## GABRIELE ROMANATO

### Query JOIN in MySQL: dalle basi all'avanzato

Le query JOIN in MySQL permettono di combinare righe provenienti da due o più tabelle in base a una condizione correlata. Sono uno strumento fondamentale per ottenere dati relazionati in modo efficiente. Vediamo le principali tipologie, da quelle più semplici a quelle più complesse.

# 1. INNER JOIN

Restituisce solo le righe che hanno corrispondenze in entrambe le tabelle.

```
SELECT utenti.nome, ordini.data
FROM utenti
INNER JOIN ordini ON utenti.id = ordini.utente_id;
```

# 2. LEFT JOIN (o LEFT OUTER JOIN)

Restituisce tutte le righe dalla tabella a sinistra, e le corrispondenze dalla tabella a destra. Se non ci sono corrispondenze, i valori della tabella destra saranno NULL.

```
SELECT utenti.nome, ordini.data
FROM utenti
LEFT JOIN ordini ON utenti.id = ordini.utente_id;
```

# 3. RIGHT JOIN (o RIGHT OUTER JOIN)

Funziona come il LEFT JOIN, ma restituisce tutte le righe dalla tabella a destra.

```
SELECT utenti.nome, ordini.data
FROM utenti
RIGHT JOIN ordini ON utenti.id = ordini.utente_id;
```

## 4. FULL OUTER JOIN (simulato)

MySQL non supporta direttamente il FULL OUTER JOIN, ma può essere simulato con una combinazione di LEFT e RIGHT JOIN con UNION.

```
SELECT utenti.nome, ordini.data
FROM utenti
LEFT JOIN ordini ON utenti.id = ordini.utente_id
UNION
SELECT utenti.nome, ordini.data
FROM utenti
RIGHT JOIN ordini ON utenti.id = ordini.utente_id;
```

## 5. JOIN su più tabelle

È possibile concatenare più JOIN per unire più tabelle.

```
SELECT utenti.nome, ordini.data, prodotti.nome AS prodotto
FROM utenti
INNER JOIN ordini ON utenti.id = ordini.utente_id
INNER JOIN prodotti ON ordini.prodotto_id = prodotti.id;
```

## 6. SELF JOIN

Un SELF JOIN consente di unire una tabella con se stessa. Utile ad esempio per rappresentare relazioni gerarchiche.

```
SELECT A.nome AS dipendente, B.nome AS manager
FROM impiegati A
```

```
JOIN impiegati B ON A.manager_id = B.id;
```

## 7. CROSS JOIN

Restituisce il prodotto cartesiano delle due tabelle, cioè tutte le combinazioni possibili.

```
SELECT *
FROM colori
CROSS JOIN taglie;
```

## Conclusione

Comprendere le differenze tra i vari tipi di JOIN è essenziale per scrivere query SQL efficienti e corrette. Man mano che i dati crescono in complessità, padroneggiare le JOIN permette di ottenere insight precisi da database relazionali.