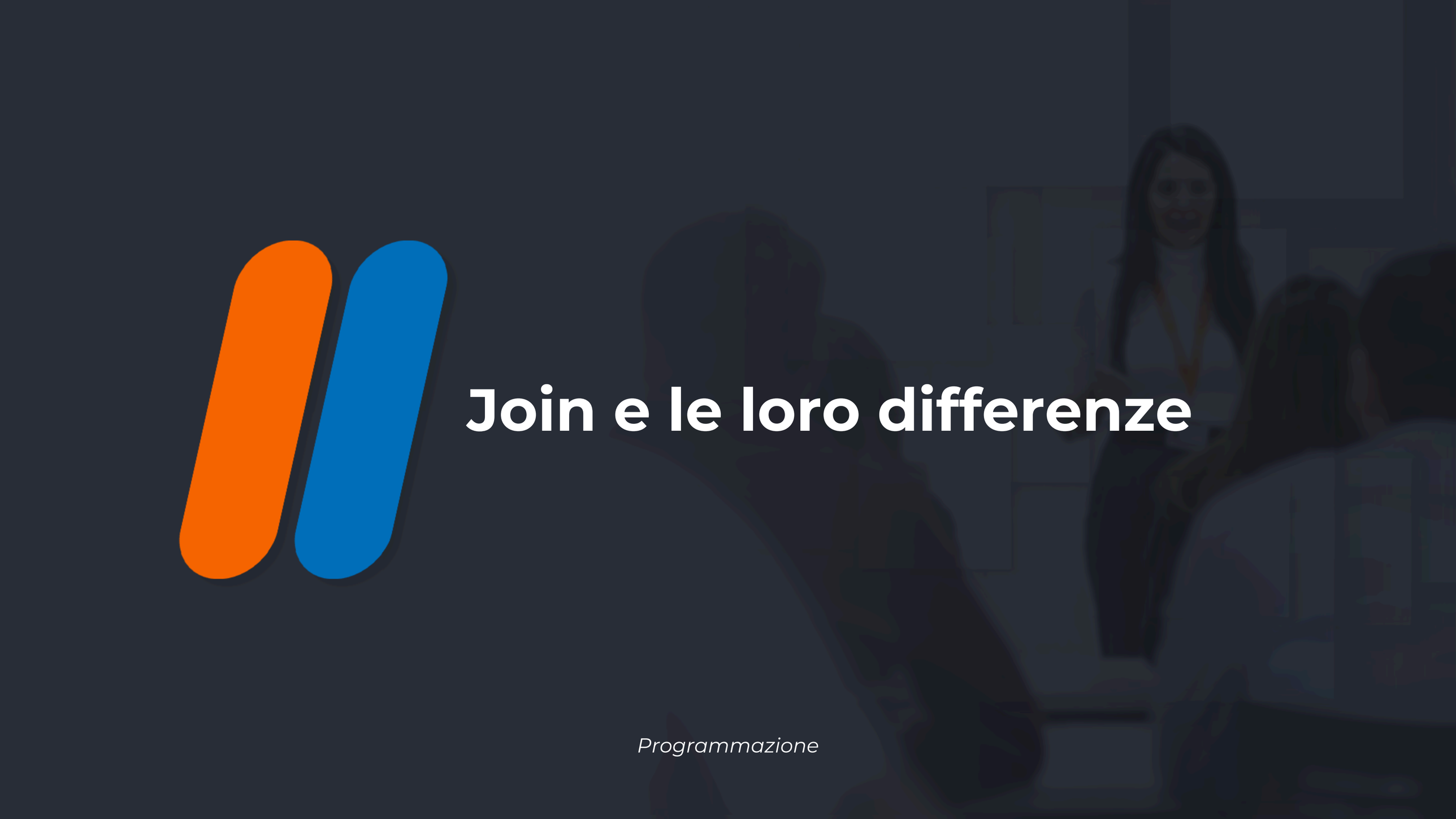


# Join e le loro differenze

*Programmazione*

Nel contesto dei database relazionali, le JOIN sono strumenti fondamentali per combinare dati provenienti da più tabelle in base a una relazione logica tra di esse.

Normalmente questa relazione viene stabilita attraverso una chiave primaria in una tabella e una chiave esterna in un’altra.

L’uso delle JOIN permette di superare i limiti della singola tabella, ricostruendo informazioni complesse a partire da dati normalizzati. Ad esempio, una tabella Clienti e una tabella Ordini possono essere unite per mostrare non solo i dettagli del cliente ma anche tutti gli ordini associati.

Esistono diverse tipologie di JOIN, ognuna con uno scopo specifico e con regole precise su come i dati vengono estratti.

Le più comuni sono: INNER JOIN, LEFT JOIN, RIGHT JOIN e FULL OUTER JOIN.

L’INNER JOIN restituisce solo i record che hanno corrispondenze nelle due tabelle.

La LEFT JOIN mostra tutti i record della tabella di sinistra anche se non trovano corrispondenza in quella di destra, inserendo valori nulli nei campi mancanti.

La RIGHT JOIN privilegia la tabella di destra, mentre la FULL OUTER JOIN restituisce tutti i record di entrambe le tabelle, indipendentemente dal fatto che abbiano o meno corrispondenze.

Le differenze tra queste tipologie incidono direttamente sul risultato finale e quindi sulla logica delle query.

La scelta del tipo di JOIN dipende dall'informazione che si vuole ottenere: se serve solo l'intersezione dei dati è più adatta una INNER JOIN; se invece si vuole conservare tutte le righe di una tabella indipendentemente dalle corrispondenze, si preferisce una LEFT o RIGHT JOIN; quando l'obiettivo è avere una visione completa dei dati di entrambe le tabelle, anche senza relazioni esistenti, si utilizza una FULL OUTER JOIN.

Comprendere queste differenze è cruciale per costruire query efficienti e interpretare correttamente i risultati, evitando dati mancanti o duplicazioni indesiderate.

| ***TIPO DI JOIN*** | ***COSA RESTITUISCE*** | ***ESEMPIO TIPICO*** |
|---|---|---|
| **INNER JOIN** | **Solo le righe che hanno corrispondenza in entrambe le tabelle** | **Clienti che hanno fatto ordini** |
| **LEFT JOIN** | **Tutte le righe della tabella di sinistra + corrispondenze della destra (NULL se assenti)** | **Tutti i clienti, anche quelli senza ordini** |
| **RIGHT JOIN** | **Tutte le righe della tabella di destra + corrispondenze della sinistra (NULL se assenti)** | **Tutti gli ordini, anche se non collegati a un cliente registrato** |
| **FULL OUTER JOIN** | **Tutte le righe di entrambe le tabelle, con valori NULL dove non ci sono corrispondenze** | **Tutti i clienti e tutti gli ordini, anche se non legati tra loro** |

## Esempi pratici

## Supponiamo di avere due tabelle:

| ClienteID | Nome |
|---|---|
| 1 | Anna |
| 2 | Marco |
| 3 | Luca |

| OrdineID | ClienteID | Prodotto |
|---|---|---|
| 101 | 1 | Laptop |
| 102 | 1 | Mouse |
| 103 | 4 | Tastiera |

# INNER JOIN

```
1. SELECT Clienti.Nome, Ordini.Prodotto
2. FROM Clienti
3. INNER JOIN Ordini ON Clienti.ClienteID = Ordini.ClienteID;
```

**vengono presi solo i clienti che hanno ordini registrati.**
**Marco e Luca non compaiono perché non hanno ordini, mentre l'ordine con ClienteID=4 non compare perché non esiste nessun cliente con quell'ID.**

**Risultato:**

| Nome | Prodotto |
|---|---|
| Anna | Laptop |
| Anna | Mouse |

# LEFT JOIN

```
1. SELECT Clienti.Nome, Ordini.Prodotto
2. FROM Clienti
3. LEFT JOIN Ordini ON Clienti.ClienteID = Ordini.ClienteID;
```

**tutti i clienti vengono visualizzati. Marco e Luca non hanno ordini, quindi nella colonna Prodotto compaiono valori NULL.**

**Risultato:**

| Nome | Prodotto |
|---|---|
| Anna | Laptop |
| Anna | Mouse |
| Marco | NULL |
| Luca | NULL |

## RIGHT JOIN

```
1. SELECT Clienti.Nome, Ordini.Prodotto
2. FROM Clienti
3. RIGHT JOIN Ordini ON Clienti.ClienteID = Ordini.ClienteID;
```

**tutti gli ordini vengono mostrati, anche quelli senza cliente corrispondente. L'ordine con ClienteID=4 viene visualizzato con il nome cliente NULL.**

**Risultato:**

| Nome | Prodotto |
|---|---|
| Anna | Laptop |
| Anna | Mouse |
| NULL | Tastiera |

## FULL OUTER JOIN

```
1. SELECT Clienti.Nome, Ordini.Prodotto
2. FROM Clienti
3. FULL OUTER JOIN Ordini ON Clienti.ClienteID = Ordini.ClienteID;
```

**vengono inclusi tutti i clienti e tutti gli ordini. Se non c’è corrispondenza, compaiono valori NULL.**

**Risultato:**

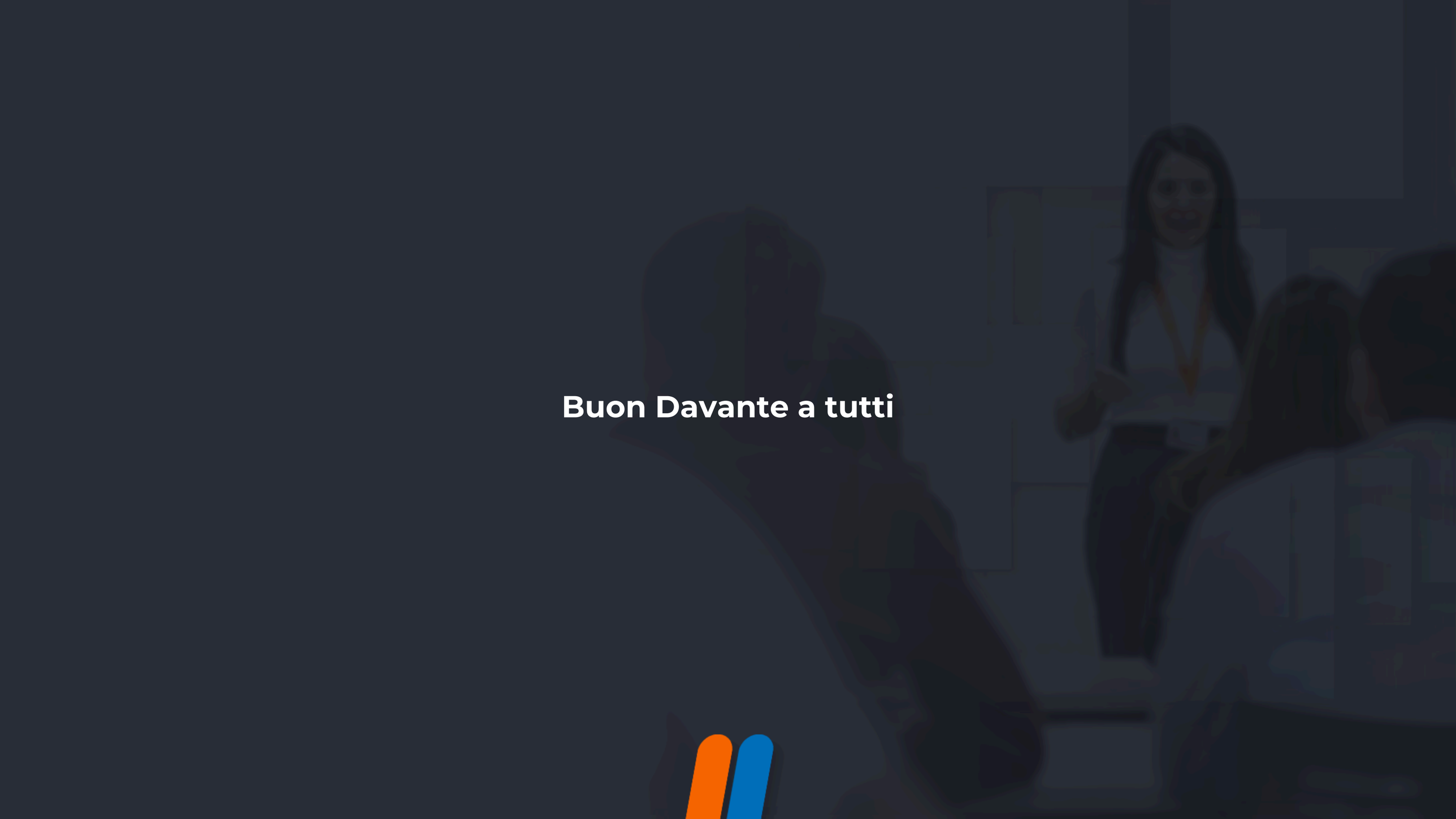
| Nome | Prodotto |
|---|---|
| Anna | Laptop |
| Anna | Mouse |
| Marco | NULL |
| Luca | NULL |
| NULL | Tastiera |

Un errore frequente è l’uso improprio delle condizioni di join: dimenticare di specificare correttamente la clausola ON o confonderla con la WHERE può portare a risultati incompleti o, peggio, a cartesian product (tutte le combinazioni possibili tra le righe delle tabelle, con un numero di record enorme e privo di senso).

Un altro problema diffuso riguarda la presenza di valori NULL, che nelle join (soprattutto LEFT, RIGHT e FULL OUTER) possono generare righe inattese o apparire come dati mancanti difficili da gestire se non previsti.

Inoltre, quando le tabelle hanno relazioni uno-a-molti o molti-a-molti, il risultato può contenere duplicati non desiderati: in questi casi diventa fondamentale utilizzare clausole come DISTINCT o funzioni di aggregazione (COUNT, SUM, ecc.) per evitare ridondanze.

Infine, query con più join su tabelle grandi possono diventare molto lente se non si definiscono correttamente indici sulle colonne usate per le relazioni, portando a seri problemi di performance.


Buon Davante a tutti