**

Part 5: Design Challenge

Task 5.1: Real-World Application

Scenario: Your university wants to track student clubs and organizations with the following

requirements:

System Requirements:

• Student clubs and organizations information

• Club membership (students can join multiple clubs, clubs have multiple members)

• Club events and student attendance tracking

• Club officer positions (president, treasurer, secretary, etc.)

• Faculty advisors for clubs (each club has one advisor, faculty can advise multiple clubs)

• Room reservations for club events

• Club budget and expense trackingYour Tasks: 1. Create a complete ER diagram for this system 2. Convert your ER diagram to a normalized

relational schema 3. Identify at least one design decision where you had multiple valid options and

explain your choice 4. Write 3 example queries that your database should support (in English, not SQL)

Example Query Format: - “Find all students who are officers in the Computer Science Club” - “List all

events scheduled for next week with their room reservations”

1 task

**Student**(StudentID, Name, Email, Major, Year)

**Club**(ClubID, ClubName, Description, Budget, AdvisorID [FK])

**FacultyAdvisor**(AdvisorID, Name, Department, Email)

**Membership**(StudentID [FK], ClubID [FK], JoinDate, Role) ← связь Student–Club

**Event**(EventID, ClubID [FK], Title, Date, RoomID [FK])

**Attendance**(EventID [FK], StudentID [FK]) ← кто был на мероприятии

**OfficerPosition**(ClubID [FK], StudentID [FK], PositionTitle, StartDate, EndDate)

**Room**(RoomID, Location, Capacity)

**Expense**(ExpenseID, ClubID [FK], Amount, Category, Date, Description)

2 task

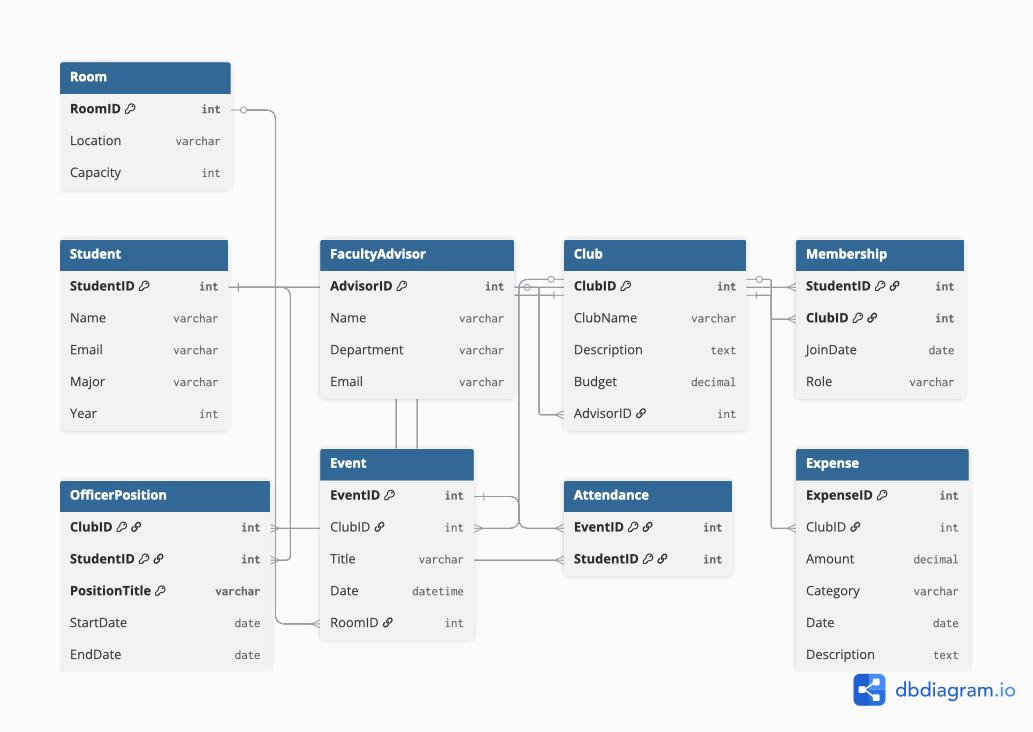


Table Student {

StudentID int [pk]

Name varchar

Email varchar

Major varchar

Year int

}

Table FacultyAdvisor {

AdvisorID int [pk]

Name varchar

Department varchar

Email varchar

}

Table Club {

ClubID int [pk]

ClubName varchar

Description text

Budget decimal

AdvisorID int [ref: > FacultyAdvisor.AdvisorID]

}

Table Membership {

StudentID int [pk, ref: > Student.StudentID]

ClubID int [pk, ref: > Club.ClubID]

JoinDate date

Role varchar

}

Table OfficerPosition {

ClubID int [pk, ref: > Club.ClubID]

StudentID int [pk, ref: > Student.StudentID]

PositionTitle varchar [pk]

StartDate date

EndDate date

}

Table Room {

RoomID int [pk]

Location varchar

Capacity int

}

Table Event {

EventID int [pk]

ClubID int [ref: > Club.ClubID]

Title varchar

Date datetime

RoomID int [ref: > Room.RoomID]

}

Table Attendance {

EventID int [pk, ref: > Event.EventID]

StudentID int [pk, ref: > Student.StudentID]

}

Table Expense {

ExpenseID int [pk]

ClubID int [ref: > Club.ClubID]

Amount decimal

Category varchar

Date date

Description text

}

Normalized schema

**Student**(StudentID **PK**, Name, Email, Major, Year)  
**FacultyAdvisor**(AdvisorID **PK**, Name, Department, Email)  
**Club**(ClubID **PK**, ClubName, Description, Budget, AdvisorID **FK**)  
**Membership**(StudentID **PK, FK**, ClubID **PK, FK**, JoinDate, Role)  
**OfficerPosition**(ClubID **PK, FK**, StudentID **PK, FK**, PositionTitle **PK**, StartDate, EndDate)  
**Room**(RoomID **PK**, Location, Capacity)  
**Event**(EventID **PK**, ClubID **FK**, Title, Date, RoomID **FK**)  
**Attendance**(EventID **PK, FK**, StudentID **PK, FK**)  
**Expense**(ExpenseID **PK**, ClubID **FK**, Amount, Category, Date, Description)

Task 3

I considered two options for modeling officer positions. The first option was to include the role directly in the Membership table, so that membership and officer roles would be stored together. The second option was to create a separate OfficerPosition table and link it to Membership. I chose the second option because a student can hold multiple officer roles at different times, and it is easier to track the start and end dates of each role separately. This design is more flexible and scalable.

Task 4

**1 Find all students who are officers in the Computer Science Club.**

**2 List all events scheduled for next week with their room reservations.**

**3 Show the total expenses for each club in the current semester.**