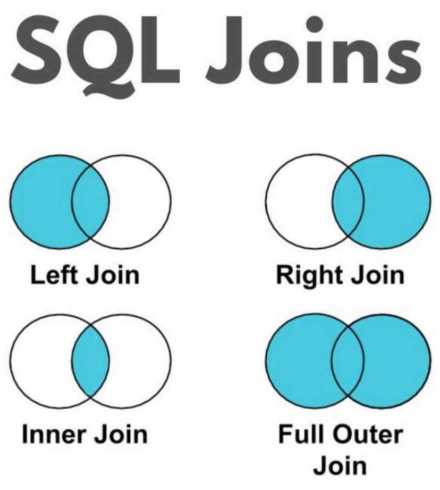

Understanding JOINS in Databases: A Detailed Guide with PostgreSQL Examples

Kāshān Asim
Software Developer: SpringBoot || Angular || ApacheSpark || Scala || PKI || eSignature
6 articles + Follow

In relational databases, JOINS are used to combine rows from two or more tables based on a related column. They are fundamental to querying data that is spread across multiple tables in a relational database. This article provides an in-depth look at different types of JOINS with scenario-based examples using PostgreSQL.

Types of JOINS

1. **INNER JOIN**
2. **LEFT JOIN (or LEFT OUTER JOIN)**
3. **RIGHT JOIN (or RIGHT OUTER JOIN)**
4. **FULL JOIN (or FULL OUTER JOIN)**
5. **CROSS JOIN**
6. **SELF JOIN**

Scenario Setup

Imagine we have two tables, employees and departments.

```
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    department_id INT,
    salary DECIMAL(10, 2)
);

CREATE TABLE departments (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100)
);
```

We will populate these tables with some sample data:

```
INSERT INTO employees (name, department_id, salary)
VALUES
('Alice', 1, 50000),
('Bob', 2, 60000),
('Charlie', NULL, 55000),
('David', 3, 70000),
('Eve', 1, 80000);

INSERT INTO departments (name) VALUES
('HR'),
    ('Engineering'),
    ('Marketing');
```

1. INNER JOIN

INNER JOIN returns rows when there is a match in both tables.

Example

We want to list all employees along with their department names.

```
SELECT e.name AS employee_name, d.name AS
department_name
FROM employees e
INNER JOIN departments d ON e.department_id = d.id;
```

Result

```
employee_name | department_name
-------------|----------------
Alice           | HR
Bob             | Engineering
David           | Marketing
Eve             | HR
```

2. LEFT JOIN (LEFT OUTER JOIN)

LEFT JOIN returns all rows from the left table, and the matched rows from the right table. If no match is found, NULLs are returned for columns of the right table.

Example

We want to list all employees along with their department names, including those who do not belong to any department.

```
SELECT e.name AS employee_name, d.name AS
department_name
FROM employees e
LEFT JOIN departments d ON e.department_id = d.id;
```

Result

```
employee_name | department_name
       --------------|----------------
Alice                    | HR
Bob                     | Engineering
Charlie                | NULL
David                  | Marketing
Eve                      | HR
```

3. RIGHT JOIN (RIGHT OUTER JOIN)

RIGHT JOIN returns all rows from the right table, and the matched rows from the left table. If no match is found, NULLs are returned for columns of the left table.

Example

We want to list all departments along with their employees, including departments that do not have any employees.

```
SELECT e.name AS employee_name, d.name AS
department_name
FROM employees e
RIGHT JOIN departments d ON e.department_id = d.id;
```

Result

```
employee_name | department_name
       --------------|----------------
Alice                    | HR
Bob                      | Engineering
David                   | Marketing
Eve                      | HR
NULL                    | Marketing
```

4. FULL JOIN (FULL OUTER JOIN)

FULL JOIN returns rows when there is a match in one of the tables. It returns NULLs for columns of the table that does not have a match.

Example

We want to list all employees and their department names, including employees without departments and departments without employees.

```
SELECT e.name AS employee_name, d.name AS
```

```
FULL JOIN departments d ON e.department_id = d.id;
```

Result

```
employee_name | department_name
 ------------------|----------------
Alice                    | HR
Bob                     | Engineering
Charlie                | NULL
David                  | Marketing
Eve                      | HR
NULL                   | Marketing
```

5. CROSS JOIN

CROSS JOIN returns the Cartesian product of the two tables, i.e., it returns all possible combinations of rows from the two tables.

Example

We want to list all possible pairs of employees and departments.

```
SELECT e.name AS employee_name, d.name AS
department_name
FROM employees e
CROSS JOIN departments d;
```

Result

```
employee_name | department_name
 ------------------|----------------
```

```
Alice                    | HR
Alice                    | Engineering
Alice                    | Marketing
Bob                      | HR
Bob                      | Engineering
Bob                      | Marketing
Charlie                  | HR
Charlie                  | Engineering
Charlie                  | Marketing
David                    | HR
David                    | Engineering
David                    | Marketing
```

Like Comment Share

6. SELF JOIN

SELF JOIN is used to join a table with itself. This is useful for hierarchical data or finding relations within the same table.

Example

We want to find all pairs of employees working in the same department.

```
SELECT e1.name AS employee1, e2.name AS employee2,
d.name AS department_nam
FROM employees e1
INNER JOIN employees e2 ON e1.department_id =
e2.department_id
INNER JOIN departments d ON e1.department_id = d.id
WHERE e1.id <> e2.id;
```

Result

```
employee1 | employee2 | department_name
-------------|-------------|----------------
Alice          | Eve             | HR
Eve            | Alice           | HR
```

Conclusion

Understanding JOINS is fundamental for querying relational databases efficiently. Each type of JOIN serves a specific purpose and understanding when to use each can greatly enhance your ability to manipulate and retrieve data from a relational database. This guide with PostgreSQL examples provides a comprehensive look at each type of JOIN, helping you to apply them effectively in your database operations.

Published by

Kāshān Asim

Software Developer: SpringBoot || Angular || ApacheSpark || Scala || PKI || eSignature

6 articles

+ Follow

Like Comment Share

0 Comments

Add a comment...

Kāshān Asim

Software Developer: SpringBoot || Angular || ApacheSpark || Scala || PKI || eSignature

+ Follow