

Query in SQL (interrogazioni)

SELECT * ← Proietta tutte le colonne
FROM tabella

SELECT 4 * 12, 1 + 1

↑
Proietta il risultato delle operazioni

SELECT nome, cognome
FROM

↑
Proiezione solo sulle colonne indicate

SELECT nome, cognome
FROM impiegati

WHERE livello = 4 ← Restrizione sulle righe che limita il risultato solo a quelle che soddisfano la condizione

| name | cognome | livello |
|------|---------|---------|
| ~ | ~ | 3 |
| ~ | ~ | 4 |
| ~ | ~ | 4 |

cardinalità = numero righe

grado = numero colonne

La join tra tabelle

La join è un'operazione tra due tabelle, legate da una relazione 1:N, in cui viene fatto il prodotto cartesiano e vengono mantenute solo le righe in cui c'è corrispondenza tra PK e relativa FK.

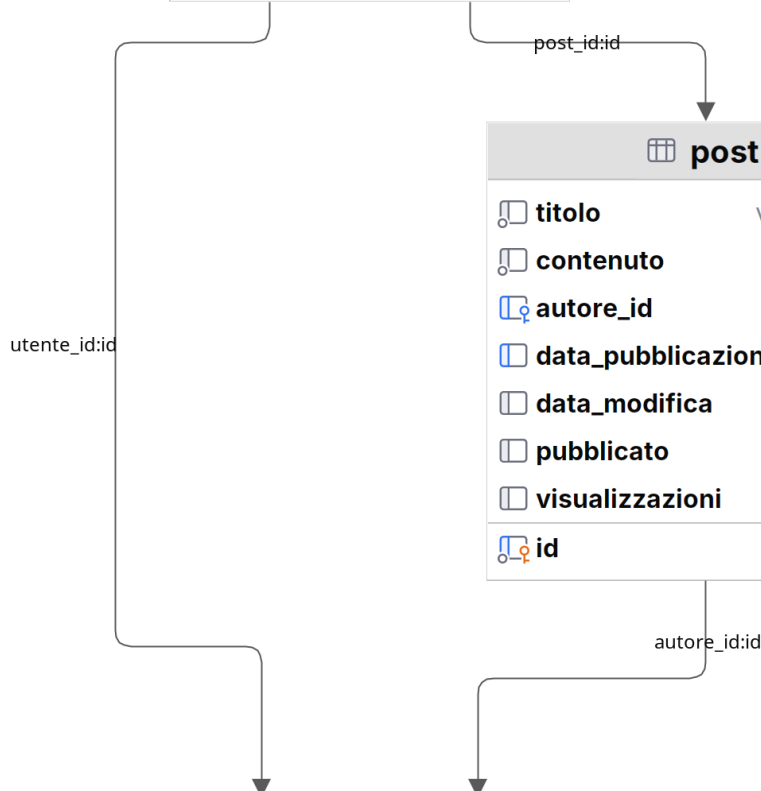
Il prodotto cartesiano è l'insieme di tutti gli elementi di una tabella messi in coppia con tutti gli elementi dell'altra

$$\begin{aligned}
 A &= [1, 2] \\
 B &= [3, 4] \\
 \Rightarrow A \times B &= [(1, 3), (1, 4), \\
 &\quad (2, 3), (2, 4)]
 \end{aligned}$$

| commenti | |
|---------------|------------|
| post_id | int(11) |
| utente_id | int(11) |
| contenuto | text |
| data_commento | timestamp |
| approvato | tinyint(1) |
| id | int(11) |

| post | |
|--------------------|--------------|
| titolo | varchar(200) |
| contenuto | text |
| autore_id | int(11) |
| data_pubblicazione | timestamp |
| data_modifica | timestamp |
| pubblicato | tinyint(1) |
| visualizzazioni | int(11) |
| id | int(11) |

| utenti | |
|--------------------|--------------|
| nome | varchar(100) |
| cognome | varchar(100) |
| email | varchar(150) |
| password_hash | varchar(255) |
| data_registrazione | timestamp |
| ultimo_accesso | timestamp |
| attivo | tinyint(1) |
| id | int(11) |



Join implicita utilizzando la WHERE

```
SELECT post.titolo, utenti.nome, utenti.cognome  
FROM post, utenti  
WHERE post.autore_id = utenti.id;
```

Join esplicita utilizzando JOIN...ON

```
SELECT post.titolo, utenti.nome, utenti.cognome  
FROM post JOIN utenti  
ON post.autore_id = utenti.id
```

Questo tipo join è anche detto INNER JOIN, in cui eventuali righe che non hanno corrispondenza spariscono, e i due modi di scriverlo sono equivalenti. Vedremo più avanti altri casi (RIGHT e LEFT JOIN) che possono essere utilizzati solo attraverso la seconda modalità.

Esempi di join con più di due tabelle

```
-- Elenco dei post con i relativi commenti e gli autori dei commenti  
SELECT titolo, commenti.contenuto, nome, cognome  
FROM post, commenti, utenti  
WHERE post.id = commenti.post_id  
AND commenti.utente_id = utenti.id;
```

```
-- Equivalente con JOIN esplicito  
SELECT titolo, commenti.contenuto, nome, cognome  
FROM commenti JOIN utenti  
    ON commenti.utente_id = utenti.id  
JOIN post  
    ON post.id = commenti.post_id;
```

Ordinamento ORDER BY

Il risultato di una query non garantisce nessun tipo di ordinamento, quindi, se interessa, va specificato utilizzando la clausola ORDER BY successivamente alla richiesta.

L'ordinamento può utilizzare una o più colonne. Nel caso di più colonne la priorità è da sinistra a destra.

L'ordinamento naturale è dal piccolo al grande, se si vuole l'ordine decrescente si usa la parola chiave DESC.

Generalmente è meglio mettere sempre l'ORDER BY

Limite sul numero di righe LIMIT

La parola chiave LIMIT, aggiunta in coda a una query, permette di specificare l'intervallo di righe che ci interessano (per paginazione o scrolling infinito).

LIMIT numero di righe

LIMIT inizio, numero di righe

```
SELECT *  
FROM players  
LIMIT 100, 50;
```

In questo esempio vengono restituiti 50 giocatori a partire dal 101-esimo.

Eliminare i duplicati DISTINCT

Una tabella non può contenere righe duplicate. Se però vengono fatte delle proiezioni, possono comparire dei duplicati. Per rimuovere i duplicati e tenere una sola copia di ogni elemento, si usa la parola DISTINCT

```
SELECT DISTINCT tourney_fin_currency  
FROM tournaments  
WHERE tourney_fin_currency IS NOT NULL;
```


Operatori di aggregazione

Questi operatori estraggono informazioni dal DB che non sono memorizzate esplicitamente, ma che per essere estratte hanno bisogno di effettuare dei calcoli su gruppi di elementi.

Operatore di conteggio delle righe

La parola chiave utilizzata è COUNT e "assomiglia" a una funzione, quindi vengono utilizzate le parentesi tonde ed eventuali parametri:

```
SELECT COUNT(*) AS n_mancini  
FROM players  
WHERE handedness = "Left-Handed";
```

In questo esempio vengono contati i giocatori mancini.

L'asterisco indica che vanno contate tutte le righe

Se al posto dell'asterisco si inserisse il nome di un campo, verrebbero contate solo le righe in cui quel campo è non nullo.

Per quanto riguarda tutti gli operatori di aggregazione, nella proiezione in cui vengono usati, non possono comparire altri campi non aggregati.

Operatore di Somma

Questo operatore, chiamato SUM, serve a sommare i valori di un campo passato come attributo (ovviamente l'operazione di somma deve avere senso per quel campo).

```
SELECT winner_name AS giocatore,  
       SUM(winner_games_won) AS game_vinti  
FROM match_scores  
WHERE winner_name = "Alexander Bublik";
```

Operatori per il calcolo di minime e massimo

Questi operatori ritornano il valore minimo o massimo di un certo campo passato come parametro. Si applicano a qualsiasi tipo di dato e seguono l'ordine naturale

```
SELECT MAX(height_cm) AS altezza_massima, MIN(height_cm) AS altezza_minima  
FROM players  
WHERE height_cm > 100;
```

Operatore per il calcolo della media

Fa la media dei valori del campo passato come parametro, escludendo i valori null. Si chiama AVG

```
SELECT AVG(height_cm) AS altezza_media, COUNT(*) AS tennisti  
FROM players  
WHERE height_cm > 100;
```

Query annidate (e sottoquery)