upGrad ENTERPRISE

Joins and Subqueries in SQL Server

By
Narasimha Rao T
Microsoft.Net FSD Trainer
Professional Development Trainer
tnrao.trainer@gmail.com

Introduction to Joins

A **Join** in SQL Server is used to combine rows from two or more tables based on a related column between them. Joins help retrieve data that is distributed across multiple tables.

Advantages of Joins

- **Data Consolidation:** Combines related data from multiple tables.
- **Efficiency:** Reduces data redundancy by avoiding repeated data.
- **Flexibility:** Enables complex queries over normalized databases.
- **Performance:** Typically more efficient than using subqueries in some cases.

Types of Joins in SQL Server

1. INNER JOIN
2. LEFT OUTER JOIN
3. RIGHT OUTER JOIN
4. FULL OUTER JOIN
5. SELF JOIN
6. CROSS JOIN

Understanding Joins

1. INNER JOIN

Returns only those records that have matching values in both tables.

```
SELECT E.Name, D.DepartmentName
FROM Employees E
INNER JOIN Departments D ON E.DeptID = D.DeptID;
```

🟢 **Use Case**: Show employees with assigned departments.

2. LEFT OUTER JOIN

Returns all records from the **left table**, and the matched records from the right table. Unmatched right-side values are shown as NULL.

```
SELECT E.Name, D.DepartmentName
FROM Employees E
LEFT OUTER JOIN Departments D ON E.DeptID = D.DeptID;
```

🟢 **Use Case**: List all employees, even if they're not assigned to a department.

3. RIGHT OUTER JOIN

Returns all records from the **right table**, and the matched records from the left table.

```
SELECT E.Name, D.DepartmentName
FROM Employees E
RIGHT OUTER JOIN Departments D ON E.DeptID = D.DeptID;
```

🟢 **Use Case**: Show all departments, even if they have no employees.

4. FULL OUTER JOIN

Returns all records when there is a match in one of the tables.

```
SELECT E.Name, D.DepartmentName
FROM Employees E
FULL OUTER JOIN Departments D ON E.DeptID = D.DeptID;
```

🟢 **Use Case**: Combine unmatched employees and departments.

5. SELF JOIN

A table is joined with itself.

```
SELECT E1.Name AS Employee, E2.Name AS Manager
FROM Employees E1
JOIN Employees E2 ON E1.ManagerID = E2.EmpID;
```

🟢 **Use Case**: Show employee-manager relationships in a single table.

6. CROSS JOIN

Generates a Cartesian product (every row of the first table joined to every row of the second table).

```
SELECT E.Name, P.ProjectName
FROM Employees E
CROSS JOIN Projects P;
```

🟢 **Use Case**: Generate all possible employee-project combinations.

⚠️ **Pitfall**: If Table A has 100 rows and Table B has 100 rows, the result will have 10,000 rows.

Introduction to Subqueries (Nested Queries)

A **Subquery** is a query nested inside another SQL query.

```
SELECT Name
FROM Employees
WHERE DeptID = (SELECT DeptID FROM Departments WHERE DepartmentName = 'IT');
```

Types of Subqueries

1. Scalar Subquery

Returns a single value.

```
SELECT Name
FROM Employees
WHERE Salary > (SELECT AVG(Salary) FROM Employees);
```

2. Multi-Row Subquery

Returns multiple rows.

```
SELECT Name
FROM Employees
WHERE DeptID IN (SELECT DeptID FROM Departments WHERE Location = 'New York');
```

3. Correlated Subquery

Depends on the outer query. Executed once for each row of the outer query.

```
SELECT E1.Name
FROM Employees E1
WHERE Salary > (
    SELECT AVG(Salary)
    FROM Employees E2
    WHERE E1.DeptID = E2.DeptID
);
```

🟢 **Use Case**: Find employees earning more than the average in their department.

EXISTS Operator

Tests for the existence of rows in a subquery.

```
SELECT Name
FROM Employees E
WHERE EXISTS (
    SELECT 1
    FROM Projects P
    WHERE P.EmpID = E.EmpID
);
```

🟢 **Use Case**: List employees who are assigned to at least one project.

Real-Time Case Studies

Case 1: Employee Salary Report

Problem: Generate a report of employees who earn more than the average salary of their department.

Case 2: Unassigned Departments

Problem: List all departments, even those with no employees assigned.

Case 3: Project Allocation

Problem: Show all possible combinations of employees and projects to evaluate allocation possibilities.

Case 4: Manager Hierarchy

Problem: Display employees and their managers.

upGrad ENTERPRISE

Q & A

Narasimha Rao T
tnrao.trainer@gmail.com