Corso di Laurea in Ingegneria Gestionale
SAPIENZA Università di Roma
Esercitazioni del corso di Basi di Dati
Prof.ssa Catarci e Prof.ssa Scannapieco

Anno Accademico 2011/2012

5 – SQL : Definizione e manipolazione dei dati

Francesco Leotta

Ultimo aggiornamento : 25/04/2012

SQL : le varie versioni nel tempo

Nome informale	Nome ufficiale	Caratteristiche
SQL base	SQL-86	Costrutti base
	SQL-89	Integrità referenziale
SQL-2	SQL-92	Modello relazionale Vari costrutti nuovi 3 livelli: entry, intermediate, full
SQL-3	SQL:1999	Modello relazionale a oggetti Organizzato in diverse parti Trigger, funzioni esterne, ...
	SQL:2003	Estensioni del modello a oggetti Eliminazione di costrutti non usati Nuove parti: SQL/JRT, SQL/XML, ...
	SQL:2006	Estensione della parte XML
	SQL:2008	Lievi aggiunte (per esempio, trigger `instead of`)

Possiede solo le primitive per il controllo delle interrogazioni.

Supporto limitato per la definizione e manipolazione degli schemi e delle istanze.

Contiene tutte le carattistiche di base dell'SQL.

Introduce molte nuove funzionalità ed è pienamente compatibile con SQL-2.

SQL-92

- è un linguaggio ricco e complesso.
- ancora nessun sistema commerciale mette a disposizione tutte le funzionalità del linguaggio.
- 3 livelli di aderenza allo standard (i sistemi possono essere caratterizzati in base al modello a cui aderiscono) :
  - **Entry SQL:** abbastanza simile a SQL-89.
  - **Intermediate SQL:** caratteristiche più importanti per le esigenze del mercato; supportato dai DBMS commerciali.
  - **Full SQL:** rappresenta lo standard nella sua interezza, con funzioni avanzate in via di inclusione nei sistemi.
- i sistemi offrono anche funzionalità non standard, offrendo piccole differenze nell'implementazione del linguaggio SQL :
  - rischio di incompatibilità tra sistemi.
  - rischio di incompatibilità con i nuovi standard (es. trigger in SQL:1999).

SQL : Structured Query Language

- SQL non è un semplice linguaggio per le interrogazioni...
- ...ma contiene 3 sotto-linguaggi :
  - **DDL (D**ata **D**efinition **L**anguage**) :** 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 (D**ata **M**anipulation **L**anguage**) :** 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 (D**ata **C**ontrol **L**anguage**) :** permette di gestire gli utenti ed i permessi.

SQL : Structured Query Language

- Un DBMS basato su SQL consente di gestire basi di dati relazionali; dal punto di vista sistemistico è un **server**.
- Quando ci si connette ad un DBMS basato su SQL, si deve indicare, implicitamente o esplicitamente, su quale basi di dati si vuole operare.
- Se si vuole operare su una base di dati non ancora esistente, si utilizzerà un meccanismo messo a disposizione dal server per la sua creazione.
- Coerentemente con la filosofia del modello relazionale, una base di dati in SQL è caratterizzata dallo schema (livello intensionale) e da una istanza (quella corrente – livello estensionale).
  - In più, una base di dati SQL è caratterizzata da un insieme di meta-dati (il *catalogo* – vedi dopo).

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 } }
          { ,Vincoli 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   character (6)    primary key,
Nome        varchar(20)      not null,
Dipart      varchar(15),
Stipendio   numeric(9)       default 0,
Salario     real,
foreign key(Dipart) references Dipartimento(NomeDip),
unique(Cognome Nome)
)
```

ATTENZIONE

- **Attenzione:** una tabella in SQL è definita come un multiinsieme 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.**

- 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 consigliabile definire almeno una primary key per ogni tabella SQL**.

Domini

- **Domini elementari** (predefiniti)
  - **Carattere**: singoli caratteri o stringhe, anche di lunghezza variabile
  - **Bit**: singoli bit o stringhe di bit
  - **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*
      - *per CHAR in MySQL sono ammessi valori di n che variano da 0 a 255. Memorizza esattamente n byte*
      - *per VARCHAR in MySQL sono ammessi valori di n che variano da 0 a 65535. Memorizza 1 byte di prefisso che specifica la lunghezza del dato (se n>255, i byte di prefisso saranno 2) e 1 byte per ogni carattere inserito*
    - 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**

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

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

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

Possibile rappresentazione dei domini numerici in un calcolatore.

Tipo	Bytes	Valore minimo	Valore massimo
		(Signed/Unsigned)	(Signed/Unsigned)
SMALLINT	2	-32768	32767
		0	65535
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

Introdotto in SQL-3.

SQL : Domini Predefiniti

- **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 (*in MySQL* da 0 a 65).
    - **s** rappresenta la *scala, ovvero il numero di cifre che devono comparire dopo la virgola* (*in MySQL* da 0 a 30, 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***
  - **bit, bit(n) :** sequenza fissa di n bit (valori appartenenti all'insieme <0,1>) :
    - ***Esempio:***
      - ***bit(2) : <(00),(01),(10),(11)>***

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

Rimosso da SQL:2003

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"**: i suoi limiti teorici vanno da -3.402823466^(+38) a -1.175494351^(-38) e da 1.175494351^(-38) a 3.402823466^(+38), oltre allo zero
  - **I valori DOUBLE sono invece a "precisione doppia"**: i limiti teorici sono da -1.7976931348623157^(+308) a -2.2250738585072014(^-308) e da 2.2250738585072014^(-308) a 1.7976931348623157^(+308), oltre allo zero
  - Per entrambe i tipi di dato i limiti reali dipendono dall'hardware e dal sistema operativo
  - Se *m* e *d* non sono indicati i valori possono essere memorizzati fino ai limiti effettivi
  - Per questi dati 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)***

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 [*with time zone*] :**
    - contiene un valore di tempo ('*ore:minuti:secondi*') nel formato ***('hh:mm:ss')***.
  - **timestamp [*with time zone*] :**
    - 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

time (o timestamp) with time zone

- se l'opzione *with time zone* è specificata, allora risulta possibile accedere ai campi *timezone_hour* e *timezone_minute*
- rappresentano la differenza di fuso orario tra *l'ora locale* e *l'ora universale* (UCT – Coordinated Universal Time)
  - Perciò, **21:03:04+1:00** e **20:03:04+0:00** corrispondono allo stesso istante temporale, il primo rappresentato nell'ora solare italiana (differenza con il fuso orario di 1:00), il secondo nell'ora universale.

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 rispettata 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

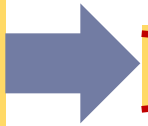
drop domain *NomeDominio*
[restrict|cascade]

L'opzione ***restrict*** (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
)
```



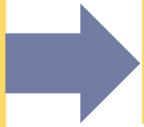
~~DROP DOMAIN ETAIMPIEGATI~~

L'opzione ***restrict*** (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
```

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 condizioni che ogni istanza legale di una relazione deve soddisfare.
  - *limita* i dati che possono essere memorizzati in un istanza della base di dati.

A. **Vincoli intrarelazionali:** vincoli che coinvolgono una sola relazione (***not null***, ***unique***, ***primary key***, ***default***).
B. **Vincoli interrelazionali:** vincoli di integrità referenziale (***foreign key***).

Vincoli predefiniti sugli attributi

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

[NOT NULL | NULL] - 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 [(attributo)]] - Permette di definire un vincolo di chiave esterna su *NomeAttributo* verso la chiave primaria di *nome_tabella*.

Vincoli predefiniti sulla tabella

CREATE TABLE *NomeTabella*(
NomeAttributo Dominio {Vincoli }
....
{ ,*NomeAttributo Dominio* {Vincoli } }
{ ,Vincoli 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 per la tabella sfruttando un certo insieme di attributi.
[FOREING KEY (nome_att1,nome,att2,…)
REFERENCES nome_tab[(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

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',
eta int NOT NULL DEFAULT 0
)
```

Chiavi Candidate e Primarie

- ***Vincoli di chiave***
  - è l'imposizione che un certo *sottoinsieme minimale* dei campi di una relazione sia un identificatore unico per una tupla
- ***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 attributi 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',
eta 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,
eta int default 0,
UNIQUE(nome,cognome)
)

CREATE TABLE Persona(
nome varchar(20) not null **UNIQUE**,
cognome varchar(20) not null **UNIQUE**,
eta 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

```
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

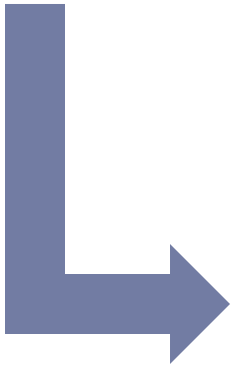
Studenti

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 può essere inserito tra i *[vincoli di tabella]*

```
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 )
```



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
        [(colonna 1,…,colonna n)]
        VALUES (valore 1,…,valore n)
```

Esempio

```
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

- Una 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

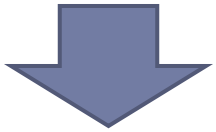
Paziente : CHAR(4)	Inizio : Date	Fine : Date	Reparto : CHAR
A102	02/05/2008	05/06/2008	A
A102	10/08/2008	13/08/2008	A
A102	00/00/0000	18/08/2008	A
B444	05/06/2008	NULL	B

Inserimento dei dati in una tabella

- ***ATTENZIONE :*** una tabella in SQL è definita come un multi-insieme 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		A
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'
```

Aggiornamento di tuple

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

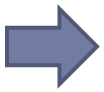
UPDATE *NomeTabella*
SET *Attributo = Espressione* {, *Attributo = Espressione*}
[**WHERE** *Condizione*]

- ***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, eta, 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 'Gianni'
  - 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,
Eta NUMERIC(3),
Sesso CHAR
);

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

Garantire l'integrità referenziale

- ***Vincoli di 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*** al ***nome*** di qualche ***persona*** (esistente) di default
- 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 è parte della chiave primaria di Genitori e quindi non può essere impostata a NULL

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

Garantire l'integrità

- ***Vincoli di integrità referenz***

```
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 'Gianni' REFERENCES Persone
ON UPDATE CASCADE
ON DELETE SET DEFAULT,
Genitore VARCHAR (20) REFERENCES Persone
PRIMARY KEY (Figlio,Genitore))
```

Persone

Nome	Reddito	Eta	Sesso
Mario	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	Eta	Sesso
Mario	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	Eta	Sesso
Mario	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
Gianni	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

***ESEMPIO* :**

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

Figlio :
ON UPDATE CASCADE
ON DELETE SET DEFAULT

Genitore :
ON UPDATE NO ACTION
ON DELETE NO ACTION

Persone

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

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

Genitori

Figlio	Genitore
Andrea	Gianni
Andrea	Maria
Gianni	Giuseppe

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**

Cancellazione di schemi

- **creazione di un nuovo database** (alternativo a **create schema**)

```
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.

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

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

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

- **Si aggiunga la colonna Età (si vuole anche che sia chiave) alla relazione Genitori**

```
ALTER TABLE Genitori
        ADD COLUMN Età NUMERIC(3) PRIMARY KEY
```

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 INDEX NomeVincolo
```

Modifica di tabelle

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

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

- **Si aggiunga una PRIMARY KEY alla coppia Figlio-Genitore**

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

- **Si aggiunga una CHIAVE ESTERNA all' attributo Figlio referenziato dall' attributo Nome contenuto nella relazione Persone**

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

Esercizio 1\6

- Dare le definizioni SQL delle tabelle

  AUTORE (Nome, Cognome, DataNascita, Nazionalità)

  LIBRO (TitoloLibro, NomeAutore, CognomeAutore, Lingua)

  inserendo un vincolo di chiave esterna tra (NomeAutore,CognomeAutore) relativi a Libro e (Nome,Cognome) relativi alla relazione Autore
- Specificare una politica di
  - **CASCADE** sulla cancellazione
  - **SET NULL** sulle modifiche
- Successivamente, dato lo schema della relazione, spiegare cosa può capitare con l'esecuzione dei seguenti comandi di aggiornamento:
  - delete from AUTORE where Cognome = 'Rossi'
  - update LIBRO set Nome= 'Umberto' where Cognome = 'Eco'
  - insert into AUTORE(Nome,Cognome) values('Antonio','Bianchi')
  - update AUTORE set Nome = 'Italo' where Cognome = 'Calvino'

Esercizio 2\6

- Vincolo di chiave esterna fra *(NomeAutore,CognomeAutore)* e *(Nome Cognome)*
- politica di
  CASCADE sulle cancellazioni
  SET NULL sulle modifiche

Autore

Nome	Cognome	DataNascita	Nazionalità

Libro

TitoloLibro	NomeAutore	CognomeAutore	Lingua

```
CREATE TABLE Autore (
Nome VARCHAR(20),
Cognome VARCHAR(20),
DataNascita DATE,
Nazionalità VARCHAR(20),
PRIMARY KEY(Nome, Cognome)
)
```

```
CREATE TABLE Libro (
TitoloLibro VARCHAR(30) PRIMARY KEY,
NomeAutore  VARCHAR(20),
CognomeAutore VARCHAR(20),
Lingua VARCHAR(20),
FOREIGN KEY (NomeAutore, CognomeAutore)
        REFERENCES Autore (Nome, Cognome)
ON DELETE CASCADE
ON UPDATE SET NULL
)
```

Esercizio 3\6

- Vincolo di chiave esterna fra *(NomeAutore,CognomeAutore)* e *(Nome Cognome)*
- politica di
  CASCADE sulle cancellazioni
  SET NULL sulle modifiche

Autore

Nome	Cognome	DataNascita	Nazionalità

Libro

TitoloLibro	NomeAutore	CognomeAutore	Lingua

DELETE FROM Autore where Cognome = 'Rossi'

Il comando DELETE cancella dalla tabella ***Autore*** tutte le tuple con Cognome = 'Rossi'. A causa della politica **CASCADE** anche le tuple di ***Libro*** con CognomeAutore = 'Rossi' verranno eliminate

Esercizio 4\6

- Vincolo di chiave esterna fra *(NomeAutore,CognomeAutore)* e *(Nome Cognome)*
- politica di
  CASCADE sulle cancellazioni
  SET NULL sulle modifiche

Autore

Nome	Cognome	DataNascita	Nazionalità

Libro

TitoloLibro	NomeAutore	CognomeAutore	Lingua

```
UPDATE Libro set Nome = 'Umberto'
WHERE Cognome = 'Eco'
```

Il comando **non è corretto**: *Nome* e *Cognome* sono attributi della tabella ***Autore*** e non della tabella ***Libro***

Esercizio 5\6

- Vincolo di chiave esterna fra *(NomeAutore,CognomeAutore)* e *(Nome Cognome)*
- politica di
  CASCADE sulle cancellazioni
  SET NULL sulle modifiche

Autore

Nome	Cognome	DataNascita	Nazionalità

Libro

TitoloLibro	NomeAutore	CognomeAutore	Lingua

INSERT INTO Autore(Nome,Cognome)
VALUES('Antonio','Bianchi')

Il comando aggiunge una nuova tupla alla tabella ***Autore***. Non ha alcun effetto sulla tabella ***Libro***

Esercizio 6\6

- Vincolo di chiave esterna fra *(NomeAutore,CognomeAutore)* e *(Nome Cognome)*
- politica di
  CASCADE sulle cancellazioni
  SET NULL sulle modifiche

Autore

Nome	Cognome	DataNascita	Nazionalità

Libro

TitoloLibro	NomeAutore	CognomeAutore	Lingua

UPDATE Autore **SET** Nome = 'Italo' **WHERE** Cognome = 'Calvino'

Le tuple di ***Autore*** con Cognome = ‘Calvino’ vengono aggiornate a Nome = ‘Italo’. A causa della politica **SET NULL** gli attributi *NomeAutore* e *CognomeAutore* delle tuple di ***Libro*** con CognomeAutore = ‘Calvino’ vengono posti a NULL

Esercizio per casa

CREATE DOMAIN DOMINIO **AS** INTEGER **DEFAULT** 10

CREATE TABLE Tabella (
Attributo DOMINIO default 5,
)

- indicare cosa avviene in seguito ai comandi:
  1. alter table Tabella alter column Attributo drop default
  2. alter domain Dominio drop default
  3. drop domain Dominio