DML: CRUD

Operazioni di modifica dell'istanza

Le operazioni DML sono:

- ***Inserimento di righe: `INSERT`***
- ***Eliminazione di righe: `DELETE`***
- ***Modifica di righe: `UPDATE`***

- Cambiano l'istanza
- Non cambiano lo schema
- Possono riguardare una o più righe di una tabella

- Non generano un risultato (Resultset)

Operazioni sui dati

Tra le DML rientra anche la

- ***Lettura di righe: `SELECT`***

- E' Un'interrogazione alla base di dati
- Non cambia l'istanza
- Non cambia lo schema
- E' l'unica istruzione che genera un Resultset

Operazioni CRUD sui dati

interrogazione:

- ***SELECT***

- ***modifica:***
  - ***INSERT, DELETE, UPDATE***

Vengono sintetizzate con l'acronimo CRUD
Create
Read
Update
Delete

INSERT

L'istruzione insert inserisce una riga all'interno di una tabella.

```
INSERT INTO nome_tabella[( lista attributi ) ]
VALUES( lista valori )
```

Creazione tabella Utenti (MySQL)

Nel database **prova** creare la tabella

`utenti(id,nome,cognome,età)`

dove id è chiave primaria autoincrementante;

```
CREATE TABLE utenti(
     id int unsigned PRIMARY KEY auto_increment,
     nome varchar(30),
     cognome varchar(30),
     età tinyint unsigned UNIQUE
);
```

Sintassi INSERT esplicita

```
INSERT INTO nome_tabella( lista attributi )
VALUES( lista valori )
```

La `lista attributi` contiene la lista delle colonne da popolare, ordinatamente, con la `lista valori`

`lista attributi e lista valori` dovranno coincidere come numero e come tipo, altrimenti si avrà un errore di matching.

Vedremo in seguito che la `lista attributi` può essere omessa (sintassi implicita)

Sintassi INSERT esplicita

EX Inserimenti corretti

```
INSERT INTO utenti(id,nome,cognome,età)
VALUES( 1,'Gino','Rossi',22 );

INSERT INTO utenti(id,nome,cognome,età)
VALUES( 2,'Lino','Rossi',null );

INSERT INTO utenti(nome,id)
VALUES('Pino',3);
```

```
+----+------+---------+------+
| id | nome | cognome | età  |
+----+------+---------+------+
|  1 | Gino | Rossi   |   22 |
|  2 | Lino | Rossi   | NULL |
|  3 | Pino | NULL    | NULL |
+----+------+---------+------+
```

Sintassi INSERT esplicita

EX Inserimenti errati

```
INSERT INTO utenti(id,nome,cognome,età)
VALUES( 1,'Gino');

INSERT INTO utenti(id,nome,cognome,età)
VALUES('Pino','Rossi',8,42 );

INSERT INTO utenti(nome,id)
VALUES(3,'Pino');
```

```
ERROR 1136 (21S01): Column count doesn't match value count at row 1
ERROR 1366 (HY000): Incorrect integer value: 'Pino' for column 'id' at row 1
```

Sintassi INSERT implicita

```
INSERT INTO nome_tabella
VALUES( lista valori )
```

La `lista attributi` c può essere omessa (sintassi implicita): in tal caso la `lista valori` dovrà contenere ordinatamente tutti i valori

Sintassi INSERT implicita

EX Inserimenti corretti

```
INSERT INTO utenti
VALUES( 4,'Rino','Rano',23 );

INSERT INTO utenti
VALUES( 5,'Gino','Verdi',null );
```

```
+----+------+---------+------+
| id | nome | cognome | età  |
+----+------+---------+------+
|  1 | Gino | Rossi   |   22 |
|  2 | Lino | Rossi   | NULL |
|  3 | Pino | NULL    | NULL |
|  4 | Rino | Rano    |   23 |
|  5 | Gino | Verdi   | NULL |
+----+------+---------+------+
```

Sintassi INSERT esplicita

EX Inserimenti errati

```
INSERT INTO utenti
VALUES( 9,'Gino');

INSERT INTO utenti
VALUES('Pino','Rossi',22,10);

INSERT INTO utenti
VALUES(10,'Pino',null);
```

```
ERROR 1136 (21S01): Column count doesn't match value count at row 1
ERROR 1366 (HY000): Incorrect integer value: 'Pino' for column 'id' at row 1
```

Campo AUTO_INCREMENT

- Quando si utilizza l'opzione `AUTO_INCREMENT` il DBMS si occuperà di assegnare un valore
  - Numerico
  - Intero
  - Positivo
  - Autoincrementato (a partire da 1)

Popolare una tabella con campo AUTO_INCREMENT

- Il funzionamento del campo `AUTO_INCREMENT` è assimilabile a quello di un elimina code: il DBMS assegna un numero ad ogni tupla **prima** che essa venga inserita nel database

- Per far generare l'id al DBMS ci sono due strategie:

1)Passare il valore null per il campo id oppure

2)Non fornire il valore dell'attributo `id`

Popolare una tabella con campo AUTO_INCREMENT

- passo null e il dbms genera l'id

```
INSERT INTO utenti VALUES(null,'cino','rossi',11);  //OK
INSERT INTO utenti VALUES('nino','verdi',12);  //FALLISCE
```

```
INSERT INTO utenti (id,nome,cognome,età) VALUES(null,'pino','rossi',NULL);//OK
INSERT INTO utenti (nome,cognome,età) VALUES(null,'pino','rossi',NULL);//NO
```

- visualizzazione dei valori inseriti

```
select * from utenti;
+----+------+---------+------+
| id | nome | cognome | età  |
+----+------+---------+------+
| 55 | cino | rossi   |   11 |
| 56 | pino | rossi   | NULL |
+----+------+---------+------+
```

Popolare una tabella con campo AUTO_INCREMENT

- NON passo il campo id e NON ne passo il valore:il DBMS genera l'id

```
INSERT INTO utenti (nome,cognome) VALUES('lino','rossi'); //OK
INSERT INTO utenti (nome,cognome) VALUES(null,'lino','rossi'); //NO
```

- visualizzazione dei valori inseriti

```
select * from utenti;
+----+------+---------+------+
| id | nome | cognome | età  |
+----+------+---------+------+
| 55 | cino | rossi   |   11 |
| 56 | pino | rossi   | NULL |
| 57 | lino | rossi   | NULL |
+----+------+---------+------+
```

Inserimento di una data (MySQL)

Le date vanno inserite tra apici, come se fossero stringhe

Il formato di default della data di MySQL è

```
'YYYY-MM-DD hh:mm:ss'
```

```
INSERT INTO utente(data_ora_nascita)
VALUES ('2008-06-18 10:34:09');
```

Esercizio

- Aggiungere una colonna "nascita" alla tabella `utenti`
- Aggiungere te stesso alla tabella;

Esercizio Mysql

```
alter table utenti add column nascita date;
```

```
MYSQL
insert into utenti (nome,cognome,nascita)
  VALUES ("gino","pilotino","2000-01-01");
```

```
select * from utenti;
```

Data e ora di sistema MySQL :`now()`

La funzione `now()` restituisce data e ora correnti

Aggiungendo alla tabella una colonna di tipo `datetime` può essere utilizzata per inserire data e ora di inserimento di un record nel DB

```
insert into utenti (nome,cognome,nascita) VALUES
   ("Pino","Silvestre",now());

select * from utenti;
```

inserimento multiplo dati (bulk)

L'inserimento bulk consente di inserire più righe con una singola query

Garantisce elevate performance di inserimento rispetto all'inserire singolarmente i record.

I record dovranno avere tutti lo stesso numero di valori

inserimento multiplo dati (bulk)

```
INSERT INTO utenti (nome,cognome,nascita)
VALUES ('Dario','Lampa','2023-01-13'),
       ('Dina','Lampa',now());
```

DELETE

L'istruzione DELETE elimina una o più righe di una tabella.

La clausola WHERE specifica le condizioni di eliminazione delle righe.

```
DELETE FROM Tabella
[ WHERE Condizione ]
```

DELETE

DELETE FROM Tabella

[WHERE Condizione]

esempi:

```
DELETE FROM AUTO
WHERE id=5;

DELETE FROM AUTO
WHERE colore is null;

DELETE FROM AUTO
WHERE colore != 'rosso';

DELETE FROM AUTO
WHERE MARCA='FIAT' AND COLORE='BIANCO';
```

Eliminazione di righe

DELETE FROM Tabella

[WHERE Condizione]

esempi:

```
DELETE FROM Persone
WHERE Eta < 35;

DELETE FROM Persone
WHERE reddito IS NOT NULL ;

DELETE FROM categoria;
```

Eliminazione, commenti

- La clausola WHERE permette di selezionare le righe da eliminare

- Se la clausola WHERE viene omessa l'operazione di delete riguarderà tutte le righe della tabella: la tabella verrà svuotata completamente

Indecisione DELETE e DROP

Spiegare le differenze (ambito, effetti, sintassi, esempi) tra le due istruzioni

Update

L'istruzione UPDATE modifica righe esistenti, selezionandole mediante la clausola WHERE

la modifica è definitiva e del valore precedente non vi sarà più traccia

```
UPDATE NomeTabella
SET  Attributo = <valore>,Attributo = <valore>
 [ WHERE Condizione ]
```

```
UPDATE user
SET reddito=20
WHERE id=5;
```

Update

```
UPDATE user
SET nome='Mario', reddito=30
WHERE id=5;
```

Aggiorna il valore corrente di `nome a 'Mario'` e il `reddito` a 30 della tabella `user` solo della riga con id=5;

Update

```
UPDATE user
SET data_ora=now();      MySQL
```

```
UPDATE user
SET data_ora=SYSDATE;      Oracle
```

Aggiorna il valore corrente di `data_ora` della tabella `user` con data e ora di sistema di tutte le righe della tabella.

Update

```
UPDATE persone
SET età = 26, reddito = 14
WHERE nome = 'pino';
```

```
UPDATE Persone
SET Reddito = Reddito * 1.1
WHERE Eta < 30;
```

```
UPDATE utenti
SET nascita=now();
```

```
UPDATE Persone
SET Reddito = 10;
```

Update

- La clausola `WHERE` permette di selezionare le righe da aggiornare

- Se la clausola `WHERE` viene omessa l'operazione di `update` riguarderà tutte le righe della tabella

Riepilogo

- Quali sono istruzioni CRUD
- A cosa serve l'istruzione `DROP`
- A cosa serve l'istruzione `DELETE`
- A cosa serve l'istruzione `UPDATE`
- Parlare delle differenze tra l'istruzione `INSERT` e l'istruzione `CREATE`
- In quali istruzioni è prevista la clausola `WHERE`?
- Quali istruzioni DML modificano lo schema
- Quali istruzioni DML modificano l'istanza
- Quale istruzione DML restituisce un ResultSet