Database Lab 2 - Querying Multiple Tables

Objectives

* + To provide experience in formulating nested queries
  + To provide practice in the use of multiple table queries
* To understand and overcome the limitations of SQL regarding nested queries

Approach

This lab worksheet provides a combination of initial queries for you to practice along with some queries to formulate yourselves as a part of your coursework. You can make use of the SQLite on Repl.it to develop and test your queries. Scroll down at the end of the document to the details of MSc coursework submission.

Setting up the test tables

To complete the example queries and problems provided in this worksheet you will have to set up a number of tables and import some data into them. Use the following declarations to create the tables:

CREATE TABLE Item (

ItemName VARCHAR (30) NOT NULL,

ItemType CHAR(1) NOT NULL,

ItemColour VARCHAR(10),

PRIMARY KEY (ItemName));

CREATE TABLE Employee (

EmployeeNumber SMALLINT UNSIGNED NOT NULL ,

EmployeeName VARCHAR(10) NOT NULL ,

EmployeeSalary INTEGER UNSIGNED NOT NULL ,

DepartmentName VARCHAR(10) NOT NULL REFERENCES Department,

BossNumber SMALLINT UNSIGNED NOT NULL REFERENCES Employee,

PRIMARY KEY (EmployeeNumber));

CREATE TABLE Department (

DepartmentName VARCHAR(10) NOT NULL,

DepartmentFloor SMALLINT UNSIGNED NOT NULL,

DepartmentPhone SMALLINT UNSIGNED NOT NULL,

EmployeeNumber SMALLINT UNSIGNED NOT NULL REFERENCES

Employee,

PRIMARY KEY (DepartmentName));

CREATE TABLE Sale (

SaleNumber INTEGER UNSIGNED NOT NULL,

SaleQuantity SMALLINT UNSIGNED NOT NULL DEFAULT 1,

ItemName VARCHAR(30) NOT NULL REFERENCES Item,

DepartmentName VARCHAR(10) NOT NULL REFERENCES Department,

PRIMARY KEY (SaleNumber));

CREATE TABLE Supplier (

SupplierNumber INTEGER UNSIGNED NOT NULL,

SupplierName VARCHAR(30) NOT NULL,

PRIMARY KEY (SupplierNumber));

CREATE TABLE Delivery (

DeliveryNumber INTEGER UNSIGNED NOT NULL,

DeliveryQuantity SMALLINT UNSIGNED NOT NULL DEFAULT 1,

ItemName VARCHAR(30) NOT NULL REFERENCES Item,

DepartmentName VARCHAR(10) NOT NULL REFERENCES Department,

SupplierNumber INTEGER UNSIGNED NOT NULL REFERENCES

Supplier,

PRIMARY KEY (DeliveryNumber));

The data for these tables is held in the files item.txt , employee.txt , department.txt , sale.txt , supplier.txt , and delivery.txt . Use INSERT INTO to enter all the data. There is also a bulk insert method, a video of which will be posted on Moodle in the Panopto area where you can find the other lecture videos.

For each table use the command SELECT \* FROM tablename to check that the data has been entered into each table correctly.

Some Queries to Try...

This section provides some examples of problems and their translation into an SQL query that will answer the problem. For each case, take time to identify how the query might have been produced from the problem statement, and to identify how the query operates. Then try out each query on the database.

1. Find the items sold by the departments on the second floor.

SELECT DISTINCT ItemName

FROM Sale, Department

WHERE Sale.DepartmentName = Department.DepartmentName AND Department.DepartmentFloor = 2;

Note that this query performs an equijoin on Sale and Department.

SELECT DISTINCT ItemName

FROM (Sale NATURAL JOIN Department)

WHERE Department.DepartmentFloor = 2;

Now replace NATURAL JOIN with JOIN in the same query. Was there any difference in the result? Why / Why not?

1. Identify by floor the items available on floors other than the second floor

SELECT DISTINCT ItemName, Department.DepartmentFloor AS 'On Floor'

FROM Delivery, Department

WHERE Delivery.DepartmentName = Department.DepartmentName AND

Department.DepartmentFloor <> 2

ORDER BY Department.DepartmentFloor, ItemName;

1. Find, for each department, the average salary of the employees in that department and report by descending salary.

SELECT DepartmentName, AVG(EmployeeSalary) AS 'Average

Salary'

FROM Employee

GROUP BY DepartmentName

ORDER BY 'Average Salary' DESC;

1. List the suppliers that deliver at least 10 items.

SELECT Supplier.SupplierNumber, Supplier.SupplierName

FROM Delivery, Supplier

WHERE Delivery.SupplierNumber = Supplier.SupplierNumber GROUP BY Supplier.SupplierNumber, Supplier.SupplierName HAVING COUNT(DISTINCT Delivery.ItemName) >= 10;

Note: The HAVING clause provides a simple form of WHERE clause on the GROUP BY part of the SELECT statement. Think of it as a WHERE clause that is applied to the result returned by the query, rather than to the data in the tables being queried. You cannot use it to compare with values that can only be computed through a further query. In these cases, a nested query would be required.

1. Count the number of direct employees of each manager

SELECT Boss.EmployeeNumber, Boss.EmployeeName, COUNT(\*) AS 'Employees'

FROM Employee AS Worker, Employee AS Boss

WHERE Worker.BossNumber = Boss.EmployeeNumber

GROUP BY Boss.EmployeeNumber, Boss.EmployeeName;

Note that this example makes use of Tuple Variables to distinguish between the Equi Join of Employee with itself when we are querying the recursive relationship on the Employee table.

1. Find, for each department that sells items of type 'E' the average salary of the employees.

SELECT Department.DepartmentName,

AVG(EmployeeSalary) AS 'Average Salary'

FROM Employee, Department, Sale, Item

WHERE Employee.DepartmentName = Department.DepartmentName AND Department.DepartmentName = Sale.DepartmentName AND Sale.ItemName = Item.ItemName

AND ItemType = 'E'

GROUP BY Department.DepartmentName;

Note that we could try to translate this query into one using joins. However, we would have to be careful in doing this, because the tables Employee and Department have both DepartmentName and EmployeeNumber in common (to establish two different relationships). We are therefore better to use an Equi-Join to establish this query.

1. Find the total number of items of type 'E' sold by departments on the second floor

SELECT SUM(SaleQuantity) AS 'Number of Items'

FROM Department, Sale, Item

WHERE Department.DepartmentName = Sale.DepartmentName AND Sale.ItemName = Item.ItemName

AND ItemType = 'E' AND

DepartmentFloor = '2';

Translate this query into its NATURAL JOIN equivalent, and demonstrate that your translation produces the same results.

1. What is the average delivery quantity of items of type 'N' delivered by each company?

SELECT Delivery.SupplierNumber, SupplierName, Delivery.ItemName, AVG(Delivery.DeliveryQuantity) AS 'Average Quantity'

FROM ((Delivery NATURAL JOIN Supplier) NATURAL JOIN Item)

WHERE Item.ItemType = 'N'

GROUP BY Delivery.SupplierNumber, SupplierName, Delivery.ItemName ORDER BY Delivery.SupplierNumber, SupplierName, 'Average Quantity' DESC, Delivery.ItemName;

Would re-ordering the JOINs create a more efficient query in this case? If the query were translated into an EQUI JOIN equivalent, could we produce a more efficient query? Try out your translation into the EQUI JOIN equivalent.

Nested Queries

Try to solve the questions below using nested queries. See if the answer given below matches the one you wrote. Study them carefully so that you understand how the queries are formulated.

1. What are the names of items sold by departments on the second floor? This was previously solved in the preceding section by the use of a join. However, it could be more efficiently solved by using an inner query:

SELECT DISTINCT ItemName

FROM Sale

WHERE DepartmentName IN

(SELECT DepartmentName

FROM Department

WHERE DepartmentFloor = 2);

2. Find the salary of Clare's manager.

SELECT EmployeeName, EmployeeSalary

FROM Employee

WHERE EmployeeNumber =

(SELECT BossNumber

FROM Employee

WHERE EmployeeName = 'Clare');

3. Find the name and salary of the managers with more than two employees

SELECT EmployeeName, EmployeeSalary

FROM Employee

WHERE EmployeeNumber IN

(SELECT BossNumber

FROM Employee

GROUP BY BossNumber HAVING COUNT(\*) > 2);

4. List the names of the employees who earn more than any employee in the Marketing department

SELECT EmployeeName, EmployeeSalary

FROM Employee

WHERE EmployeeSalary >

(SELECT MAX(EmployeeSalary)

FROM Employee

WHERE DepartmentName = 'Marketing');

5. Find the suppliers that deliver compasses and at least one other kind of item

SELECT DISTINCT Delivery.SupplierNumber, Supplier.SupplierName FROM (Supplier NATURAL JOIN Delivery)

WHERE (ItemName <> 'Compass' AND SupplierNumber IN (SELECT SupplierNumber

FROM Delivery

WHERE ItemName = 'Compass'));

MSc Coursework Questions

The following are problems for you to try to create queries for (the table definitions are here if you need them.) As part of your graded assignment, you will need to submit all six SQL queries from questions below.

Develop queries for the following...

1. What are the names of employees in the Marketing Department?
2. Find the average salary of the employees in the Clothes department.
3. List the items delivered by exactly one supplier (i.e. the items always delivered by the same supplier).

Develop nested queries for the following...

1. List the departments for which each item delivered to the department is delivered to some other department as well.
2. Among all the departments with a total salary greater than £25000, find the departments that sell Stetsons.
3. Find the suppliers that deliver compasses and at least three other kinds of item.

How to submit:

The coursework must be submitted via Moodle. All SQL queries along with the repl.it link to your code should be collected in a single .txt file with the filename YourBathUsername\_Lab\_2.txt. Make sure your repl.it file is not edited after the submission date and the code matches the one that is submitted in the .txt file.

Marking Scheme:

A total of 100 marks are awarded for this assignment. The first three queries are worth 10 marks each and the last three queries are worth 20 marks each. 10 marks are discretionarily reserved for the correct formatting and presentation of lab submission. Poor presentation and failing to adhere to the submission instructions (e.g., Wrong file type, wrong naming of files) may result in a penalty. Your submission file should contain the number of the query, the description (i.e., text) and the query.

Feedback:

Written individual feedback will be provided through Moodle within 3 weeks of the

submission deadline. Clarification of this feedback can always be obtained for the unit

lecturer.

You are encouraged to evaluate your answers before submission. All queries can be easily tested manually. If in doubt, ask one of the tutors in the lab.

Plagiarism:

As this is individual coursework and you need to be mindful about collusion. Collusion

which is a form of plagiarism where two or more people work together to produce a piece of

work all or part of which is then submitted by each of them as their own individual work.

Plagiarism is a serious academic offence but more often than not it results from

misunderstanding rather than a deliberate intention to cheat. Many students simply do not

understand what plagiarism is. Working The University of Bath and the Department of

Computer Science takes this offence very seriously. On additional information of what may be

considered plagiarism and different types of it, refer to the relevant Library resource.

(<http://www.bath.ac.uk/library/help/infoguides/plagiarism.html> )

Extension requests:

Please note that any requests for extensions to coursework deadlines should be submitted to the Director of Studies.