Chapter-4 DQL - SQL Joins with MS SQL Server

In MS SQL Server, **SQL Joins** are used to retrieve data from multiple tables based on a related column between them. A join allows you to combine rows from two or more tables by using a related column. SQL joins are essential for working with normalized relational databases, where data is spread across multiple tables to avoid redundancy.

There are several types of SQL joins in MS SQL Server, each serving different purposes based on the relationship between the tables.

---

1. INNER JOIN

An **INNER JOIN** returns only the rows that have matching values in both tables. If no match is found, those rows are excluded from the result set.

Syntax:

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column = table2.column;
```

Example:

```
SELECT Employees.EmployeeName, Departments.DepartmentName
FROM Employees
INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;
```

This query will return only employees who are assigned to a department. Employees without a department or departments without employees will be excluded.

---

2. LEFT JOIN (or LEFT OUTER JOIN)

A **LEFT JOIN** returns all rows from the left table (table1), and the matched rows from the right table (table2). If there is no match, NULL values will be returned for columns from the right table.

Syntax:

```
SELECT columns
FROM table1
LEFT JOIN table2
ON table1.column = table2.column;
```

Example:

```
SELECT Employees.EmployeeName, Departments.DepartmentName
FROM Employees
LEFT JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;
```

This query will return all employees, even those who are not assigned to any department. For employees without a department, the DepartmentName will be NULL.

---

3. RIGHT JOIN (or RIGHT OUTER JOIN)

A **RIGHT JOIN** is the opposite of the LEFT JOIN. It returns all rows from the right table (table2), and the matched rows from the left table (table1). If no match is found, NULL values are returned for columns from the left table.

Syntax:

```
SELECT columns
FROM table1
RIGHT JOIN table2
ON table1.column = table2.column;
```

Example:

```
SELECT Employees.EmployeeName, Departments.DepartmentName
FROM Employees
RIGHT JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;
```

This query will return all departments, including those without any employees. For departments without employees, the EmployeeName will be NULL.

---

4. FULL JOIN (or FULL OUTER JOIN)

A **FULL JOIN** returns all rows when there is a match in either the left or right table. If there is no match, the result will include NULL values for columns from the table that lacks a matching row.

Syntax:

```
SELECT columns
FROM table1
FULL JOIN table2
ON table1.column = table2.column;
```

Example:

```
SELECT Employees.EmployeeName, Departments.DepartmentName
FROM Employees
FULL JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;
```

This query will return all employees and all departments. Employees without departments and departments without employees will show NULL in the respective columns.

---

5. CROSS JOIN

A **CROSS JOIN** returns the Cartesian product of two tables. It returns all possible combinations of rows from the left and right tables. This join does not require a condition to join the tables, and it can result in a large number of rows.

Syntax:

```
SELECT columns
FROM table1
CROSS JOIN table2;
```

Example:

```
SELECT Employees.EmployeeName, Departments.DepartmentName
FROM Employees
CROSS JOIN Departments;
```

This query will return every combination of employees and departments, regardless of any matching columns. For example, if there are 10 employees and 5 departments, the result will contain 50 rows (10 x 5).

---

6. SELF JOIN

A **SELF JOIN** is a join where a table is joined with itself. This is useful when you need to compare rows within the same table. You can use aliases to differentiate the two instances of the same table.

Syntax:

```
SELECT A.column1, B.column2
FROM table A, table B
WHERE A.column = B.column;
```

Example:

```
SELECT E1.EmployeeName AS Manager, E2.EmployeeName AS Employee
FROM Employees E1
INNER JOIN Employees E2 ON E1.EmployeeID = E2.ManagerID;
```

In this example, the Employees table is joined with itself to show a list of managers and the employees they manage. Each row in Employees represents either a manager (aliased as E1) or an employee (aliased as E2).

---

7. JOIN with Multiple Tables

You can join more than two tables in a single query by using multiple JOIN clauses. The tables are joined in sequence based on their relationships.

Syntax:

```
SELECT columns
FROM table1
INNER JOIN table2 ON table1.column = table2.column
INNER JOIN table3 ON table2.column = table3.column;
```

Example:

```
SELECT Employees.EmployeeName, Departments.DepartmentName, Locations.LocationName
FROM Employees
INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID
INNER JOIN Locations ON Departments.LocationID = Locations.LocationID;
```

This query joins three tables (Employees, Departments, and Locations) to retrieve employees, their departments, and the locations of those departments.

---

8. Using Aliases with Joins

Table aliases are often used in joins to simplify the query and improve readability, especially when dealing with multiple joins or self-joins. Aliases allow you to refer to tables more succinctly.

Example:

```
SELECT E.EmployeeName, D.DepartmentName
FROM Employees E
INNER JOIN Departments D ON E.DepartmentID = D.DepartmentID;
```

Here, `E` is an alias for the `Employees` table, and `D` is an alias for the `Departments` table.

Summary of Join Types:

Join Type	Description
INNER JOIN	Returns only the rows with matching values in both tables.
LEFT JOIN	Returns all rows from the left table and matched rows from the right table.
RIGHT JOIN	Returns all rows from the right table and matched rows from the left table.
FULL JOIN	Returns rows when there is a match in either the left or right table.
CROSS JOIN	Returns the Cartesian product of both tables.
SELF JOIN	Joins a table with itself to compare rows within the same table.