Corso di Laurea in Ingegneria Gestionale
Sapienza Università di Roma

Corso di Basi di Dati
A.A. 2019/2020

5 – SQL : Definizione e manipolazione dei dati

Tiziana Catarci

SQL : Structured Query Language

- SQL non è un semplice linguaggio per le interrogazioni...
- ...ma contiene 3 sotto-linguaggi :
  - **DDL (Data Definition Language) :** linguaggio che permette di creare\eliminare\modificare gli oggetti in un database.
    - i comandi DDL permettono la definizione dello schema di una base di dati (*livello intensionale*).
  - **DML (Data Manipulation Language) :** linguaggio che permette di leggere\inserire\modificare\eliminare i dati di un database.
    - i comandi DML permettono di interrogare e modificare un'istanza di una base di dati (*livello estensionale*).
  - **DCL (Data Control Language) :** permette di gestire gli utenti ed i permessi.

SQL : Alcune Notazioni

- Notazione utilizzata per specificare la sintassi dei comandi:
  - Le parentesi quadre **[ ]** indicano che il termine contenuto al suo interno è opzionale, cioè può non comparire o comparire una sola volta.
  - Le parentesi graffe **{ }** indicano che il termine racchiuso può non comparire o essere ripetuto un numero arbitrario di volte.
  - Le barre verticali **|** indicano che deve essere scelto **solo uno** tra i termini separati dalle barre.
  - Le parentesi tonde **( )** dovranno essere intese **sempre** come termini del linguaggio SQL e **non come simboli** per la definizione della grammatica.

Creazione di una tabella

- L'istruzione più importante del DDL di SQL è :

CREATE TABLE

- definisce la struttura di uno **schema di relazione** (specificandone gli **attributi** e un insieme - eventualmente vuoto - di **vincoli**).
- crea un'istanza vuota dello schema.
- Il nome della tabella può essere formato da lettere o numeri, ma il primo carattere deve essere sempre una lettera.
- **Sintassi :**

```
CREATE TABLE NomeTabella (
    NomeAttributo Dominio [Vincoli]
                    ....
 { ,NomeAttributo Dominio [Vincoli] }
          { ,Vincolo di tabella }
)
```

Deve essere **diverso** dai nomi delle altre tabelle **nello stesso database**.

Deve essere **diverso** dai nomi degli altri attributi **nella stessa tabella**.

CREATE TABLE, esempio

CREATE TABLE *NomeTabella* (
NomeAttributo Dominio {Vincoli }
....
{ ,*NomeAttributo Dominio* {Vincoli } }
{ ,Vincoli di tabella }
)

```
CREATE TABLE Impiegato (
Matricola   char(6) PRIMARY KEY,
Nome        varchar(20) NOT NULL,
Cognome     varchar(20) NOT NULL,
Dipart      varchar(15),
Stipendio   numeric(9) DEFAULT 0,
Salario     real,
FOREIGN KEY(Dipart) REFERENCES Dipartimento(NomeDip),
UNIQUE(Cognome,Nome))
```

ATTENZIONE

- Una tabella in SQL è definita come un **multinsieme** di tuple.
- In particolare, se una tabella non ha una *primary key* o un insieme di attributi definiti come *chiave*, allora potranno comparire due tuple uguali nella tabella; ne segue che:
  - **una tabella SQL non è in generale una relazione matematica.**

- Se invece una tabella ha una *primary key* o un insieme di attributi definiti come *chiave*, allora non potranno mai comparire nella tabella due tuple uguali; per questo,
  - **è indispensabile definire almeno una primary key per ogni tabella SQL**.

Domini

- **Domini elementari** (predefiniti)
  - **Carattere**: singoli caratteri o stringhe, anche di lunghezza variabile
  - **Bit**: singoli booleani
  - **Numerici**: esatti e approssimati
  - **Data**, **ora**, **intervalli di tempo**
  - Introdotti in SQL:1999
    - **Boolean**
    - **BLOB**, **CLOB** (binary/character large object): per grandi immagini e testi
- **Domini definiti dall'utente** (semplici, ma riutilizzabili)

SQL : Domini Predefiniti

- **Carattere :** singoli caratteri o stringhe
  - **char(n) o character(n)** – stringhe di lunghezza fissa.
  - **varchar(n)** – stringhe di lunghezza variabile.
    - ***n** è il numero massimo di caratteri che si vogliono memorizzare.*
    - la lunghezza di un campo definito con dominio CHAR(n) è **precisamente di dimensione n bytes**, indipendentemente dal dato inserito.
    - la lunghezza di un campo definito con dominio VARCHAR(n) **assume la dimensione del dato inserito + 1 byte di prefisso**.

Il DBMS elimina automaticamente i caratteri "non utilizzati" di troppo.

Possibile rappresentazione dei domini *char* e *varchar* in un calcolatore.

Valore	CHAR(4)	Spazio Richiesto	VARCHAR(4)	Spazio Richiesto
''	' '	4 byte	''	1 byte
'ab'	'ab '	4 byte	'ab'	3 bytes
'abcd'	'abcd'	4 byte	'abcd'	5 bytes

SQL : Domini Predefiniti

Di default vale **SIGNED.**

UNSIGNED permette la rappresentazione dei soli interi positivi**.**

- **Numerici : valori numerici esatti**
  - **int (o smallint, bigint) [UNSIGNED]** – interi di lunghezza fissa.
  - utili nei casi in cui non interessa avere una rappresentazione della parte frazionaria.
  - lo standard non fissa la precisione dei tipi numerici, che dipende dalla rappresentazione interna del calcolatore.
- **Numerici : valori numerici esatti con eventuale parte frazionaria**.
  - **numeric (o numeric(p) o numeric(p,s)) :** valori numerici esatti (anche negativi) utilizzabili se interessa una rappresentazione della parte frazionaria.
    - **p** rappresenta il numero massimo di cifre definibili. **s** rappresenta la *scala,* ovvero il numero di cifre che devono comparire dopo la virgola (con ***s<=p***).
    - ***Esempio:***
      - ***numeric(3,1) : numeri da -99.9 a +99.9***
      - ***numeric(3,2) : numeri da -9.99 a +9.99***
      - ***numeric(3) : numeri da -999 a +999***

SQL : Domini Predefiniti

- **Numerici : valori numerici approssimati (utili per rappresentare grandezze fisiche).**
  - **float (real** è equivalente**), (float(m,d), double, double(m,d)) [UNSIGNED]:** numeri in virgola mobile.
  - ***m*** rappresenta la precisione – numero di cifre – dedicate alla mantissa. La mantissa è un valore frazionario.
  - ***d*** è la precisione dell'esponente. L'esponente è un numero intero.
  - **FLOAT è a "precisione singola", DOUBLE sono invece a "precisione doppia".**
  - Per entrambi i tipi di dato l'uso di UNSIGNED disabilita i valori negativi, ma non ha effetto sui valori massimi positivi memorizzabili
    - ***Esempio:***
      - **0.17E16** ***rappresenta il valore 1,7 x 10^(15)***
      - ***-0.4E-6 rappresenta il valore -4x10^(-7)***

Utili per rappresentare *flag,* che specificano se l'oggetto rappresentato da una tupla possiede o meno una certa proprietà.

- **bit, bit(n) :** sequenza fissa di n bit (valori appartenenti all'insieme <0,1>)
  - ***Esempio:*** ***bit(2) : <(00),(01),(10),(11)>***

SQL : Domini Predefiniti

ATTENZIONE
agli apici

- **Data e Ora :**
  - **date :**
    - ammette i campi *('anno-mese-giorno')* nel formato ***('aaaa-mm-gg').***
    - rappresentabili tutte le date da *'1000-01-01' (1° gennaio 1000) a '9999-12-31' (31 dicembre 9999).*
    - ***Esempio:***
      - ***INSERT into table name_table VALUES ('2009-03-26')***
  - **time :**
    - contiene un valore di tempo ('*ore:minuti:secondi*') nel formato ***('hh:mm:ss')***.
  - **timestamp :**
    - Visualizzato nel formato ***('AAAAMMGGhhmmss')***.
    - Comodo per memorizzare il momento dell'inserimento o aggiornamento di record, in quanto può essere assegnato automaticamente.

Valore di default :
timestamp corrente

Domini introdotti in SQL:1999

- **boolean :**
  - utilizzato per rappresentare i valori booleani (restrizione del dominio bit) *true* o *false*.

- **BLOB e CLOB :**
  - permettono di rappresentare oggetti di grandi dimensioni, costituiti da una sequenza arbitraria di valori binari *(blob – "binary large object")* o di caratteri (*clob – "character large object")*.
  - il sistema garantisce solo di memorizzarne il valore, ma **non permette che il valore venga utilizzato come criterio di selezione per le interrogazioni**.
  - utili per gestire contenuti di grandi dimensioni (semi-strutturati o multimediali – immagini, documenti, video) e la loro fruizione richiede l'uso di applicazioni specifiche.

Domini definiti dall'utente

- SQL permette di specificare *nuovi domini* utilizzando il comando **CREATE DOMAIN.**

CREATE DOMAIN *NomeDominio* **AS** *TipoDiDato*
[DEFAULT *ValoreDiDefault***]**
[CHECK *Valore***]**

- ***NomeDominio*** – nome del nuovo dominio.
- ***TipoDiDato*** – dominio elementare predefinito (INT,CHAR,...) o definito dall'utente in precedenza.
- **[DEFAULT *ValoreDiDefault*]** - usato per impostare un valore di default.
- **[CHECK *Valore*]** – condizioni che devono essere rispettate dai valori del dominio.

Domini definiti dall'utente

ESEMPIO :

```
CREATE DOMAIN ETAIMPIEGATI AS INTEGER
DEFAULT 31
check (VALUE >= 18 and VALUE <= 80)
```

VALUE = parola chiave utilizzata per indicare il valore del dominio in fase di definizione.

```
CREATE TABLE Impiegato(
ID INT,
Nome VARCHAR(20),
Età ETAIMPIEGATI,
Salario REAL
)
```

La dichiarazione di nuovi domini permette di associare un insieme di vincoli ad un dominio, il che è importante quando (per esempio) si deve ripetere la stessa definizione di attributo nell'ambito di diverse tabelle. Definendo un dominio apposito si rende la definizione più facilmente modificabile, **garantendo che la modifica si propaghi a tutte le tabelle in cui il dominio viene usato.**

Modifica e Cancellazione di domini

```
alter domain NomeDominio
        set DEFAULT ValoreDiDefault |
        DROP DEFAULT |
        add constraint [NomeVincolo] DefVincolo |
        drop constraint NomeVincolo
```

Si noti che quando si modifica un dominio, la "versione modificata" deve essere soddisfatta dai dati già presenti. In caso contrario, la modifica viene rifiutata.

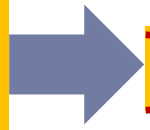
```
drop domain NomeDominio
        [restrict|cascade]
```

L'opzione ***restrict*** (utilizzata di default) specifica che il comando non deve essere eseguito in presenza di oggetti non vuoti; un dominio non viene rimosso se appare in qualche definizione di tabella. L'opzione ***cascade*** restituisce il dominio elementare a tutti gli attributi legati al particolare dominio che si vuole rimuovere, scatenando una *reazione a catena*.

Cancellazione di domini definiti dall'utente

```
CREATE DOMAIN ETAIMPIEGATI AS INTEGER
DEFAULT 31
check (VALUE >= 18 and VALUE <= 80)
)
```

```
CREATE TABLE Impiegato(
ID INT,
Nome VARCHAR(20),
Età ETAIMPIEGATI,
Salario REAL
)
```



~~DROP DOMAIN ETAIMPIEGATI~~

L'opzione ***restrict*** (utilizzata di default) specifica che il comando non deve essere eseguito in presenza di oggetti non vuoti; in questo caso il dominio non viene rimosso dato che appare nella definizione della tabella **Impiegato**.

Cancellazione di domini definiti dall'utente

```
CREATE DOMAIN ETAIMPIEGATI AS INTEGER
DEFAULT 31
check (VALUE >= 18 and VALUE <= 80)
)
```

```
CREATE TABLE Impiegato(
ID INT,
Nome VARCHAR (20),
Età ETAIMPIEGATI,
Salario REAL
)
```

DROP DOMAIN ETAIMPIEGATI **cascade**

Età ~~**ETAIMPIEGATI**~~

Età **INTEGER**

L'opzione ***cascade*** restituisce il dominio elementare a tutti gli attributi legati al particolare dominio che si vuole rimuovere.

Modifica di domini definiti dall'utente

```
CREATE DOMAIN ETAIMPIEGATI AS INTEGER
DEFAULT 31
check (VALUE >= 18 and VALUE <= 80)
)
```

```
CREATE TABLE Impiegato(
ID INT,
Nome VARCHAR (20),
Età ETAIMPIEGATI,
Salario REAL
)
```

```
ALTER DOMAIN ETAIMPIEGATI
DROP DEFAULT
```

Se il valore dell'età di Marco (prima tupla) è stato inserito direttamente, il comando va a buon fine…

altrimenti

se il valore dell'età di Marco (prima tupla) deriva dal *valore di default* del nuovo dominio, il comando fallisce, perché la nuova versione della tabella non è più soddisfatta dai dati presenti.

Impiegato

ID	Nome	Età	Salario
1	Marco	31	100
2	Francesco	24	200

Vincoli di Integrità

- Un ***vincolo di integrità*** descrive le condizioni che ogni istanza legale di una relazione deve soddisfare.
  - *Limita* i dati che possono essere memorizzati in un'istanza della base di dati.
  - Inserimenti/cancellazioni/aggiornamenti che violano i vincoli di integrità non sono permessi.
  - Possono essere usati per garantire la semantica dell'applicazione, o per prevenire inconsistenze.

A. **Vincoli intrarelazionali:** vincoli che coinvolgono una sola relazione (***not null***, ***unique***, ***primary key***, ***default***).
   - Esistono anche i ***vincoli di dominio***, che specificano che i valori dei campi devono essere sempre del tipo corretto.

B. **Vincoli interrelazionali:** vincoli di integrità referenziale (***foreign key***).

Vincoli predefiniti sugli attributi

CREATE TABLE *NomeTabella*(
NomeAttributo Dominio [Vincoli]
....
{ ,*NomeAttributo Dominio* [Vincoli] }
{ ,Vincolo di tabella }
)

- **[NOT NULL | NULL]** - In assenza del vincolo, di default può contenere valori NULL.
- **[DEFAULT valore]** - usato per impostare un valore di default, utile quando in un inserimento il valore dell'attributo non viene specificato. In assenza del vincolo, si suppone come default il valore NULL.
- **[UNIQUE | PRIMARY KEY]** - UNIQUE rappresenta un attributo che non può contenere valori duplicati (una chiave); PRIMARY KEY indica la chiave primaria, che oltre a non ammettere duplicati non può contenere valori NULL.
- **[REFERENCES *Nome_Tabella* [(*Nome_Attributo*)]] -** Permette di definire un vincolo di chiave esterna su *Nome_Attributo* verso la chiave primaria di *Nome_Tabella*.

Vincoli predefiniti sulla tabella

CREATE TABLE *NomeTabella*(
NomeAttributo Dominio [Vincoli]
....
{ ,*NomeAttributo Dominio* [Vincoli] }
{ ,Vincolo di tabella }
)

- **[PRIMARY KEY (nome_attributo1, nome_attributo2,…)] -** Permette di definire una chiave primaria per la tabella sfruttando un certo insieme di attributi.
- **[UNIQUE (nome_attributo1, nome_attributo2,…)] -** Permette di definire una chiave candidata per la tabella sfruttando un certo insieme di attributi.
- **[FOREING KEY (nome_att1,nome_att2,…) REFERENCES nome_tabella [(nome_att1,nome,att2,…)]]** - Permette di definire vincoli di chiave esterna su più attributi.

NOT NULL e DEFAULT

- ***NOT NULL* :** serve per specificare che NULL **non è ammesso** come valore dell'attributo.
  - di **default**, se non si inserisce questo vincolo, l'attributo potrà accettare valori nulli.
- ***DEFAULT* :** usato per impostare un valore di default per un attributo, che sia compatibile con il suo dominio. NULL rappresenta il valore di default di base.

Si crea la tabella persona con i seguenti vincoli:

- ***nome*** non può assumere attributi NULL e non ha attributi di DEFAULT.
- ***cognome*** può assumere valori NULL ed ha un valore di DEFAULT.
- ***età*** non può assumere valori NULL ed ha un valore di DEFAULT.

```
CREATE TABLE Persona(
nome varchar(20) NOT NULL,
cognome varchar(20) DEFAULT 'pippo',
età int NOT NULL DEFAULT 0
)
```

Chiavi Candidate e Primarie

- **Vincoli di chiave**
  - Insieme minimo di campi tale che in ogni istanza legale due tuple distinte siano diverse nei valori di tali campi.
- ***UNIQUE*** rappresenta la definizione di (super)*chiave*, cioè di un insieme di campi che non può contenere valori duplicati.
  - viene fatta un'eccezione per il valore NULL, che può comparire su righe diverse senza violare il vincolo.
- ***PRIMARY KEY*** indica la chiave primaria (scelta tra le *chiavi candidate*) che, oltre a non ammettere duplicati, non può contenere valori NULL.
  - **è FONDAMENTALE definire una e una sola PRIMARY KEY per ogni relazione**.
    - per evitare la possibilità di avere due tuple identiche nella relazione.
    - perché il DBMS può creare un indice con i campi della chiave primaria come campi di ricerca.

Chiavi Candidate e Primarie

Si crea la tabella persona con i seguenti vincoli:

- ***nome*** è una PRIMARY KEY, perciò non ammette duplicati e non può assumere valori NULL.
- ***cognome*** può assumere valori NULL ed ha un valore di DEFAULT.
- ***età*** non ammette duplicati, può assumere valori NULL ed ha un valore di DEFAULT.

```
CREATE TABLE Persona(
nome varchar(20) PRIMARY KEY,
cognome varchar(20) DEFAULT 'pippo',
età int UNIQUE DEFAULT 0
)
```

Chiavi Candidate e Primarie

PRIMARY KEY e **UNIQUE** definite per più attributi.

```
CREATE TABLE NomeTabella(
        NomeAttributo1 Dominio [Vincoli],
        NomeAttributo2 Dominio [Vincoli],
        NomeAttributo3 Dominio [Vincoli],
        NomeAttributo4 Dominio [Vincoli],
PRIMARY KEY(NomeAttributo1, NomeAttributo2),
UNIQUE(NomeAttributo3, NomeAttributo4)
)
```

Chiavi Candidate e Primarie

- ***I DUE COMANDI CREANO LA STESSA TABELLA?***

```
CREATE TABLE Persona(
nome varchar(20) NOT NULL,
cognome varchar(20) NOT NULL,
età int default 0,
UNIQUE(nome,cognome)
)
```

```
CREATE TABLE Persona(
nome varchar(20) NOT NULL UNIQUE,
cognome varchar(20) NOT NULL UNIQUE,
età int default 0
)
```

- ***NO! Perché?***
- ***il primo comando crea una chiave su nome e cognome, impedendo l'inserimento di tuple con valori identici su quei due campi; possiamo però avere tante tuple che hanno il campo nome uguale o il campo cognome uguale.***
- ***il secondo comando crea esattamente una chiave per nome ed una chiave per cognome.***

Vincoli di foreign key

- A volte le informazioni memorizzate in una relazione sono collegate alle informazioni contenute in un'altra relazione. In questo caso, si può utilizzare un vincolo di ***chiave esterna.***
  - Insieme dei campi di una relazione R che viene usato per far riferimento a tuple in un'altra relazione S. I campi dovrebbero essere una chiave (idealmente primaria) di S.

```
CREATE TABLE Esami(
Esame VARCHAR (20),
studID CHAR(4) REFERENCES Studenti
)
```

La chiave esterna nella relazione referenziante **deve essere compatibile con la chiave primaria della relazione referenziata**, cioè deve avere lo stesso numero di colonne e i tipi di dati compatibili.

Relazione referenziante

Esami

Esame	studID
Basi di Dati	2013
Analisi	2345

Relazione referenziata

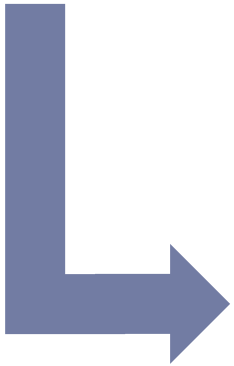
Studenti

ID	Nome	Cognome
2013	Mario	Rossi
2345	Marco	Bianchi
5467	Gianni	Verdi

Vincoli di foreign key

- in alternativa, se gli attributi da referenziare sono multipli, il vincolo esterno può essere inserito tra i *vincoli di tabella* utilizzando la clausola **FOREIGN KEY**.

```
CREATE TABLE Infrazioni(
        Codice CHAR(5) PRIMARY KEY,
        Data Date,
        Vigile VARCHAR (4) ,
        Prov CHAR(2),
        Numero CHAR(6),
FOREIGN KEY (Prov,Numero) REFERENCES Auto(Prov,Numero)
```



Infrazioni

Codice	Data	Vigile	Prov	Numero
34321	1/2/95	3987	MI	39548K
53524	4/3/95	3295	TO	E39548
64521	5/4/96	3295	PR	839548
73321	5/2/98	9345	PR	839548

Auto

Prov	Numero	Cognome	Nome
MI	39548K	Rossi	Mario
TO	E39548	Rossi	Mario
PR	839548	Neri	Luca

Inserimento dei dati in una tabella

- Sintassi :

```
INSERT INTO NomeTabella
        [(attributo_1,...,attributo_n)]
        VALUES (valore 1,...,valore n)
```

Esempio: **cominciamo con il creare una relazione SQL**

```
CREATE TABLE Ricoveri (
Paziente CHAR(4),
Inizio DATE DEFAULT ('0000-00-00') ,
Fine DATE,
Reparto CHAR
)
```

Ricoveri

Paziente	Inizio	Fine	Reparto

Inserimento dei dati in una tabella

- Per ogni colonna deve essere specificato un valore corrispondente del giusto dominio.
- Se non viene inserita nessuna lista di colonne, allora deve essere dato un valore (seguendo l'ordine originale di definizione degli attributi nello schema relazionale) per ogni colonna della relazione.
- ***ESEMPIO*** **:**

```
INSERT INTO Ricoveri(Paziente, Inizio, Fine, Reparto)
        VALUES('A102','2008-06-02','2008-06-05','A')
```

```
INSERT INTO Ricoveri
        VALUES('B444','B')
```

Ricoveri

Paziente : CHAR(4)	Inizio : Date	Fine : Date	Reparto : CHAR
A102	02/05/2008	05/06/2008	A

Inserimento dei dati in una tabella

- Un'inserzione non necessariamente deve seguire l'ordine degli attributi come specificato nella CREATE TABLE
- Se una colonna viene omessa, allora per essa viene assegnato il valore di DEFAULT (o in assenza di questo il valore NULL, se non viola alcun vincolo)
- ***ESEMPIO*** :

```
INSERT INTO Ricoveri(Inizio, Paziente, Fine, Reparto)
       VALUES('2008-08-10','A102','2008-08-13','A')
```

```
INSERT INTO Ricoveri(Fine, Paziente, Reparto)
       VALUES('2008-08-18','A102','A')
```

Ricoveri

Paziente : CHAR(4)	Inizio : Date	Fine : Date	Reparto : CHAR
A102	02/05/2008	05/06/2008	A
A102	10/08/2008	13/08/2008	A
A102	00/00/0000	18/08/2008	A

Valore di DEFAULT per l'attributo *Inizio*

Inserimento dei dati in una tabella

- Se non viene inserita nessuna lista di colonne, allora deve essere dato un valore (seguendo l'ordine originale di definizione degli attributi nello schema relazionale) per ogni colonna della relazione
- ***ESEMPIO*** :

```
INSERT INTO Ricoveri
        VALUES('B444','2008-06-05',NULL,'B')
```

Ricoveri

Paziente : CHAR(4)	Inizio : Date	Fine : Date	Reparto : CHAR
A102	02/05/2008	05/06/2008	A
A102	10/08/2008	13/08/2008	A
A102	00/00/0000	18/08/2008	A
B444	05/06/2008	NULL	B

Inserimento dei dati in una tabella

- ***ATTENZIONE :*** una tabella in SQL è definita come un multinsieme di tuple. Ciò significa che **potranno comparire** due tuple uguali nella tabella....ne segue che una tabella SQL **non è** una relazione.

- ***ESEMPIO :***

```
INSERT INTO Ricoveri
        VALUES('A102','2008-08-10','2008-08-13','A')
```

E' necessario definire vincoli di chiave sulle tabelle per evitare duplicazioni.

Paziente : CHAR(4)	Inizio : Date		
A102	02/05/2008		
A102	10/08/2008	13/08/2008	A
A102	00/00/0000	18/08/2008	A
B444	05/06/2008	NULL	B
A102	10/08/2008	13/08/2008	A

Cancellazione di tuple

- **cancellare una o più tuple da una tabella**

```
DELETE FROM NomeTabella
  [WHERE Condizione]
```

- ***ESEMPIO*** **:**
- **Cancellare tutte le righe della tabella Dipartimento**

```
DELETE FROM Dipartimento
```

Se l'argomento della clausola WHERE non viene specificato, il comando cancella tutte le righe della tabella.

- **Cancellare tutte le righe della tabella Dipartimento relative al settore Produzione**

```
DELETE FROM Dipartimento
        WHERE Settore='Produzione'
```

La clausola WHERE può contenere interrogazioni annidate.

Aggiornamento di tuple

- **Aggiornare una o più tuple di una tabella**

UPDATE *NomeTabella*
SET *Attributo* = *Espressione* {, *Attributo* = *Espressione*}
[**WHERE** *Condizione*]

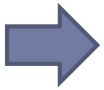
La clausola WHERE viene applicata per prima e determina i campi da modificare. La clausola SET determina i nuovi valori.

- ***ESEMPIO*** **:**
- **Incrementare di 5 lo stipendio dei dipendenti afferenti al settore Amministrazione**

UPDATE Dipendente **SET** Stipendio = Stipendio + 5
WHERE Settore = 'Amministrazione'

Dipendente

ID	Nome	Stipendio	Settore
0	Marco	200	Direzione
1	Paola	300	Amministrazione



ID	Nome	Stipendio	Settore
0	Marco	200	Direzione
1	Paola	**305**	Amministrazione

Garantire l'integrità referenziale

- ***ESEMPIO*** **:** Si vogliono creare le seguenti tabelle in SQL

- **persone(nome, reddito, età, sesso)**
  - nome è una stringa di 20 caratteri (chiave primaria);
  - reddito è un valore reale;
  - età è un intero di 3 cifre;
  - sesso è un carattere.
- **genitori(figlio, genitore)**
  - figlio (stringa di 20 caratteri, chiave esterna su PERSONE); con valore di DEFAULT 'Jack';
  - genitore (stringa di 20 caratteri, chiave esterna su PERSONE);
  - chiave primaria formata da "figlio" e "genitore".

Garantire l'integrità referenziale

- ***ESEMPIO*** :

```
CREATE TABLE Persone(
Nome VARCHAR (20) PRIMARY KEY,
Reddito REAL,
Età NUMERIC(3),
Sesso CHAR
)

CREATE TABLE Genitori(
Figlio VARCHAR (20) DEFAULT 'Jack'  REFERENCES Persone,
Genitore VARCHAR (20) REFERENCES Persone,
PRIMARY KEY (Figlio,Genitore)
)
```

Garantire l'integrità referenziale

- *Cosa dovremmo fare se una riga di **Persone** viene cancellata (aggiornata)? Le opzioni sono:*

1. Cancellare (aggiornare) tutte le righe di ***Genitori*** che referenziano quella riga di ***Persone***.
2. Non permettere la cancellazione (aggiornamento) della riga di ***Persone***, se essa è referenziata (cioè, legata da un vincolo di *foreign key*) da una riga di ***Genitori***.
3. Per ogni riga di ***Genitori*** referenziata dalla riga cancellata (aggiornata) di ***Persone,*** impostare i valori dell'attributo ***figlio*** *e* dell'attributo ***genitore*** ai loro valori *di default*.
   - Questa opzione è in conflitto col fatto che il valore di default per ***genitore*** sarebbe NULL, ma dato che ***genitore*** è parte della chiave primaria di **Genitori**, **non può essere impostato a NULL**.
4. Per ogni riga di ***Genitori*** eliminata (aggiornata), impostare i corrispondenti valori dell'attributo ***figlio*** e dell'attributo ***genitore*** referenzianti a **NULL**.
   - Nell'esempio questa opzione è in conflitto col fatto che ***figlio*** e ***genitore*** sono parte della chiave primaria di ***Genitori*** e quindi **non possono essere impostati a NULL**.

SQL permette di scegliere una qualunque delle quattro opzioni per DELETE e UPDATE

Garantire l'integrità r

```
CREATE TABLE NomeTabella(
     NomeAttributo Dominio [Vincoli]
{, NomeAttributo Dominio [Vincoli] }
[
          ……
FOREIGN KEY (NomeAttr1)
  REFERENCES tabella_referenziata(A1)
  [ON DELETE [CASCADE|SET DEFAULT|
                          SET NULL|NO ACTION]
   ON UPDATE [CASCADE|SET DEFAULT|
                          SET NULL|NO ACTION]
  ]
)
```

SET DEFAULT : all'attributo referenziante viene assegnato il valore di DEFAULT al posto del valore modificato nella tabella referenziata.
CASCADE : aggiorna i valori nella tabella referenziante partendo dal valore della tabella referenziata.
SET NULL : all'attributo referenziante viene assegnato il valore NULL al posto del valore modificato nella tabella referenziata.
NO ACTION : la modifica \cancellazione non viene consentita se vìola i vincoli di foreign key.

In assenza di comportamenti specifici, **"NO ACTION"** è il valore di default.

Garantire l'integrità referenziale

- ***ESEMPIO*** **:**

```
CREATE TABLE Genitori(
Figlio VARCHAR (20) DEFAULT 'Jack' REFERENCES Persone
ON UPDATE CASCADE
ON DELETE SET DEFAULT,
Genitore VARCHAR (20) REFERENCES Persone
PRIMARY KEY (Figlio,Genitore))
```

Persone

Nome	Reddito	Età	Sesso
Jack	15	80	M
Carlo	25	24	M
Giuseppe	30	45	M
Maria	76	43	F
Gianni	60	50	M
Francesca	18	26	F

Genitori

Figlio	Genitore
Carlo	Gianni
Carlo	Maria
Francesca	Giuseppe

ESEMPIO **:**

DELETE FROM Persone where Sesso = 'F'

Persone

Nome	Reddito	Età	Sesso
Jack	15	80	M
Carlo	25	24	M
Giuseppe	30	45	M
Maria	76	43	F
Gianni	60	50	M
Francesca	18	26	F

Nome	Reddito	Età	Sesso
Jack	15	80	M
Carlo	25	24	M
Giuseppe	30	45	M
Maria	76	43	F
Gianni	60	50	M

Figlio :
ON UPDATE CASCADE
ON DELETE SET DEFAULT

Genitore :
ON UPDATE NO ACTION
ON DELETE NO ACTION

Genitori

Figlio	Genitore
Carlo	Gianni
Carlo	Maria
Jack	Giuseppe

La tupla contenente il valore 'Maria' **non può essere cancellata**, in quanto violerebbe i vincoli di foreign key per l'attributo Genitore, la cui politica di cancellazione è **NO ACTION**.

La tupla contenente il valore 'Francesca' **viene cancellata**, e il corrispondente valore dell'attributo Figlio viene settato a **DEFAULT**. N.B. Questo è possibile poiché il valore Jack è presente nella colonna Nome di Persone.

***ESEMPIO* :**

UPDATE Persone **SET** Nome = Andrea
WHERE Età=24

Persone

Nome	Reddito	Età	Sesso
Jack	15	80	M
Carlo	25	24	M
Giuseppe	30	45	M
Maria	76	43	F
Gianni	60	50	M

Nome	Reddito	Età	Sesso
Jack	15	80	M
Andrea	25	24	M
Giuseppe	30	45	M
Maria	76	43	F
Gianni	60	50	M

***Figlio* :**
ON UPDATE CASCADE
ON DELETE SET DEFAULT

***Genitore* :**
ON UPDATE NO ACTION
ON DELETE NO ACTION

Genitori

Figlio	Genitore
Andrea	Gianni
Andrea	Maria
Jack	Giuseppe

ATTENZIONE: Se ci fossero state più persone con Età=24, **l'aggiornamento non sarebbe andato a buon fine**, in quanto sarebbe stato violato il vincolo di chiave primaria sull'attributo Nome.

La tupla contenente il valore 'Carlo' può essere aggiornata al valore 'Andrea', in quanto :
-non viola il vincolo di foreign key per l'attributo Genitore;
- viola il vincolo di foreign key per l'attributo Figlio, la cui politica di aggiornamento però è **CASCADE**.

Creazione/Cancellazione di schemi

- **creazione di un nuovo database**

  create database *NomeDatabase*

- **cancellazione di un database**

  drop database *NomeDatabase* **[restrict|cascade]**

- **cancellazione di una tabella**

  drop table *NomeTabella* **[restrict|cascade]**

- L'opzione ***restrict*** (di default) specifica che il comando non deve essere eseguito in presenza di oggetti non vuoti. Uno schema non viene rimosso se contiene tabelle o altri oggetti. Una tabella non viene rimossa se possiede delle righe o se è presente in qualche definizione di tabella o vista.
- Con l'opzione ***cascade*** tutti gli oggetti specificati devono essere rimossi.

DROP TABLE Persone **cascade**

Distrugge la relazione Persone. Le informazioni sullo schema e le tuple vengono cancellate.

Modifica di tabelle

- **aggiungere colonne ad una tabella**

```
ALTER TABLE NomeTabella
        ADD COLUMN NomeColonna
                                Dominio [Vincoli]
```

- **eliminare colonne da una tabella**

```
ALTER TABLE NomeTabella
        DROP NomeColonna
```

Modifica di tabelle

- **aggiungere una chiave ad una tabella**

```
ALTER TABLE NomeTabella
  ADD [CONSTRAINT [Nome_Vincolo]]
   PRIMARY KEY|UNIQUE
        (NomeColonna1,...,NomeColonnaN)
```

- **eliminare una chiave primaria da una tabella**

```
ALTER TABLE NomeTabella
  DROP PRIMARY KEY
```

- **aggiungere una chiave esterna ad una tabella**

```
ALTER TABLE TabellaReferenziante
 ADD [CONSTRAINT [Nome_Vincolo]]
  FOREIGN KEY(AttributiReferenzianti)
REFERENCES TabellaReferenziata(AttributiReferenziati)
```

Modifica di tabelle

- **aggiungere\eliminare un valore di default da una tabella**

```
ALTER TABLE NomeTabella
  ALTER COLUMN NomeAttributo
    <SET DEFAULT Valore di default | DROP DEFAULT>
```

- **eliminare un vincolo unique da una tabella**

```
ALTER TABLE NomeTabella
  DROP UNIQUE (NomeColonna1,...,NomeColonnaN)
```

Modifica di tabelle

- ***ESEMPIO*** **:**

```
CREATE TABLE Genitori(
Figlio VARCHAR (20),
Genitore VARCHAR (20)
)
```

- **Si aggiunga una PRIMARY KEY (Figlio, Genitore) alla relazione Genitori.**

```
ALTER TABLE Genitori
        ADD PRIMARY KEY(Figlio,Genitore)
```

- **Si aggiunga un vincolo di FOREIGN KEY all'attributo Figlio della relazione Genitori. Figlio referenzia l'attributo Nome, chiave primaria della relazione Persone.**

```
ALTER TABLE Genitori
  ADD FOREIGN KEY(Figlio)
      REFERENCES Persone(Nome)
```