#PostgreSQL#

Section 3. Joining Multiple Tables
Joins – show you a brief overview of joins in PostgreSQL.
Table aliases – describes how to use table aliases in the query.
Inner Join– select rows from one table that have the corresponding rows in other tables.
Left Join – select rows from one table that may or may not have the corresponding rows in other tables.
Self-join – join a table to itself by comparing a table to itself.
Full Outer Join – use the full join to find a row in a table that does not have a matching row in another table.
Cross Join – produce a Cartesian product of the rows in two or more tables.
Natural Join – join two or more tables using implicit join conditions based on the common column names in the joined tables.

Can You Teach Me above all the topics with Realtime example (create a simple table and insert some values) and show output in Table Format and Also Explain Each topic why this needed, what happen, like syntax and so on.

PostgreSQL Joins: A Brief Overview with Real-World Examples

Let's explore each join type with a simple table and some example data. I'll show how they work, why they're useful, and provide the output in table format.

1. Inner Join

Description: Selects rows from both tables that have matching values in both tables. **Syntax**:

```sql


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

Example:

Tables:

- `employees`: Employee ID, Name, Department
- `departments`: Department ID, Department Name

Data:

- `employees`:

EmployeeID	Name	DepartmentID
1	John	101
2	Jane	102
3	Mike	103

- `departments`:

DepartmentID	DepartmentName
101	HR
102	IT
103	Sales

Query:

```sql
SELECT employees.Name, departments.DepartmentName
FROM employees
INNER JOIN departments ON employees.DepartmentID = departments.DepartmentID;
```

Output:

Name	DepartmentName
John	HR
Jane	IT
Mike	Sales

2. Left Join

Description: Selects all rows from the left table and the matched rows from the right table. If there is no match, it returns NULL for the right side. **Syntax**:

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

Example:

Tables:

- `employees`: Employee ID, Name, Department
- `departments`: Department ID, Department Name

Data:

- `employees`:

EmployeeID	Name	DepartmentID
1	John	101
2	Jane	102
3	Mike	103
4	Alice	NULL

- `departments`:

DepartmentID	DepartmentName
101	HR
102	IT
103	Sales

Query:

```sql
SELECT employees.Name, departments.DepartmentName
FROM employees
LEFT JOIN departments ON employees.DepartmentID = departments.DepartmentID;
```

Output:

Name	DepartmentName
John	HR
Jane	IT
Mike	Sales
Alice	NULL

3. Self-Join

Description: A self-join is used to join a table with itself, by comparing the table to itself. **Syntax**:

```sql
SELECT columns
FROM table AS alias1
INNER JOIN table AS alias2 ON alias1.common_column = alias2.common_column;
```

Example:

Tables:

- `employees`: EmployeeID, Name, ManagerID
- This table contains information about employees and their respective managers.

Data:

- `employees`:

EmployeeID	Name	ManagerID
1	John	NULL
2	Jane	1
3	Mike	1
4	Alice	2
5	Bob	3

Query:

```sql
SELECT e1.Name AS Employee, e2.Name AS Manager
FROM employees e1
INNER JOIN employees e2 ON e1.ManagerID = e2.EmployeeID;
```

Output:

Employee	Manager
John	NULL
Jane	John
Mike	John
Alice	Jane
Bob	Mike

4. Full Outer Join

Description: Returns all rows when there is a match in either left or right table. If no match, NULL values are returned. **Syntax**:

```sql
SELECT columns
FROM table1
FULL OUTER JOIN table2 ON table1.common_column = table2.common_column;
```

Example:

Tables:

- `**employees**`: EmployeeID, Name
- `**departments**`: DepartmentID, DepartmentName

Data:

- `**employees**`:

EmployeeID	Name
1	John

EmployeeID	Name
2	Jane

- `departments`:

DepartmentID	DepartmentName
101	HR
103	Sales

Query:

```sql
SELECT employees.Name, departments.DepartmentName
FROM employees
FULL OUTER JOIN departments ON employees.DepartmentID = departments.DepartmentID;
```

Output:

Name	DepartmentName
John	NULL
Jane	NULL
NULL	HR
NULL	Sales

5. Cross Join

Description: Produces a Cartesian product of the rows in two or more tables. **Syntax**:

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

Example:

Tables:

- `colors`: ID, Color
- `shapes`: ID, Shape

Data:

- `colors`:

ID	Color
1	Red
2	Blue

- `shapes`:

ID	Shape
1	Circle
2	Square

Query:

```sql
SELECT colors.Color, shapes.Shape
FROM colors
CROSS JOIN shapes;
```

Output:

Color	Shape
Red	Circle
Red	Square
Blue	Circle
Blue	Square

6. Natural Join

Description: Joins two or more tables using the common column names without explicitly specifying them in the `ON` clause. **Syntax**:

```sql
SELECT columns
FROM table1
NATURAL JOIN table2;
```

Example:

Tables:

- `employees`: EmployeeID, Name, DepartmentID
- `departments`: DepartmentID, DepartmentName

Data:

- `employees`:

EmployeeID	Name	DepartmentID
1	John	101
2	Jane	102
3	Mike	103

- `departments`:

DepartmentID	DepartmentName
101	HR
102	IT
103	Sales

Query:

```sql
SELECT employees.Name, departments.DepartmentName
FROM employees
NATURAL JOIN departments;
```

Output:

Name	DepartmentName
John	HR
Jane	IT
Mike	Sales

Each join type serves a specific purpose and helps in combining data from different tables based on their relationships. They allow us to analyze data from multiple perspectives, making it easier to understand relationships between entities.