Chapter 6

SQL: SubQueries

Definition

- A subquery contains one or more nested Select statements
- Example: List the staff who work in the branch at '163 Main St'

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE branchNo = (SELECT branchNo
                  FROM Branch
                  WHERE street = '163 Main St');
```

Subquery Resolution

- If the branchNo corresponding to ‘163 Main St’ is ‘B003’, then the query resolves to:

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE branchNo = ‘B003’;
```

Equivalent Query with Join

- Sometimes, but not always, a subquery can be replaced with a join:

```
SELECT staffNo, fName, lName, position
FROM Staff, Branch
WHERE (Staff.branchNo =Branch.branchNo)
                AND (street = '163 Main St');
```

Types of Subqueries

- Scalar subquery: Returns a single value

  Example: See previous example

  Example: List all staff whose salary is greater than the average salary, and show by how much their salary is greater than the average

```
SELECT staffNo, fName, lName, position,
      salary - (SELECT AVG(salary) FROM Staff) AS salDiff
FROM Staff
WHERE salary > (SELECT AVG(salary) FROM Staff);
```

Subquery Resolution

- Suppose the average salary is 17,000.
- Then the query resolves as:

```
SELECT staffNo, fName, lName, position, salary – 17000
FROM Staff
WHERE salary > 17000;
```

Using SQL Variables

- **You can use SQL variables to store intermediate results**
  - **1) Limited to storing single values (i.e., scalar values)**
  - ***2) Cannot store tables***
  - **3) Prefix name with '@'**
- **Example**

```
SELECT @avgSalary := AVG(salary) FROM Staff;
SELECT staffNo, fName, lName, position, salary - @avgSalary AS salDiff
    FROM Staff
        WHERE salary > @avgSalary;
```

- **To suppress any output when you make the assignment:**

```
SELECT AVG(salary) INTO @avgSalary FROM Staff;
```

Types of Subqueries (Cont)

- Row subquery: Returns an entire relation
- Used to compute an intermediate result

Row Subquery Example

List the name and score of the top scoring student in each course.

- We can use a Group By query to get the top score in each class but cannot list the student name/id.
- Solution is to compute an intermediate result with the top scores in each class, then join this result with studentcourses to get student id's, and finally join this result with the student relation to get student names

```
SELECT sc.courseid, FirstName, LastName, topscore
        FROM (SELECT courseId, max(score) as topscore
                FROM studentcourses GROUP BY courseid) m
        JOIN studentcourses sc ON m.courseid = sc.courseid
                AND m.topscore = sc.score
        JOIN student s ON sc.studentid = s.id
        ORDER BY courseid;
```

Table Subquery

- Returns a table suitable for use with IN (only 1 column allowed in result returned by subquery)

Example: List the properties that are handled by staff who work in the branch at '163 Main St'

```
SELECT propertyNo, street, city, postcode, type, rooms, rent
FROM PropertyForRent
WHERE staffNo IN (SELECT staffNo
                  FROM Staff
                  WHERE branchNo = (SELECT branchNo
                                    FROM Branch
                                    WHERE street =
                                      '163 Main St');
```

How to Do It With a Join

```
SELECT propertyNo, street, city, postcode, type, rooms, rent
FROM PropertyForRent
WHERE staffNo IN (SELECT staffNo
                  FROM Staff
                  WHERE branchNo = (SELECT branchNo
                                    FROM Branch
                                    WHERE street =
                                      '163 Main St'));
```

- SELECT * FROM PropertyForRent p NATURAL JOIN staff s
  NATURAL Join Branch b
  WHERE b.street = '163 Main St';

Subquery Rules

- ORDER BY clause may not be used in a subquery
- The subquery SELECT list must consist of a single column name or expression (except row queries)
- By default, column names in a subquery refer to the table name in the FROM clause of the subquery.

  It is possible to refer to a table in a FROM clause of an outer query by qualifying the column name
- When a subquery is one of the two operands involved in a comparison, the subquery must be the right hand side operand

  Example: In the previous aggregate subquery, it would not be permissible to have written:

  WHERE (SELECT AVG(salary**) FROM** Staff**)** < salary**;**

Exercise

- List all guests currently staying at the Grosvenor Hotel

  Hotel (hotelNo, hotelName, city)

  Room (roomNo, hotelNo, type, price)

  Booking (hotelNo, guestNo, dateFrom, dateTo, roomNo)

  Guest (guestNo, guestName, guestAddress)

Exercise

- List all guests currently staying at the Grosvenor Hotel

Hotel	(hotelNo, hotelName, city)
Room	(roomNo, hotelNo, type, price)
Booking	(hotelNo, guestNo, dateFrom, dateTo, roomNo)
Guest	(guestNo, guestName, guestAddress)

```
SELECT * FROM Guest
        WHERE guestNo  IN (SELECT guestNo FROM Booking
                WHERE (CURRENT_DATE BETWEEN dateFrom AND
                  DATE_SUB(dateTo, INTERVAL 1 DAY) )
                        AND hotelNo = (SELECT hotelNo FROM Hotel
                                WHERE hotelName = 'Grosvenor Hotel'));
```

A Join Solution to the Exercise

- There is also a solution that involves join, without using a subquery

```
SELECT Guest.guestNo, Guest.guestName, Guest.guestAddr
FROM Guest, Booking, Hotel
WHERE (Guest.guestNo = Booking.guestNo)
    AND (CURRENT_DATE BETWEEN dateFrom AND
            DATE_SUB(dateTo, INTERVAL 1 DAY) )
    AND (Booking.hotelNo = Hotel.hotelNo)
    AND (hotelName = 'Grosvenor Hotel');
```