Database Systems Laboratory Work

Part 1: Key Identification Exercises

Task 1.1: Employee Relation

Superkeys: {EmpID}, {SSN}, {Email}, {EmpID,Name}, {Phone,EmpID}, {SSN,Email}
Candidate Keys: {EmpID}, {SSN}, {Email}, {Phone}
Primary Key choice: EmpID
Can two employees have the same phone?

Based on the provided sample data, two employees cannot have the same phone number. The Phone column values are unique for each employee shown (555-0101, 555-0102, 555-0103), which makes Phone a candidate key. However, in a real-world scenario, two employees could potentially share a home or company phone number, which is a limitation of using sample data to make assumptions about business rules.

Task 1.1: Course Registration

Minimum Attributes for Primary Key: The minimum attributes needed for the primary key are Student ID, CourseCode, and Semester

Necessity of Each Attribute:

Student ID is necessary because a single course can be taken by multiple students.

CourseCode is necessary because a single student can take multiple courses.

Semester is necessary because a student can take the same course again in a different semester.

Additional Candidate Keys: There are no additional candidate keys

Task 1.2: Foreign Keys

• Student.AdvisorID → Professor.ProfID
• Course.DepartmentCode → Department.DeptCode
• Department.ChairID → Professor.ProfID
• Enrollment.StudentID → Student.StudentID
• Enrollment.CourseID → Course.CourseID

Part 4: Normalization Workshop

Task 4.1:

Given Table: StudentProject (StudentID, StudentName, StudentMajor, ProjectID, ProjectTitle, ProjectType, SupervisorID, SupervisorName, SupervisorDept, Role, HoursWorked, StartDate, EndDate)

1. Functional Dependencies (FDs):

   StudentID → StudentName, StudentMajor

   ProjectID → ProjectTitle, ProjectType

   SupervisorID → SupervisorName, SupervisorDept

   StudentID, ProjectID → Role, HoursWorked, StartDate, EndDate

2. Problems:

   Redundancy: Student and project details are repeated for each project a student is on.

   Update Anomaly: A student's major must be updated in multiple places.

   Insert Anomaly: A new project cannot be added without a student being assigned to it.

   Delete Anomaly: Deleting the last student from a project removes all project information.

3. Apply 1NF: The table is in 1NF as it has no repeating groups.

4. Apply 2NF:

   Primary Key: (StudentID, ProjectID)

   Partial Dependencies:

   StudentID → StudentName, StudentMajor

   ProjectID → ProjectTitle, ProjectType

   2NF Decomposition:

   Student (StudentID, StudentName, StudentMajor)

   Project (ProjectID, ProjectTitle, ProjectType)

StudentProject (StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate, SupervisorID)

5. Apply 3NF:

   Transitive Dependency: SupervisorID → SupervisorName, SupervisorDept

   3NF Decomposition:

   Student (StudentID, StudentName, StudentMajor)

   Project (ProjectID, ProjectTitle, ProjectType)

   Supervisor (SupervisorID, SupervisorName, SupervisorDept)

   StudentProject (StudentID, ProjectID, SupervisorID, Role, HoursWorked, StartDate, EndDate)

Task 4.2:

Given Table: CourseSchedule (StudentID, StudentMajor, CourseID, CourseName, InstructorID, InstructorName, TimeSlot, Room, Building)

1. Primary Key: (StudentID, CourseID, TimeSlot, Room)
2. Functional Dependencies (FDs):

   StudentID → StudentMajor

   CourseID → CourseName

   InstructorID → InstructorName

   TimeSlot, Room → Building

   CourseID, TimeSlot, Room → InstructorID

3. Check if BCNF: The table is not in BCNF because determinants like StudentID, CourseID, and TimeSlot, Room are not superkeys.
4. BCNF Decomposition:

   Student (StudentID, StudentMajor)

   Course (CourseID, CourseName)

   Instructor (InstructorID, InstructorName)

   Room (TimeSlot, Room, Building)

   Section (CourseID, TimeSlot, Room, InstructorID)

Enrollment (StudentID, CourseID, TimeSlot, Room)

5. Loss of Information: The decomposition is lossless, as the original data can be fully reconstructed by joining the decomposed tables.

Task 5.1:

3. Design Decision: I chose to model the officer role using a separate associative table (ClubOfficer) with a foreign key to an OfficerPosition table. This is better than including an officer role attribute in the Membership table. This design decision offers more flexibility and avoids redundancy, as a student can hold multiple officer roles in the same club without creating separate membership records.

4. Example Queries

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

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

Show the total budget for **each club, categorized by income and expenses.**