15.564 Information Technologies

Database Design Review

DATABASE DESIGN EXAMPLE

The Sloan Career Development Office has hired you as a consultant to help design a relational database for the Sloan career development office. The database will help them maintain placement statistics for graduating MBA students. The office wants to use the database to be able to answer at least the following questions:

1. Who is our recruiting contact at Goldman Sachs? What is her telephone number?
2. What was the average number of scheduled interviews per student in each of the past 5 years?
3. Which were the top industries selected by graduating Sloan students in 2002? Which were the top job functions selected? (Assuming that the offers were all accepted by students)
4. Which companies were the top ten interviewers last year? How many interviews did each company conduct? How many people did it finally hire? (Assuming that the offers were all accepted by students)
5. Overall, what was the maximum, minimum and average salary of positions that students were offered? How about those that they did not receive an offer? (Assuming that the offers were all accepted by students)
6. For every industry type, what was the maximum, minimum, and average salary of positions that students were interviewed for?

The Career Development Office couldn't afford your high consulting fees for the entire design project, so a certain Professor Dell or something like that created an initial design and you have been asked to clean up the mess he left behind. Being a proponent of minimalist design, he sketched out a single data table and said that identifying keys and foreign keys was an exercise for the reader.

Placement Stats

Student Id	
Last Name	
First Name	
Middle Initial	
Year	← Graduation year
Company Name1	← Name of first company that scheduled interview
Industry1	← Industry sector of first company
Position1	← Title of position offered
Salary1	← Position Salary
Offer1	← Offer extended? (yes/no)
Company Name2	← Name of second company that scheduled interview
Industry2	← Industry sector of second company
Position2	← Title of position offered
Salary2	← Position Salary
Offer2	← Offer extended? (yes/no)

1. Briefly critique the present form of the database. In your answer describe at least three reasons why the current database makes it impossible or very awkward to answer some of the previously outlined questions.

The basic problem is that a single table is used to encode information about three different entities:

- Students
- Companies
- Offers

Here are some practical problems which are consequences of this shortcoming:

1. There are no fields for storing recruiting contacts of companies or additional information associated with a company. Even if there were such fields, a given company might be mentioned in several records (because it would have made offers to several students). On which record would we then store the company-specific information?
2. There is currently room for 2 offers per student maximum. This wastes a lot of space in the case of students with 1 or 0 interviews but, more realistically (for Sloan MBAs at least!!!), it does not allow us to record information about students who are scheduled 3 or more interviews in a single record.
3. All the queries for calculating answers to the questions of the preceding page would become very awkward and complex. For example, to calculate the average number of offers per student we would have to do the following:
   a) run a query to calculate the total number of students
   b) run a query to calculate the total number of interviews in the "first company" part of the record (a lot might be blank)
   c) run a query to calculate the total number of interviews in the "second company" part of the record (a lot might be blank)
   d) add the results of the previous queries together
   e) divide the total interviews by the number of students

2. Propose a new design for the database that solves the problems you identified above. In your answer show all tables and fields of your improved design. Also, show all primary keys, and any foreign keys that you defined.

Primary keys are shown in **bold.**
Foreign keys are shown in *italics.*
Fields which are members of a primary key but also foreign keys of other tables are shown in ***bold and italics.***

Students

Student Id	← key
Last Name	
First Name	
Middle Initial	
Year	
Other student info….	← We could define additional fields such as Address, student concentration, etc.

Companies

Company Id	
Company Name	
Industry	← If we felt that there could be ambiguity in specifying industry names in a standardized way, we could have added a separate table which encodes industry names and connected it to table Companies through an Industry Id
Other company info…	← headquarters address, number of employees, etc.

Company Contacts

← A separate table for company contacts is necessary if we assume that there might be multiple contact persons for each company. If there was only one contact person per company, we could have merged this information into the corresponding record of the table Companies

Contact Id
Company Id
Last Name
First Name
Position
Address
Telephone
Fax
Email
Notes

Interviews

The key for this table is the combination (Student Id, Company Id, Position Offered). It assumes that it is possible to receive more than one offers from the same company (for different positions).

Student Id	
Company Id	
Position Interviewed	← Again, if we needed a standardized way to refer to positions, we could have created a separate table with various Positions and their corresponding Position Ids.
Position Salary	
Offered?	

Write SQL queries to extract the following information from your new, improved database.

3. List the average number of scheduled interviews per student for each year for which data is kept in the database.

First, write a query TOT_STUDENTS to calculate the total number of students in each class:

```
SELECT [Year], COUNT(*) As [Students]
FROM [Students]
GROUP BY [Year];
```

Then, write a query TOT_INTERVIEWS to calculate the total number of conducted interviews in each class. We need to join with table Students to get access to the graduation year of each student:

```
SELECT [Students].[Year], COUNT(*) As  [Interviews]
FROM [Students], [Interviews]
WHERE [Students].[Student Id] = [Interviews].[Student Id]
GROUP BY [Students].[Year];
```

Finally, do a join on the results of the previous two queries to calculate the final result:

```
SELECT TOT_ INTERVIEWS.Year,
        (TOT_INTERVIEWS. Interviews / TOT_STUDENTS.Students) As [Interviews per Student]
FROM TOT_ INTERVIEWS, TOT_STUDENTS
WHERE TOT_ INTERVIEWS.Year = TOT_STUDENTS.Year;
```

4. List the 10 top hirers of the class of 1999. The output of your query should be as follows (#People who were extended an offer means: number of people who have interviewed and received an offer):

Company Name	# People who were Extended Offers	Minimum Salary	Maximum Salary	Average Salary

```
SELECT TOP 10 Companies.[Company Name], Count(*) As [# People Hired], Min(Offers.Salary) As [Minimum
Salary], Max(Offers.Salary) As [Maximum Salary], Avg(Offers.Salary) As [Average Salary]
FROM Companies, Interviews, Students
WHERE Companies.[Company Id] = Interviews.[Company Id] AND
        Interviews.[Student Id] = Students.[Student Id] AND
        Interviews.Offered = "YES" AND
        Students.Year = 1999
GROUP BY Companies.[Company Name]
ORDER BY [# People Hired] DESC;
```