## **SELECT**

# **SELECT\_TEORIA**

```
-- seleziona colonne dalla tabella
SELECT colonna1, colonna2, ...
FROM nome_tabella
WHERE condizioni;
```

e' la sintassi del SELECT in sql.

Serve a estrapolare dati dal DB che rispettano una condizione precisata.

Le condizioni sono specificate dalla clausola WHERE.

• Esiste un modo per il SELECT d'indicare tutte le colonne della tabella usando \*

```
SELECT * FROM tabella;
```

• Esiste un modo per il SELECT d'indicare solo <u>distinti</u> elementi della tabella usando <u>DISTINCT</u>. Alternativamente può essere utilizzato per le funzioni aggregate in lista sotto.

```
SELECT DISTINCT colonna FROM tabella;
```

- Esistono funzioni per il SELECT:
  - MIN() ritorna il valore piu' piccolo delle colonne selezionate;
  - MAX() ritorna il valore piu' grande delle colonne selezionate;
  - AVG() ritorna il valore medio di una colonna numerica;
  - COUNT() ritorna il <u>numero di righe</u> che rispettano un criterio;
  - SUM() ritorna la somma dei valori in una colonna numerica.

```
SELECT [ MIN | MAX | AVG | COUNT | SUM ] colonne
FROM tabelle
WHERE condizione;
```

Una colonna o tabella, puo' essere <u>rinominata</u> con la parola chiave AS.

```
SELECT colonna AS variabile
FROM tabella;
```

#### Operatori della clausola WHERE sono:

operatore	descrizione
=	uguale
>	maggiore di
<	minore di
>=	maggiore di o uguale
<=	minore di o uguale

operatore	descrizione
!=	non uguale
BETWEEN	a BETWEEN x AND y
LIKE	cerca per un pattern
IN	specifica multipli valori possibili

string	LIKE	pattern	result
'abc'	LIKE	'abc'	true
'abc'	LIKE	'a%'	true
'abc'	LIKE	'_b_'	true
'abc'	LIKE	'c'	false

I record possono essere filtrati specificando piu' condizioni:

- AND mostra un record se tutte le condizioni separate con esso vengono soddisfatte;
- OR mostra record se una qualsiasi delle condizioni separate dallo stesso viene soddisfatta;
- NOT mostra il record se il risultato della condizione e' "non vero".

```
SELECT colonna1, colonna2, ...
FROM tabella
WHERE condizione1 [ AND | OR | NOT ] condizione2
[ AND | OR | NOT ] condizione3, ...;
```

I risultati possono essere ordinati usando ORDER BY.

Di default, gli elementi verranno ordinati in modo *ascendente* ASC, ma possiamo specificare in altro modo con DESC.

```
SELECT colonna1, colonna2
FROM tabella
WHERE clausola
ORDER BY colonna [ ASC | DESC ]
```

La SELECT di default non permette di fare unioni, <u>unire le colonne</u> per fornire una nuova tabella con le colonne scelte, serve un costrutto esplicito <u>UNION</u>. Se volessimo tutti i duplicati aggiungiamo anche la parola chiave ALL.

La <u>differenza</u> viene implementata con <u>EXCEPT</u> e vengono come prima, eliminati i duplicati almeno che <u>ALL</u> non venga aggiunto.

Anche l'intersezione è possibile con INTERSECT.

```
SELECT colonna1, colonna2

FROM tabella1

UNION [ [ALL] | EXCEPT [ALL] | INTERSECT ]

SELECT colonna3, colonna4

FROM tabella2
```

Le <u>n-uple possono essere raggruppate</u> a singoli gruppetti, usando GROUP BY e specifichiamo le nostre condizioni con HAVING.

Se avessimo da utilizzare funzioni aggregate su attributi (come SUM oppure COUNT), utilizzeremmo GROUP BY per specificare in base a quale criterio raggruppare le n-uple.

```
SELECT colonna
FROM tabella
```

```
WHERE condizione
GROUP BY criterio
HAVING condizione
```

Se lavoriamo con gruppi di NULL, questi siccome non distinti verranno raggruppati

#### Interrogazione nidificata

Il confronto tra attributo e risultato di sotto-interrogazione è possibile, l'attributo ha un solo valore. Le quantificazioni esistenziali sono il caso cardine. La forma piana e la forma nidificata possono essere combinate, c'è da dire che la forma nidificata è "meno dichiarativa" ma talvolta più leggibile.

Usare EXIST come condizione esistenziale è utile per le interrogazioni nidificate:

Sommario di SELECT: sinopsi da manuale psql

# SELECT\_ESEMPI

## example1

Immaginiamo di avere un db con al suo interno una tabella chiamata Clienti. Al suo interno ci sono 7 colonne e una di queste si chiama paese. Ora, immaginiamo di:

estrarre tutte le info dei clienti italiani

```
SELECT * FROM Clienti
WHERE paese='Italia';

ID nome contatto indirizzo citta cap paese
```

ID	nome	contatto	indirizzo	citta	сар	paese
27	franchi s.p.a.	Paolo Accorti	Via Monte 34	Torino	10100	Italia
49	magazzini alimentari riuniti	Giovanni Rovelli	Via Ludovico il Moro 22	Bergamo	24100	Italia
66	Reggiani Caseifici	Maurizio Moroni	Strada Provinciale 124	Reggio Emilia	42100	Italia

#### example2

Immaginiamo ora di voler estrarre, sempre all'interno della stessa tabella Clienti, tutti i paesi che ne fanno parte.

Per farlo ci serve un modo per fare distinzione tra duplicati (non vogliamo imbrogliare contando 2/3/4 volte il paese Italia).

estraiamo i paesi nella tabella

```
SELECT DISTINCT paese
FROM Clienti;
```

paese	/
argentina	1
belgio	1
brazile	1
canada	1
danimarca	1
finlandia	1
italia	/
	1

#### example3

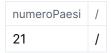
Abbiamo una lista di paesi, vorremmo ora contarla.

Gia' che ci siamo diamo un nome a quello che otteniamo.

Facciamo uso della funzione COUNT() e di AS.

contiamo i paesi nella tabella e assegnamo variabile

```
SELECT COUNT(DISTINCT paese) AS numeroPaesi
FROM Clienti;
```



## example4

estraiamo le info dei clienti in Germania ma solo dei paesi Munchen e Berlin

```
SELECT * FROM Clienti
WHERE paese = 'Germania' AND
        (citta = 'Berlin' OR citta = 'Munchen');
```

ID nome contatto indirizzo citta cap paese
--

ID	nome	contatto	indirizzo	citta	сар	paese
1	alfreds futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germania
25	Frankenversand	Peter Franken	Berliner Platz 43	Munchen	80805	Germania

# example5

il numero di figli di ciascun padre

#### paternita

Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

SELECT Padre, COUNT(\*) AS NumFigli FROM paternita GROUP BY Padre

Padre	NumFigli
Sergio	1
Luigi	2
Franco	2