lezione 22

Istruzioni del DML di SQL

Finora abbiamo trattato quella parte di SQL che assolve alle funzioni di DDL (*Data Definition Language*), ovvero la definizione dei dati senza nessuna possibilità di manipolarli, come per esempio cambiarne il valore. In questa e nelle successive lezioni tratteremo, invece, quella parte di SQL che assolve alle funzioni di DML (*Data Manipulation Language*). Questa parte di SQL ci consente di *inserire* dati nelle tabelle, di *modificarli* e, soprattutto, di *visualizzarli* attraverso opportune *interrogazioni*. Le corrispondenti istruzioni che assolvono a tale scopo sono INSERT, UPDATE, DELETE e SELECT.

Inserire i valori in una tabella

I valori delle righe possono essere inseriti utilizzando il comando **INSERT INTO**, la cui sintassi è:

```
INSERT INTO <NOMETABELLA>((<Attributo1>, <Attributo2>, ...,
    <AttributoN>))
    VALUES(<Valore1>, <Valore2>, ..., <ValoreN>);
```

Se non è presente la lista degli attributi, si intende che i valori specificati devono corrispondere in ordine, tipo e numero a quelli specificati nella dichiarazione della tabella <NOMETABELLA>.
Se, invece, si specifica una lista di attributi, l'ordine e il tipo dei valori dovranno rispettare questa lista. Si assume che il valore per gli attributi omessi sia NULL.
Inizializziamo la tabella *Azienda* del nostro solito esempio inserendo la seguente riga:

```
INSERT INTO AZIENDA
  VALUES("C001", "A&B Tessile", 1500000.00, 80);
```

Se, invece, utilizziamo l'istruzione:

```
INSERT INTO AZIENDA
  VALUES("C002");
```

inseriremo il codice "C002" come valore dell'attributo *CodAzienda* e NULL come valore degli altri attributi.

stop

Anche per inserire i valori relativi a un attributo di tipo DATE si devono utilizzare gli apici singoli o doppi.

Modificare i valori delle righe di una tabella

Per aggiornare una o più righe di una tabella utilizziamo il comando **UPDATE**, la cui sintassi è:

```
UPDATE <NOMETABELLA>
SET <Attributo1> = <Espressione1>,
    <Attributo2> = <Espressione2>,
    ...,
    <AttributoN> = <EspressioneN>
(WHERE <Condizione>);
```

dove gli attributi di <NOMETABELLA> (specificati nella clausola SET), vengono aggiornati con i valori delle corrispondenti espressioni, in tutte le righe che soddisfano la <Condizione>.

Riferiamoci sempre al nostro esempio. Per cambiare la ragione sociale dell'azienda A001, scriveremo:

```
UPDATE AZIENDA
SET RagioneSociale = "NuovaElettronica 3000"
WHERE CodAzienda = "A001";
```

Per incrementare di 100 euro il valore dello stipendio di tutti i dipendenti, scriveremo:

```
UPDATE DIPENDENTE
SET StipendioLordo = StipendioLordo + 100.0;
```

Per incrementare del 10% lo stipendio lordo di quei dipendenti che percepiscono meno di 1000 euro, scriveremo:

```
UPDATE DIPENDENTE
SET StipendioLordo = StipendioLordo + (StipendioLordo * 0,1)
WHERE StipendioLordo < 1000;
```

Per cancellare una o più righe di una tabella utilizziamo il comando **DELETE**, la cui sintassi è:

```
DELETE FROM <NomeTabella>
WHERE <Condizione>;
```

In questo modo si eliminano tutte le righe di <NomeTabella> che soddisfano la <Condizione>.

Cancellare le righe di una tabella

Per cancellare i dipendenti assunti prima del 31 dicembre 1990, scriveremo:

```
DELETE FROM DIPENDENTE
WHERE DataAssunzione <= "1990/12/31";
```

Da notare che la data è stata espressa nella forma "anno/mese/giorno".

Per cancellare i dipendenti dell'azienda caratterizzata dal codice A001 che hanno uno stipendio lordo maggiore di 6000 euro, scriveremo:

```
DELETE FROM DIPENDENTE
WHERE(CodAzienda = "A001") AND (StipendioLordo > 6000);
```

ISTRUZIONI DEL DML DI SQL


Esercizio commentato

Prendiamo in esame il seguente esercizio, al quale faremo riferimento durante la trattazione di tutte le sezioni di esempi che svilupperemo nelle prossime lezioni. Dell'esercizio proposto forniamo lo schema relazionale (dove le chiavi esterne sono riportate in corsivo) del quale, naturalmente, possono esistere altre soluzioni altrettanto corrette. Noi abbiamo preferito riferirci a questa soluzione, che non prevede alcune relazioni (per esempio quella dei *Registi*, quella delle *Categorie* e altre) e contiene attributi (per esempio *CognomeRegista*) che potrebbero risultare incongruenti. Tale scelta si basa su motivazioni didattiche, poiché questo schema risulta più semplice e più utile per un efficace apprendimento di questa parte del linguaggio SQL.

ATTORE(CodiceAttore, Cognome, Nome, Sesso, AnnoNascita, Nazionalità)
FILM(CodiceFilm, Titolo, AnnoProduzione, LuogoProduzione, CognomeRegista, Genere)
CINEMA(CodiceCinema, Nome, Posti, Città)
INTERPRETA(*CodiceAttore*, *CodiceFilm*, Personaggio)
PROGRAMMATO(*CodiceFilm*, *CodiceCinema*, Incasso, DataProiezione)

1. Creiamo le tabelle.

```
CREATE TABLE ATTORE
(
   CodiceAttore      CHAR(5),
   Cognome           CHAR(30)      NOT NULL,
   Nome              CHAR(20)      NOT NULL,
   Sesso             CHAR(1),
   AnnoNascita       INT(4),
   Nazionalità       CHAR(20),
   PRIMARY KEY(CodiceAttore)
);
```

```
CREATE TABLE FILM
(
   CodiceFilm        CHAR(5),
   Titolo            CHAR(40)      NOT NULL,
   AnnoProduzione    INT(4),
   LuogoProduzione   CHAR(30),
   CognomeRegista    CHAR(20)      NOT NULL,
   Genere            CHAR(15)      NOT NULL,
   PRIMARY KEY(CodiceFilm)
);
```

```
CREATE TABLE CINEMA
(
   CodiceCinema      CHAR(5),
   Nome              CHAR(20)      NOT NULL,
   Posti             INT(4),
   Città             CHAR(30)      NOT NULL,
   PRIMARY KEY(CodiceCinema)
);
```

```
CREATE TABLE INTERPRETA
(
    CodiceAttore        CHAR(5),
    CodiceFilm          CHAR(5),
    Personaggio         CHAR(30)        NOT NULL,
    PRIMARY KEY(CodiceAttore, CodiceFilm),
    FOREIGN KEY(CodiceAttore) REFERENCES ATTORE(CodiceAttore),
    FOREIGN KEY(CodiceFilm) REFERENCES FILM(CodiceFilm),
                ON UPDATE CASCADE,
    ON DELETE CASCADE
);
```

```
CREATE TABLE PROGRAMMATO
(
    CodiceFilm          CHAR(5),
    CodiceCinema        CHAR(5),
    Incasso             DECIMAL(8,2),
    DataProiezione      DATE,
    PRIMARY KEY(CodiceFilm, CodiceCinema),
    FOREIGN KEY(CodiceFilm)             REFERENCES FILM(CodiceFilm),
    FOREIGN KEY(CodiceCinema)           REFERENCES CINEMA(CodiceCinema),
            ON UPDATE CASCADE,
            ON DELETE CASCADE
);
```

2. Aggiungiamo le informazioni di un attore nella tabella *Attore*:

```
INSERT INTO ATTORE
VALUES('A0001', 'Verdone', 'Carlo', 'M', 1950, 'Italiana');
```

oppure:

```
INSERT INTO ATTORE(CodiceAttore, Cognome, Nome, Sesso, AnnoNascita,
                   Nazionalità)
VALUES('A0001', 'Verdone', 'Carlo', 'M', 1950, 'Italiana');
```

3. Aggiungiamo la colonna *FilmRealizzati* alla tabella *Attore*:

```
ALTER TABLE ATTORE
ADD COLUMN FilmRealizzati INT(3);
```

4. Cancelliamo la colonna *FilmRealizzati* dalla tabella *Attore*:

```
ALTER TABLE ATTORE
DROP COLUMN FilmRealizzati;
```

5. Modifichiamo le righe della tabella *Cinema*, aggiungendo 10 posti a ogni sala cinematografica:

```
UPDATE CINEMA
SET Posti = Posti + 10;
```

6. Modifichiamo le righe della tabella *Cinema*, aggiungendo 10 posti alle sole sale cinematografiche che hanno un numero di posti compreso tra 100 e 200:

```
UPDATE CINEMA
SET Posti = Posti + 10
WHERE Posti BETWEEN 100 AND 200;
```