GABRIELE ROMANATO

JOIN in PostgreSQL: dalle basi alle query avanzate

In PostgreSQL, le JOIN permettono di combinare righe da due o più tabelle basandosi su una condizione correlata. Sono fondamentali per lavorare con dati normalizzati. Vediamo i principali tipi di JOIN partendo dalle più semplici fino alle più avanzate.

1. INNER JOIN

L'**INNER JOIN** restituisce solo le righe che hanno corrispondenze in entrambe le tabelle.

```
SELECT *
FROM ordini
INNER JOIN clienti ON ordini.cliente_id = clienti.id;
```

2. LEFT JOIN (o LEFT OUTER JOIN)

La **LEFT JOIN** restituisce tutte le righe della tabella di sinistra e le righe corrispondenti della tabella di destra. Se non ci sono corrispondenze, restituisce NULL per i campi della tabella di destra.

```
SELECT *
FROM clienti
LEFT JOIN ordini ON clienti.id = ordini.cliente_id;
```

3. RIGHT JOIN

Funziona come la LEFT JOIN, ma include tutte le righe della tabella di destra.

```
SELECT *  
FROM ordini  
RIGHT JOIN clienti ON ordini.cliente_id = clienti.id;
```

4. FULL JOIN

La **FULL JOIN** restituisce tutte le righe quando c'è una corrispondenza in una delle due tabelle.

```
SELECT *  
FROM clienti  
FULL JOIN ordini ON clienti.id = ordini.cliente_id;
```

5. CROSS JOIN

La **CROSS JOIN** restituisce il prodotto cartesiano delle due tabelle, ovvero tutte le combinazioni possibili di righe.

```
SELECT *  
FROM clienti  
CROSS JOIN categorie;
```

6. JOIN con clausola USING

Quando le colonne da confrontare hanno lo stesso nome in entrambe le tabelle, si può usare **USING**.

```
SELECT *
FROM clienti
JOIN ordini USING (cliente_id);
```

7. JOIN con alias e selezione specifica

Si possono usare alias per rendere le query più leggibili e selezionare solo i campi desiderati.

```
SELECT c.nome, o.data_ordine
FROM clienti c
JOIN ordini o ON c.id = o.cliente_id;
```

8. JOIN tra più tabelle

Le JOIN possono essere concatenate per combinare più di due tabelle.

```
SELECT c.nome, o.data_ordine, p.nome AS prodotto
FROM clienti c
JOIN ordini o ON c.id = o.cliente_id
JOIN prodotti p ON o.prodotto_id = p.id;
```

9. JOIN con subquery

Le JOIN possono essere utilizzate in combinazione con subquery per filtrare o aggregare dati.

```
SELECT c.nome, o.totale
FROM clienti c
JOIN (
  SELECT cliente_id, SUM(importo) AS totale
  FROM ordini
  GROUP BY cliente_id
) o ON c.id = o.cliente_id;
```

10. JOIN LATERALE (LATERAL JOIN)

Il **LATERAL JOIN** permette a una subquery di fare riferimento a righe della tabella principale.

```
SELECT c.nome, o.*
FROM clienti c
JOIN LATERAL (
  SELECT *
  FROM ordini
  WHERE ordini.cliente_id = c.id
  ORDER BY data_ordine DESC
  LIMIT 1
) o ON true;
```

Conoscere a fondo le diverse tipologie di JOIN è fondamentale per scrivere query potenti e performanti in PostgreSQL.