

Solutions for Some of the Queries Problems in Lab 3-4

3.4.1. The following queries are based on the *Supplier* database that you have created in Labwork 2.

- Get total number of parts supplied by supplier S1.

A solution: Here, we certainly should use the `count` function.

```
Select count (*) from SupplyPart where SupplierId='S1';
```

- Get the total quantity of part P1 supplied by S1.

A solution: Most of you fell for this one, as S1 might supply P1 to multiple projects. If you do read the textbook, you will find out that there is also a `sum` function.

Thus, we should simply say

```
Select sum (QTY) from SupplyPartTo where SupplierId='S1' and PartId='P1';
```

- Get supplier names for those with status less than the current maximum status in the Supplier table.

A solution: We can use the `max` function to find out the maximum status of all the suppliers.

```
Select SNAME from Supplier
where STATUS < (Select max(STATUS) from Supplier);
```

5.10. (a) List all courses that are taught by professors who belong to the EE or MGT departments.

A solution: What we will firstly get is the professors belonging to either EE or MGT departments, then join the result with the `Teaching` table to get the tuples for the courses that those professors teach. Finally, we do a projection on the desired attributes.

Thus, an algebraic expression for this problem is as follows:

$$\pi_{CrsCode, Semester}(\sigma_{DeptId='EE' \text{ OR } DeptId='MGT'}(Professor) \bowtie_{Id=ProfId} Teaching)$$

```
Select T.CrsCode, T.Semester
From Teaching T, Professor P
Where T.ProfId=P.Id AND (P.DeptId = 'EE' OR P.DeptId = 'MGT')
```

(b) List the names of all students who took courses both in spring 1997 and fall 1998.

A solution: The following essentially says that we want to pick up those students such that the only courses they took in either in 'S1997' or 'F1998', where `Sems` contains only one attribute `Semester` and two tuples ('S1997') and ('F1998').

```

Select Distinct S.Name
From Student S, Transcript T1, Transcript T2
WHERE T1.StudId = S.Id AND T2.StudId = S.Id And
(T1.Semester ='S1997' AND T2.Semester = 'F1998')

```

- (c) List the names of all students who took courses from at least two professors in different departments.

A solution: We only give a SQL solution, which can be done in two stages, for clarity. First, we create an auxiliary view:

```

Create View StudDept(StudId, DeptId) AS
Select R.StudId, P.DeptId
From Transcript R, Teaching T, Professor P
Where R.CrsCode = T.Crscode AND R.Semester = T.Semester
      AND T.ProfId = P.Id

```

With this view, if a student takes courses from at least two professors in different departments, there will be at least two tuples for this view.

We now have the following:

```

Select S.Name
From Student S, StudDept SD1, StudDept SD2
Where S.Id = SD1.StudId AND SD1.StudId = SD2.StudId
      AND SD1.DeptId <> SD2.DeptId

```

The name of a student will be printed out if and only if there are at least two tuples in the view that share the same id.

- (d) List all courses that are offered by the MGT Department and that have been taken by all students.

A solution: We only give SQL solution again.

```

Select C.CrsCode
From Course C
Where C.DeptId = 'MGT' AND
      NOT EXISTS (
        (Select S.StudId
         From Student S)
      EXCEPT
        (Select T.StudId
         From Transcript T
         Where C.CrsCode = T.CrsCode) )

```

The solution to this one is almost the same as the example that we discussed in pages 69 through 73. The **Where** part essentially says that there is no such a student who has not taken this MGT course, namely, everyone has taken any arbitrary but fixed MGT course.

- 5.17. Express the following queries using SQL. Assume that the **Student** table is augmented with an additional attribute, **Age**, and that the **Professor** table has additional attributes, **Age** and **Salary**.

- (a) Find the average age of students who received an A for some course.

A solution:

```
Select AVG(S.Age)
From Student S
Where S.Id in
  (Select Distinct S.Id
   From Student S, Transcript T
   where S.Id =T.StudId AND T.Grade = 'A' )
```

Notice that since a student might get multiple 'A's, if we don't put in the `Distinct` clause, her age might be counted twice.

- (b) Find the minimum age among straight 'A' students per course.

A solution: The "Not Exist" structure simply says that there does not exist a course such that the student with `StudId` takes has a grade that is not 'A'.

```
Select T.CrsCode, MIN(S.Age)
From Student S, Transcript T
Where S.Id = T.StudId AND
  NOT EXISTS (Select T.Grade
              From Transcript T
              Where S.Id = T.StudId AND T.Grade<>'A' )
Group by T.CrsCode
```

At least two of you also gave me the following answer, which is not as twisted as mine, and I like it.

```
Select min(S.Age) AS MinAgeOfStraightAStudent
From Student S, Transcript T
Where S.Id = T.StudId and T.Grade ='A' and T.StudId Not In
  (Select Distinct T.StudId From Transcript T
   Where T.Grade In('B', 'C', 'D', 'F'))
```

- (c) Find the minimum age among straight 'A' students per course among the students who have taken CS305 or MAT123. (Hint: a `HAVING` clause might help.)

A solution: This is essentially the same as the previous one, except that we are only interested in those two specific courses. We thus simply add the following `Having` clause to the above query:

```
Having S.CrsCode IN ('CS305', 'MAT123')
```

- (d) Raise by 10% the salary of every professor who is now younger than 40 and who taught MAT123 in the spring 1997 or fall 1997 semester. (Hint: Try a nested subquery in the `WHERE` clause of the `UPDATE` statement.)

A solution: When we want to change a value, we use the `Update` clause.

```
Update Professor
Set Salary = Salary * 1.1
Where Id In
```

```
(Select P.Id  
From Professor P, Teaching T  
Where P.Id = T.ProfId AND P.Age < 40 AND T.CrsCode = 'MAT123'  
AND (T.Semester = 'S1997' OR T.Semester = 'F1997') );
```