La componente DML del linguaggio SQL (parte 1)

INSERT, UPDATE, DELETE

(pag. 112-115)

SELECT

(pag. 116-117)

Mappa dei principali comandi SQL

DDL

- CREATE
  - DATABASE
  - TABLE
  - INDEX
  - VIEW
- ALTER
  - TABLE
  - VIEW
- DROP
  - DATABASE
  - TABLE
  - INDEX
  - VIEW
- RENAME
  - TABLE

DML

- INSERT (INTO)
- UPDATE (SET WHERE)
- DELETE (FROM WHERE)
- SELECT

DCL

- CREATE
  - USER
- GRANT
- REVOKE

TCL

- COMMIT
- ROLLBACK
- SAVEPOINT

Inserire una nuova t-upla in una relazione

If absent, the values must be in the same order of the table complete structure (as created using CREATE TABLE statement)

```
INSERT INTO tablename [(attr1, attr2, …, attrN)]
    VALUES (val1, val2, …, valN) [, (val1, val2, …, valN), etc.];
```

It is possible to insert more than one row

- Se un campo non è presente nella INSERT, il valore viene settato al valore di default (se presente nella query di definizione della relazione CREATE TABLE), oppure al valore speciale NULL.

Esempi:

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

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

Aggiornare i dati presenti in una relazione

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

- 1° esempio – si vuole modificare solo una t-upla, dato il valore della chiave primaria

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

Aggiornare i dati presenti in una relazione

- 2° esempio – Si vuole modificare TUTTI i record nello stesso modo, utilizzando una formula:

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

- 3° esempio – Si vogliono modificare tutte le righe che soddisfano una certa condizione:

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

Cancellare una t-upla

Si utilizza l'istruzione **DELETE FROM**:

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

- Esempi:

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

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

Interrogare la base di dati: il comando SELECT

Interrogare una base di dati

- Si fa attraverso il comando **SELECT**
- Il quale consente di effettuare delle operazioni su una o più relazioni del mio database, e visualizzarne il risultato
- **Il risultato sarà sempre una relazione**, composta da un certo numero di righe e di colonne, ottenute a partire dai dati contenuti nelle tabelle, e che verrà rappresentata nel nostro DBMS sempre sotto forma tabellare

- Le operazioni che effettuiamo devono seguire le regole di quella che si chiama **ALGEBRA RELAZIONALE** (algebra delle relazioni)

Interrogare una base di dati

- Prima di esplorare le regole dell'algebra relazionale cominciamo però ad imparare ad usare il comando **SELECT**

Sintassi completa

```
SELECT
    [ALL | DISTINCT | DISTINCTROW]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    espr_select [, espr_select ...]
    [ INTO OUTFILE 'nome_file' [export_options]
    | INTO DUMPFILE 'nome_file'
    | INTO nome_var [, nome_var] ]
    [ FROM riferimenti_tabelle
    [WHERE condizione_where]
    [GROUP BY {nome_col | espr | posizione} [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {nome_col | espr | posizione} [ASC | DESC], ...]
    [LIMIT {[scarto,] num_righe | num_righe OFFSET scarto}]
      [PROCEDURE nome_procedura(lista_argomenti)]
      [FOR UPDATE | LOCK IN SHARE MODE] ]
```

Sintassi di base

```
SELECT [DISTINCT] <attributes_list>
FROM <table_name>
[WHERE <condition>];
```

- La clausola opzionale **DISTINCT** specifica che eventuali righe vengono scartate

Esercizi

- Qual è il risultato di queste query?

```
SELECT Titolo
FROM FILM;
```

```
SELECT DISTINCT Titolo
FROM FILM;
```

```
SELECT *
FROM FILM;
```

Alias e calcoli aritmetici

È possibile assegnare un diverso nome (ALIAS) alle colonne di una relazione ottenuta come risultato di una query di interrogazione:

```
SELECT Nome, Citta, Posti AS "Numero posti"
FROM CINEMA;
```

È anche possibile effettuare dei veri e propri calcoli aritmetici, usando operatori e funzioni (NO FUNZIONI DI AGGREGAZIONE):

```
SELECT Incasso * 0,8 AS "Incasso netto"
FROM PROGRAMMATO;
```

ALIAS per i nomi delle tabelle

Spesso è utile utilizzare delle abbreviazioni per fare riferimento ai nomi delle tabelle. Ad esempi
l'interrogazione vista poco fa:

```
SELECT Titolo
FROM FILM;
```

che consentiva di visualizzare il titolo di tutti i film potrebbe essere riscritta nel seguente modo:

```
SELECT FILM.Titolo
FROM FILM;
```

o meglio:

```
SELECT F.Titolo
FROM FILM F;
```

È obbligatorio indicare il nome della tabella nella clausola SELECT quando ci sono più attributi con lo stesso nome provenienti da tabelle differenti.

Se il nome della tabella è particolarmente lungo, le si può dare un alias (che vale solo in questa query).

Focus: il valore speciale NULL (1)

Da ricordare:

- È il valore di default se non viene inserito alcun valore per un dato attributo, e non era stato stabilito nessun valore di default per quel campo in sede di creazione della tabella

- Significa che il valore per quell'attributo è sconosciuto

- Non è equivalente al valore 0 (zero) per i dati numerici, e non è equivalente alla stringa vuota "" per i valori alfanumerici

Focus: il valore speciale NULL (2)

```
SELECT CodCli, Cognome, Nome
FROM CLIENTI
WHERE Telefono = NULL ;
```

Sbagliato

- Il valore NULL è differente dagli altri: si devono usare le espressioni **IS NULL** e **IS NOT NULL** per scrivere condizioni che coinvolgono questo valore

```
SELECT CodCli, Cognome, Nome
FROM CLIENTI
WHERE Telefono IS NULL;
```

Giusto