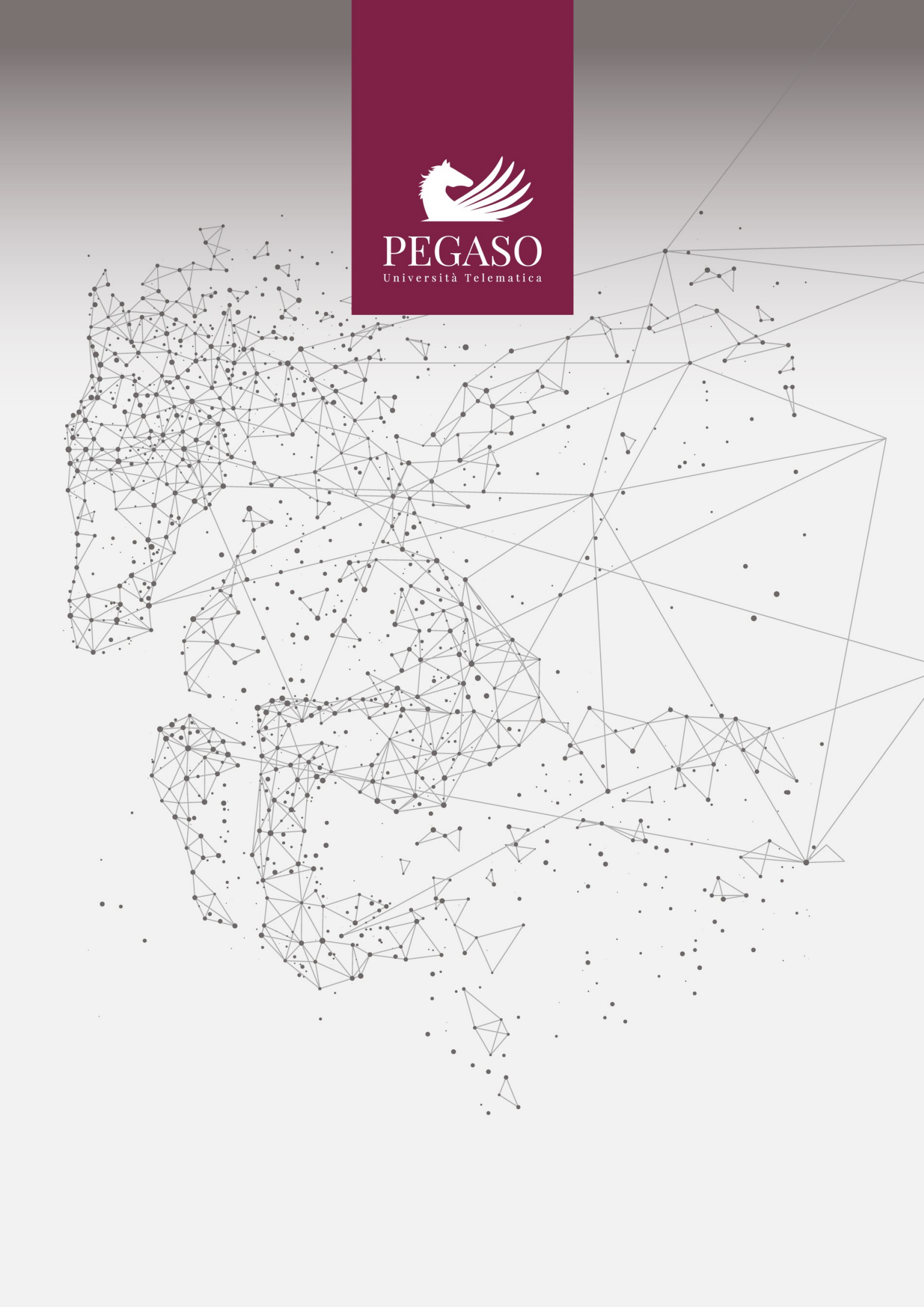
PEGASO
Università Telematica

Indice

1. Introduzione

Il linguaggio SQL possiede i comandi per inserire, modificare ed eliminare le righe di una tabella, cioè le funzioni di linguaggio DML (Data Manipulation Language). In questa unità didattica trattiamo quindi i comandi per farlo: aggiornamento, cancellazione e inserimento. Analizziamo separatamente i singoli comandi dopo aver introdotto le query nidificate.

2. Interrogazioni nidificate

Il linguaggio SQL possiede i comandi per inserire, modificare ed eliminare le righe di una tabella, cioè le funzioni di linguaggio DML (Data Manipulation Language). In questa unità didattica trattiamo quindi i comandi per farlo: aggiornamento, cancellazione e inserimento. Analizziamo separatamente i singoli comandi dopo aver introdotto le query nidificate.

In questo paragrafo affrontiamo lo studio delle interrogazioni nidificate ovvero delle subqueries.

Una "subquery" è una query inclusa in un'altra, ovvero un'interrogazione all'interno di altre interrogazioni. Una subquery ritorna dei dati (tabelle o dati singoli) necessari all'esecuzione di query ad un livello più alto. Una query così strutturata rispetta l'ordine gerarchico di esecuzione: Oracle ad esempio esegue sempre prima le query più interne e, una volta completate, quelle situate a "livelli" superiori. Una query situata a un livello più basso rispetto ad un'altra viene definita "child", al contrario la query situata ad un livello più alto rispetto alla child viene definita "parent".

Le subquery possono essere usate sia nella clausola FROM sia nella clausola WHERE di una SELECT.

In quest'ultimo caso si parla di **query nidificate** e possiamo avere al massimo 255 livelli di nidificazione ("strati" di interrogazione)[1]. Quindi il linguaggio SQL permette di scrivere interrogazioni che contengono altre interrogazioni al proprio interno. Il caso più comune è quello della clausola WHERE, dove si confronta un predicato con il risultato dell'esecuzione di una interrogazione SQL. L'espressione che compare come primo membro del confronto è il semplice nome di un attributo. A questo punto viene confrontato con il risultato di una interrogazione. In generale però il risultato di una interrogazione è un insieme di valori. Come far fronte a tale problema? Esistono le parole chiave ALL e ANY:

- **Parola chiave ANY**
  - Specifica che la riga soddisfa la condizione se risulta vero il confronto tra il valore di attributo per la riga e almeno uno degli elementi restituiti dall'interrogazione.
- **Parola chiave ALL**
  - La riga soddisfa la condizione solo se tutti gli elementi restituiti dall'interrogazione rendono vero il confronto.
  - La sintassi richiede la compatibilità di dominio tra l'attributo restituito dall'interrogazione nidificata e l'attributo con cui avviene il confronto

Vediamo adesso un esempio.

[1] https://www.html.it/pag/16940/query-nidificate/

Interrogazione 37: estrarre gli impiegati che lavorano in dipartimenti situati a Milano, dalla tabella di Figura 1 e Figura 2.

Nome	Cognome	Dipart	Ufficio	Stipendio	Città
Mario	Rossi	Amministrazione	10	45	Milano
Carlo	Bianchi	Produzione	20	36	Torino
Giovanni	Verde	Amministrazione	20	40	Roma
Franco	Neri	Distribuzione	16	45	Napoli
Carlo	Rossi	Direzione	14	80	Milano
Lorenzo	Gialli	Direzione	7	73	Genova
Paola	Rosati	Amministrazione	75	40	Venezia
Marco	Franco	Produzione	20	46	Roma

Figura 1: tabella Impiegato

Nome	Indirizzo	Città
Amministrazione	Via Livio 32	Milano
Produzione	Via Lavier 4	Torino
Distribuzione	Via Segre 9	Roma
Direzione	Via Livio 32	Milano
Ricerca	Via Venosa 6	Milano

Figura 2: tabella Dipartimento.

La query è la seguente:

- SELECT *
- FROM Impiegato
- WHERE Dipart=ANY (SELECT Nome
- FROM Dipartimento
- WHERE Città='Milano')

I record interessati nel result-set sono evidenziati nella Figura 3.

Nome	Cognome	Dipart	Ufficio	Stipendio	Città
Mario	Rossi	Amministrazione	10	45	Milano
Carlo	Bianchi	Produzione	20	36	Torino
Giovanni	Verde	Amministrazione	20	40	Roma
Franco	Neri	Distribuzione	16	45	Napoli
Carlo	Rossi	Direzione	14	80	Milano
Lorenzo	Gialli	Direzione	7	73	Genova
Paola	Rosati	Amministrazione	75	40	Venezia
Marco	Franco	Produzione	20	46	Roma

Figura 3: risultato della query. Evidenziati i record interessati

3. Inserimento dati

In questo paragrafo illustriamo l'inserimenti dei dati in tabelle. Il comando di inserimento righe in una base di dati presenta due sintassi:

Prima forma:

```
INSERT INTO Tabella [(Attributi)]
        VALUES(Valori )
```

Oppure:

Seconda forma:

```
INSERT INTO Tabella [(Attributi)]
         SELECT ...
```

La prima forma permette di inserire singole righe all'interno delle tabelle:

```
INSERT INTO Dipartimento(Nome,Città)
VALUES ('Produzione','Torino')
```

Si pone comunque un problema riguardante gli attributi che possono mancare: si aggiornano un sottoinsieme di attributi e non tutti. Che valore prendono gli altri? Si parla di attributi mancanti. Questi attributi:

- Prendono un valore di default oppure:
- Prendono il valore NULL (se permesso) altrimenti l'operazione viene rifiutata.

La seconda forma permette di aggiungere insiemi di righe estratti dal contenuto della base di dati. La forma della query è la seguente:

```
INSERT INTO ProdottiMilanesi
(SELECT Codice,Descrizione
FROM Prodotto
WHERE LuogoProd='Milano')
```

Questa query Inserisce nella tabella ProdottiMilanesi il risultato della selezione dalla relazione Prodotto di tutte le righe aventi 'Milano' come valore dell'attributo LuogoProd.

Osservazioni:

- l'ordinamento degli attributi (se presente) e dei valori è significativo;
- le due liste debbono avere lo stesso numero di elementi;
- se la lista di attributi è omessa, si fa riferimento a tutti gli attributi della relazione, secondo l'ordine con cui sono stati definiti;
- se la lista di attributi non contiene tutti gli attributi della relazione, per gli altri viene inserito un valore nullo (che deve essere permesso) o un valore di default.

4. Cancellazione dei dati

Qui affrontiamo il comando SQL per cancellare dati. Il comando DELETE elimina righe da una tabella della base di dati con la sintassi:

DELETE FROM Tabella

[WHERE Condizione]

Esempio:

DELETE FROM Persone

WHERE Eta < 35

Questa istruzione:

- Elimina le ennuple che soddisfano la condizione.
- Può causare (se i vincoli di integrità referenziale sono definiti con politiche di reazione cascade) eliminazioni da altre relazioni.
- Ricordare: se la WHERE viene omessa, si intende where true.
- La condizione rispetta la sintassi della SELECT, per cui possono comparire al suo interno anche interrogazioni nidificate che fanno riferimento ad altre tabelle.

4.1 Differenza tra DROP e DELETE

Da notare la differenza tra il comando **DROP** ed il comando **DELETE.** L'operazione:

DELETE FROM Dipartimento

Che provoca i seguenti cambiamenti:

- Elimina tutte le righe della tabella Dipartimento ed eventualmente tutte le tabelle ad essa correlate (politica CASCADE).
- Lo schema della base dati rimane immutato, viene modificata solo l'istanza.

Invece il comando:

DROP TABLE Dipartimento CASCADE

Ha le seguenti caratteristiche:

- Ha lo stesso effetto del comando DELETE.
- Lo schema della base dati viene modificato eliminando dallo schema, oltre alla tabella Dipartimento, tutte le viste e le tabelle che nella definizione fanno riferimento ad essa.

Invece il comando:

DROP TABLE Dipartimento RESTRICT

Fallisce se vi sono righe nella tabella Dipartimento.

5. Modifica dati

Dopo aver visto le modalità per inserire dati nel database e per interrogarlo, passiamo in rassegna le istruzioni per modificarne i dati. Il comando UPDATE serve a modificare dati già inseriti nelle tabelle del database. La sua sintassi è la seguente:

UPDATE *NomeTabella*

SET Attributo = < Espressione |SELECT …| NULL| DEFAULT >

[WHERE *Condizione*]

Il significato del comando è quello di modificare i campi indicati nella clausola SET delle righe della tabella *NomeTabella* che soddisfano il predicato indicato nella clausola WHERE. Su quest'ultima non ci dilunghiamo perché per essa valgono le stesse considerazioni fatte per il comando SELECT. Diciamo soltanto che **se viene omessa, l'aggiornamento avverrà su tutte le righe della tabella**. Di conseguenza bisogna porre particolare attenzione al momento del lancio di questo comando. L'utilizzo tipo è la modifica di un singolo record, utilizzando l'id, ovvero la chiave primaria. Quindi, il comando UPDATE permette di aggiornare uno o più attributi delle righe di *NomeTabella* che soddisfano l'eventuale *Condizione*.

Esempi:

UPDATE Persone SET Reddito = 45

WHERE Nome = 'Piero'

Tutte le persone di nome 'Piero' hanno il reddito settato al valore 45.

UPDATE Persone

SET Reddito = Reddito * 1.1

WHERE Eta < 30

Aumenta il reddito di tutte le persone con età inferiore a 30. Il nuovo valore cui viene posto l'attributo può essere uno dei seguenti casi:

- Il risultato della valutazione di una espressione sugli attributi di una tabella.
- Il risultato di una generica interrogazione SQL.
- Il valore nullo.
- Il valore di default per il dominio.

Bibliografia

- Atzeni P., Ceri S., Fraternali P., Paraboschi S., Torlone R. (2018). Basi di Dati. McGraw-Hill Education.
- Batini C., Lenzerini M. (1988). Basi di Dati. In Cioffi G. and Falzone V. (Eds). Calderini. Seconda Edizione.