Database Systems Laboratory Work #1

Part 1.

Task 1.1.

Relation A: Employee

Employee(EmpID, SSN, Email, Phone, Name, Department, Salary)

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

1) A **superkey** by definition is any set of attributes whose values uniquely identify every tuple in the relation instance. The attributes that can be a superkey from the given sample data are:

1. EmpID
2. SSN
3. Email
4. Phone

Also, superkey can be described as any superset of candidate key, that’s why we can make a superkeys like:

1. {EmpId, Name}
2. {Email, Department}

and others.

2) A candidate key is a minimal superkey, it doesn’t have any unnecessary attributes. They are:

1. EmpID

2.SSN

3. Email

4. Phone

3) **Which candidate key would I choose as primary key and why?**

I would choose {EmpID} as my primary key and there are several reasons for it. First of all, EmpID is literally designed and usually assigned by system to use it as primary key and to be controlled by your organization. Also, EmpID is unlikely to change over time, rather than Phone number or email. EmpID is simple and convinient to use, because it consists only of integers. I wouldn’t choose SSN, because it is sensitive personal data, so it often shouldn’t be used as primary key.

4) **Can two employees have the same phone number?**

From the sample data, we can see that no employee share the same phone number. In real-life, every phone number is unique, based on the region, operator and etc. So, i assume that two employees can’t have the same phone number.

Relation B: Course Registration

1)The minimum attributes needed for the primary key. That would be:

{StudentID, CourseCode, Section, Semester, Year}.

2)Each one of them is needed to create the primary key. StudentID – to identify which student is taking the course. CourseCode – to identify what course is getting picked. Section is necessary because a course can have multiple sections in the same semester. We need semester attribute to differentiate the fall, spring and summer semesters. And, we need a Year attribute to keep track on academic years.

3)We could have remove the StudentID from the primary key above and call it candidate key for course offers, but it can’t be Registration’s candidate key, because it doesn’t identify a unique registration, multiple students can register for the same course section in the same semester. Thats why, {StudentID, CourseCode, Section, Semester, Year} is the only candidate key for Registration.

Task 1.2: Foreign Key Design

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)

Identify all foreign key relationships.

In Student table: AdvisorID 🡪 Professor(ProfID). I assumed, that advisor and professor are the same person and that AdvisorID points to the ProfID, and because one professor can advise many students, it is a FK. In this schema there are several attributes that look like they are meant to be foreign keys, such as Professor.Department, Course.DepartmentCode, Student.Major, and Student.AdvisorID. However, because their names do not exactly match the primary keys of other tables, we cannot be completely certain.

In Enrollment: StudentID 🡪 Student(StudentID), CourseID 🡪 Course(CourseID).

Task 4.1: Denormalized Table Analysis

The table does not violate First Normal Form (1NF). All attributes contain single, atomic values, and there are no repeating groups of data.

Apply 2NF: The primary key for this table is the composite key (StudentID, ProjectID).

To achieve 2NF, we must rewrite 3 tables:

1. Students**(**StudentID, StudentName, StudentMajor**)**
2. Projects**(**ProjectID, ProjectTitle, ProjectType, SupervisorID**)**
3. Assignments**(**StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate**)**

Apply 3NF: To achieve it, we have to add Supervisors table, because there was a transitive dependency in ProjectID → SupervisorID → SupervisorName, SupervisorDept.

Students(StudentID, StudentName, StudentMajor**)** PK: StudentID

Supervisors(SupervisorID, SupervisorName, SupervisorDept) PK:SupervisorID

Projects**(**ProjectID, ProjectTitle, ProjectType, SupervisorID**)** PK:ProjectID, FK:SupervisorID

Assignments**(**StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate**)** PK:{StudentID, ProjectID}

FK:StudentID, ProjectID