

SQL

DB (ID | NOME | COGNOME)

SELECT * FROM tabella;

INSERT INTO tabella VALUES();

VALUES (ID = ' ', NOME = ' ', COGNOME = ' ')

DEFAULT applicabile mentre sto creando una tabella, ols inserire dopo il tipo ed il nome obbligatorio.

CREATE TABLE NomeTab VALUES ID INT(5) DEFAULT 12345

oppure al posto di default inserisco AUTOINCREMENT e per non

ho bisogno di mettergli i valori.

UPDATE tabella

SET NOME = 'Ermis'

WHERE nome = 'Ermis' OR nome = 'Erm'

Se non ci avessi messo WHERE tutti i record avrebbero
svolto un update di nome 'Ermis'

DELETE FROM tabella // < solo questo comando elimina tutto

WHERE colonna = valore_colonna

SELECT tutte le colonne che voglio FROM tabella;

WHERE colonna > < valore_colonna

colonna between (x, y)

ORDER BY colonna;

ORDER BY colonna1 DESC; // decrescente
ASC ascendente

colonna1, colonna2

LIMIT n; // limita i risultati a n

WHERE col1 = x OR col2 = y
AND

<, >, <=, >=, =, <>, AND, OR
(#)

WHERE col IN ('x', 'y', 'z');

Seleziona solo i valori che hanno col = x, y, z

WHERE condizione 1 **AND** condizione 2
OR

TAB1 f₁, f₂, f₃ colonne di z ≠ tabella
TAB2 D₁, D₂, D₃

Foreign Key deposita lo PK di un'altra tabella
PK, FK, Attivato.

CREATE TABLE tab ();
tab2 (nome TIPO PRIMARY KEY
FOREIGN KEY (nome) REFERENCES
tab1 (col della FK) ON DELETE SET NULL

// bisogna prima creare una tabella, alle successive tabelle
FK cesse. Si possono applicare FK riferite alla tab1, ma
per creare le FK nella prima tabella si deve fare:
ALTER TABLE tab1

ADD FOREIGN KEY col tab1
REFERENCE tab2 (col tab2 Fk)
ON DELETE SET NULL;

Si può creare una PK dalla composizione di 2 PK o 2 attributi.
ON DELETE CASCADE da usare in questo caso.

"COMPOSITE KEYS"

SQL QUERIES

① `SELECT * FROM tabella WHERE colonna_1 = (confronto)`

② `WHERE col = (select max(col) from tabella)`

// ritorna il record con il valore col massimo di col

③ `select max (col) from tab where col NOT IN (select Max (col) from tab)`

// Not in () è l' opposto di uguali. Questa query dice di non riportare il valore più alto (Not in). Ma, il seguente, ovvero il secondo

④ `WHERE valore_colonna BETWEEN valore_1 AND valore_2`
// Seleziona tutti i valori compresi fra i 2 valori

⑤ `SELECT col_1, col_2, col_3 WHERE
FROM TABELLA1 T INNER JOIN TABELLA2 D ON (FOREIGN KEY)
// Fonde 2 tabella, rimanendole T e D e le unisce attraverso
ON (T.ID = D.ID) WHERE col_1 IN (select Max (col1)
FROM TABELLA1) // ritorna il più alto valore di col1, rispetto
a col_2 e col_3, col_3 è in un altro tabella rispetto
a col_1 e col_2.`

`INNER JOIN` unisce le 2 colonne date in cui una è una PK e una è una FK di 2 tab diverse in cui `INNER JOIN (tab1.ID = tab2.ID)`

`GROUP BY` raggruppa in base ad una colonna, ovvero tutti gli stessi valori

SELECT col1
FROM tab1

UNION

SELECT col1
FROM tab2;

// bisogna avere lo stesso n° di col collegabili
per tabella e stesso tipo

JOIN

comb. ≠ row fra tabelle con colonne connestab.

SELECT tabonna-tab1 , colonna-tab2
FROM tab1

JOIN tab2

ON (FK)colonna-tab1 = colonna-tab2 (PK) // non per forza PK

LEFT JOIN include tutte le righe della prima tabella (FROM tab1)
INNER JOIN

RIGHT JOIN include tutte le righe della tab2

// Altrimenti se non ci sono collegamenti con PK fra le 2 tabelle
le escludi.

NESTED QUERIES

WHERE colonna IN (connessione)

// connessione è un altro query in un'altra tabella o nella stessa

SQL QUERIES

LIMIT n; n ∈ N

SELECT colonna AS nuovo_nome_colonna

DISTINCT col; // trova valori e non riporta doppi come unique()

FUNZIONI

SELECT COUNT(col1, ..)

Avg () // media

Sum

GROUP BY divide le funzioni in gruppi

WILDCARDS

definire pattern o associare ai dati

Where tab LIKE 'pattern';

se c'è una sotto seq.
sul record di tab 12
restituisce

LIKE '% LCC'

↑ ↑
any character pattern alla fine

'% ABC %' se c'è ABC nella seq.

'~~AAA%~~'

pattern che inizia con A, continua con 3

'A ___ - %'

rotture, poi - e poi qualcosa altro

carattere