UNIVERSITÀ DEGLI STUDI DI SALERNO

Atzeni, Ceri, Fraternali, Paraboschi, Torlone
Basi di dati *Quinta edizione*
McGraw-Hill Education, 2018

1

BASI DI DATI

Concetti Base SQL-DDL - terza parte

DIEM
DIPARTIMENTO DI
INGEGNERIA DELL'INFORMAZIONE ED ELETTRICA E MATEMATICA APPLICATA

Matteo Gaeta
Full Professor – Senior Member IEEE

BASI DI DATI

2

Concetti base – SQL

COMANDI FONDAMENTALI

TABLE ⇔ Tabella ⇔ Relazione
ROW ⇔ Riga ⇔ Tupla
COLUMN ⇔ Colonna ⇔ Attributo

➢ **SQL come DDL ha tre comandi fondamentali:**

Create
Database, Schema, Table, Domain, Constraint, ….

Alter
Database, Schema, Table, Domain, Constraint, ….

Drop
Database, Schema, Table, Domain, Constraint, ….

La cosa più importante di tutta la vita è la scelta di un lavoro, ed è affidata al caso.(Blaise Pascal)

BASI DI DATI

Concetti base – SQL

- Istruzione **CREATE TABLE**:
  - ✓ **definisce uno schema di relazione e ne crea un'istanza vuota**
  - ✓ **specifica attributi, domini e vincoli**

- ✓ In SQL una tabella è definita come un **multinsieme di tuple**, dove il multinsieme è una generalizzazione del concetto di insieme. Esso è in pratica un elenco in cui non è rilevante l'ordine in cui compaiono gli elementi e che (a differenza dell'insieme) ammette elementi ripetuti.

- ✓ In particolare, se una tabella non ha un insieme di attributi definiti chiave (o primary key), allora in essa potranno comparire due tuple uguali.

- ✓ Si noti che una Tabella SQL non è in generale una relazione matematica

Ieri ero intelligente e volevo cambiare il mondo, oggi sono saggio e sto cambiando me stesso (Gialal al-Din Rumi)

BASI DI DATI

Concetti base – SQL

- Quando la tabella ha un insieme di attributi definiti come chiave (o primary key), allora è evidente che non potranno mai comparire due tuple uguali, ciò rende indispensabile per ogni tabella definire almeno una primary key.

- I nomi della tabelle in SQL possono essere formati da lettere e numeri, ma **il primo carattere deve essere sempre una lettera**.

- **Una tabella** in un database relazionale è molto simile a una tabella su carta: **è composta da righe e colonne**.

- Il numero e l'ordine delle colonne sono fissi e ogni colonna ha un nome.

- Il numero di righe è variabile: riflette la quantità di dati archiviati in un determinato momento. Subito **Dopo aver creato una tabella risulta essere vuota**

Non abbattere mai una palizzata prima di conoscere le ragioni per cui fu costruita.
(Gilbert Chesterton)

BASI DI DATI

Concetti base – SQL

- SQL non fornisce alcuna garanzia circa l'ordine delle righe in una tabella.
- Quando una tabella viene letta, le righe appariranno in un ordine non specificato, a meno che l'ordinamento non sia esplicitamente richiesto.
- Inoltre, SQL non assegna identificatori univoci alle righe della tabella, quindi è possibile avere più righe completamente identiche in una tabella.
- Questa è una conseguenza del modello matematico che sta alla base di SQL ma dal punto di vista informativo di solito non è desiderabile.
- Individuare una chiave in una tabella è una soluzione al problema?

BASI DI DATI

Concetti base – SQL

- Ogni colonna ha un tipo di dati.
- Il tipo di dati vincola la serie di possibili valori che possono essere assegnati a una colonna e assegna la semantica ai dati archiviati nella colonna in modo che possa essere utilizzata per i calcoli.
- Ad es., una colonna di tipo numerico non accetterà stringhe di testo arbitrarie e i dati memorizzati in tale colonna possono essere utilizzati per calcoli matematici.
- Al contrario, una colonna di tipo stringa di caratteri accetterà quasi tutti i tipi di dati ma non si presta a calcoli matematici, sebbene siano disponibili altre operazioni come la concatenazione di stringhe.
- PostgreSQL include una serie di tipi di dati incorporati che si adattano a molte applicazioni. Gli utenti possono anche definire i propri tipi di dati.

BASI DI DATI

Concetti base – SQL

CREATE TABLE

- La maggior parte dei tipi di dati incorporati ha nomi e semantiche ovvie.
- Alcuni dei tipi di dati usati di frequente sono:
  - ✓ interi per numeri interi;
  - ✓ numerici per numeri possibilmente frazionari;
  - ✓ testo per stringhe di caratteri;
  - ✓ data per date;
  - ✓ ora per i valori dell'ora;
  - ✓ data e ora per i valori che contengono sia la data che l'ora.
- Per creare una tabella, si utilizza il comando **CREATE TABLE** dal nome appropriato.
- Nella creazione si specifica almeno un nome per la nuova tabella, i nomi delle colonne e il tipo di dati di ciascuna colonna.

BASI DI DATI

Concetti base – SQL

CREATE TABLE

- **CREATE TABLE**

  CREATE TABLE — create a new table
- **Synopsis**

```
CREATE TABLE Nome Tabella (
    NomeAttributo1 tipoDati(Dominio) [valore di default] [vincoli],
    { NomeAttributo1 tipoDati(Dominio) [valore di default] [vincoli] }
    AltriVincoli (es. di Tabella)
    );
```

- ***Esempio***

```
CREATE TABLE my_first_table (
first_column text,
second_column integer
);
```

BASI DI DATI

Concetti base – SQL

- Quando si creano molte tabelle correlate, è consigliabile scegliere un modello di denominazione coerente per le tabelle e le colonne. Ad es., è possibile scegliere di utilizzare nomi singolari.
- Gli RDBMS hanno un limite al numero di colonne che una tabella può contenere. A seconda dei tipi di colonna, tale numero è compreso tra 250 e 1600.
- Definire una tabella con un numero qualsiasi di colonne è molto insolito e spesso un progetto discutibile.
- Se non hai più bisogno di una tabella, puoi rimuoverla utilizzando il comando **DROP TABLE**. Ad es: **DROP TABLE my_first_table;**

BASI DI DATI

Concetti base – SQL

- Il tentativo di eliminare una tabella che non esiste è un errore.
- Tuttavia, è comune nei file di script SQL tentare incondizionatamente di eliminare ogni tabella prima di crearla, ignorando eventuali messaggi di errore, in modo che lo script funzioni indipendentemente dal fatto che la tabella esista o meno.
- Se lo desideri, puoi usare la variante **DROP TABLE IF EXISTS** per evitare i messaggi di errore, ma questo non è SQL standard.
- Se hai bisogno di modificare una tabella che già esiste, il comando è ALTER che vedremo in seguito

BASI DI DATI

Concetti base – SQL

DEFAULT VALUES

- A una colonna (un attributo della relazione) può essere assegnato un **valore predefinito**, detto anche di **Default** (talvolta per esigenze applicative).
- Quando viene creata una nuova riga e non vengono specificati valori per alcune delle colonne, tali colonne verranno riempite con i rispettivi valori predefiniti.
- Se nessun valore predefinito viene dichiarato esplicitamente, il valore predefinito è il valore **NULL**. Questo di solito ha senso perché un valore nullo può essere considerato per rappresentare dati sconosciuti.
- In una definizione di tabella, i valori predefiniti sono elencati dopo il tipo di dati della colonna. Per esempio:

```
CREATE TABLE products (
product_no integer,
name text,
price numeric DEFAULT 9.99
);
```

BASI DI DATI

Concetti base – SQL

DEFAULT VALUES

- Il valore predefinito può essere un'espressione, che verrà valutata ogni volta che viene inserito il valore predefinito (non all'atto di creazione della tabella).
- Un esempio comune è che una colonna timestamp abbia un valore predefinito CURRENT_TIMESTAMP, in modo che venga impostata sull'ora di inserimento della riga.
- Un altro esempio comune è la generazione di un "numero di serie" per ogni riga. In PostgreSQL questo è tipicamente fatto da istruzioni del tipo:

```
CREATE TABLE products (product_no integer DEFAULT
nextval('products_product_no_seq'),
...
);
```

dove la funzione **nextval()** fornisce valori successivi da un oggetto sequenza

BASI DI DATI

Concetti base – SQL

DEFAULT VALUES

- Generare un numero di serie è una esigenza sufficientemente comune. PostgreSQL fornisce una soluzione, ovvero definire l'attributo contenente il numero di serie come SERIAL:

```
CREATE TABLE products (
product_no SERIAL,
...
);
```

- ***In sintesi DEFAULT è usato per impostare un valore di default per un attributo, che ovviamente sia compatibile con il dominio dell'attributo. NULL rappresenta il valore di default di base. Quindi se non desideriamo automatismi che inseriscano valori NULL, dovremo aver cura di esplicitarlo (NOT NULL)***

BASI DI DATI

Concetti base – SQL

CONSTRAINTS - VINCOLI

- I tipi di dati (Data Types) sono un modo per limitare il tipo di dati che possono essere memorizzati in una tabella.
- Per molte applicazioni, tuttavia, tale modalità è troppo grossolana. Ad esempio, una colonna contenente il prezzo di un prodotto dovrebbe probabilmente accettare solo valori positivi. Ma non esiste un tipo di dati standard che accetta solo numeri positivi; oppure potrei dover limitare i dati di una colonna rispetto ad altre colonne o righe.
- A tal fine, SQL consente di definire vincoli su colonne e tabelle.
- I vincoli forniscono tutto il controllo che desideri sui dati nelle tabelle. Se un utente tenta di memorizzare i dati in una colonna che violerebbe un vincolo, viene generato un errore. Ciò si applica anche se il valore proviene dalla definizione del valore predefinito.

BASI DI DATI

Concetti base – SQL

CHECK CONSTRAINTS

- Un vincolo di controllo è il tipo di vincolo più generico. Esso consente di specificare che il valore in una determinata colonna deve soddisfare un'espressione booleana (valore di verità).
  - ✓ Ad esempio, per richiedere prezzi dei prodotti positivi, potresti utilizzare:

```
CREATE TABLE products (
product_no integer,
name text,
price numeric CHECK (price > 0)
);
```

- La definizione del vincolo viene dopo il tipo di dati, come per il DEFAULT. I valori predefiniti e i vincoli possono essere elencati in qualsiasi ordine.

- Un vincolo check è costituito dalla parola chiave **CHECK** seguita da un'espressione tra parentesi.

BASI DI DATI

Concetti base – SQL

SCRIPT

Query Editor Query History

```
1  /* Script Creazione di una Tabella con un Vincolo di Colonna */
2  DROP TABLE IF EXISTS PRODUCTS;
3  Create Table PRODUCTS(
4  Product_no Integer,
5  name text,
6  price numeric CHECK (price >0));
7
8  INSERT INTO PRODUCTS (Product_no, name, price)
9  VALUES (001, 'Tennent''s', '1' );
10
11 Select *
12 FROM PRODUCTS;
```

Data Output Explain Messages Notifications

	product_no integer	name text	price numeric
1	1	Tennent's	1

Query Editor Query History Scratch Pad

```
8  INSERT INTO PRODUCTS (Product_no, name, price)
9  VALUES (001, 'Tennent''s', '1' );
10
11 Select *
12 FROM PRODUCTS;
13
14 /* Script Inserimento di una riga che viola il vincolo */
15 INSERT INTO PRODUCTS (Product_no, name, price)
16 VALUES (002, 'Tennent''s', '-1' );
17
18
19
```

Data Output Explain Messages Notifications

```
ERROR: ERRORE:  la nuova riga per la relazione "products" viola il vincolo di controllo "products_price_check"
DETAIL:  La riga in errore contiene (2, Tennent's, -1).

SQL state: 23514
```

BASI DI DATI

Concetti base – SQL

CHECK CONSTRAINTS

- L'espressione del vincolo di controllo dovrebbe coinvolgere la colonna così vincolata, altrimenti il vincolo non avrebbe molto senso.

- E' possibile dare al vincolo un nome separato. Ciò chiarisce i messaggi di errore e consente di fare riferimento al vincolo quando è necessario modificarlo.

- Nel seguito un esempio che mostra tale sintassi è:

```
CREATE TABLE products (
product_no integer,
name text,
price numeric CONSTRAINT positive_price CHECK (price > 0)
);
```

- Per specificare e denominare un vincolo, occorre utilizzare la parola chiave CONSTRAINT seguita da un identificatore seguito dalla definizione del vincolo. Se non specifichi un nome al vincolo, il sistema ne assegna uno al nostro posto.

BASI DI DATI

Concetti base – SQL

CHECK CONSTRAINTS

- Un vincolo di controllo può anche fare riferimento a più colonne.
  - ✓ Supponiamo che tu memorizzi un prezzo normale e un prezzo scontato e desideri assicurarti che il prezzo scontato sia inferiore al prezzo normale:

```
CREATE TABLE products (
product_no integer,
name text,
price numeric CHECK (price > 0),
discounted_price numeric CHECK (discounted_price > 0),
CHECK (price > discounted_price)
);
```

Questo Vincolo non è collegato, come i primi due, ad una sola colonna

- I primi due vincoli sono vincoli di colonna, mentre il terzo che è un vincolo di tabella è scritto separatamente da qualsiasi definizione di colonna.
- I vincoli di colonna possono anche essere scritti come vincoli di tabella, mentre il contrario non è possibile PostgreSQL non applica questa regola, che dovrebbe essere seguita per garantire la portabilità in altri RDBMS

BASI DI DATI

Concetti base – SQL

CHECK CONSTRAINTS

- L'esempio sopra potrebbe anche essere scritto come (è solo una questione di gusti):

```
CREATE TABLE products (
product_no integer,
name text,
price numeric,
CHECK (price > 0),
discounted_price numeric,
CHECK (discounted_price > 0),
CHECK (price > discounted_price)
);
```

- **oppure:**

```
CREATE TABLE products (
product_no integer,
name text,
price numeric CHECK (price > 0),
discounted_price numeric,
CHECK (discounted_price > 0 AND price > discounted_price)
);
```

BASI DI DATI

Concetti base – SQL

CHECK CONSTRAINTS

- I nomi possono essere assegnati ai vincoli di tabella nello stesso modo dei vincoli di colonna:

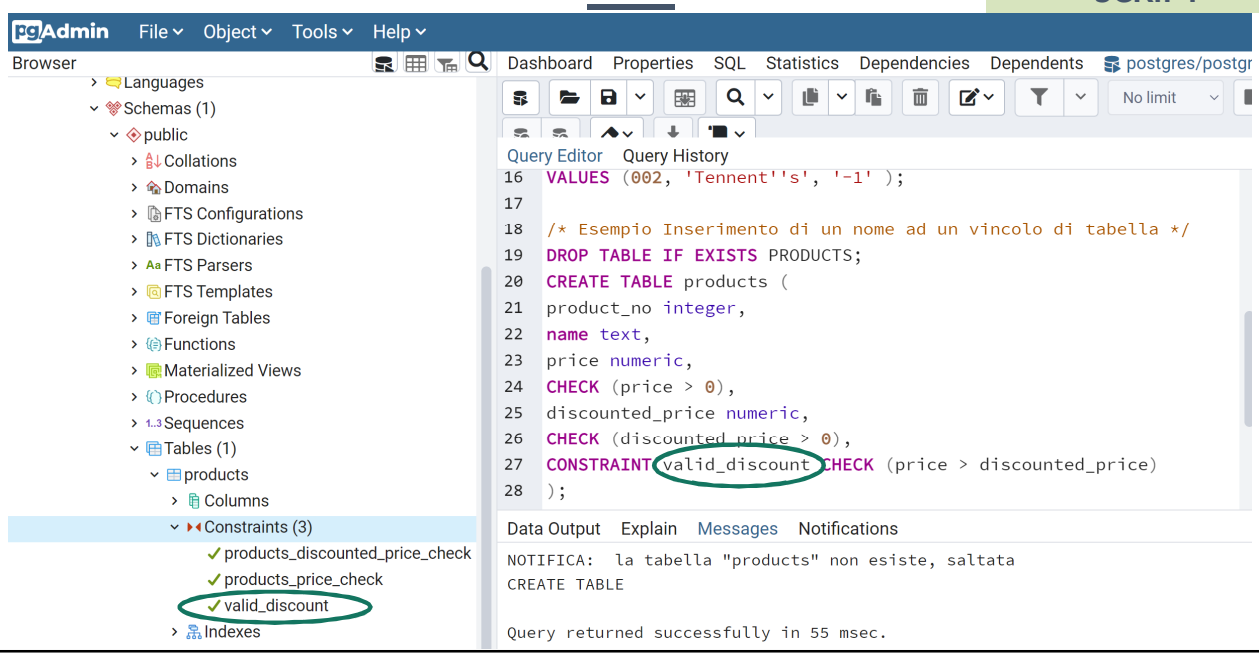
```
CREATE TABLE products (
product_no integer,
name text,
price numeric,
CHECK (price > 0),
discounted_price numeric,
CHECK (discounted_price > 0),
CONSTRAINT valid_discount CHECK (price > discounted_price)
);
```

- Un vincolo di controllo è soddisfatto se l'espressione di controllo restituisce true o il valore NULL. Poiché la maggior parte delle espressioni restituirà il valore NULL se un qualsiasi operando è NULL, possiamo avere valori NULL nelle colonne vincolate. Per garantire che una colonna non contenga valori NULL, occorre utilizzare il vincolo not-null.
- **Warning: PostgreSQL non supporta i vincoli CHECK che fanno riferimento a dati della tabella diversi dalla riga nuova o aggiornata da controllare**

BASI DI DATI

Concetti base – SQL

SCRIPT



BASI DI DATI

Concetti base – SQL

DEFINIZIONE DI DOMINI

- Adesso che conosciamo i **vincoli di DEFAULT e di CHECK,** vediamo come SQL permette di specificare nuovi domini usando il seguente comando:
- **CREATE DOMAIN**

  CREATE DOMAIN — create a new domain
- **Synopsis**

  CREATE domain NomeDominio AS TipodiDato
  [**DEFAULT** ValorediDefault]
  [**CHECK** Valore]

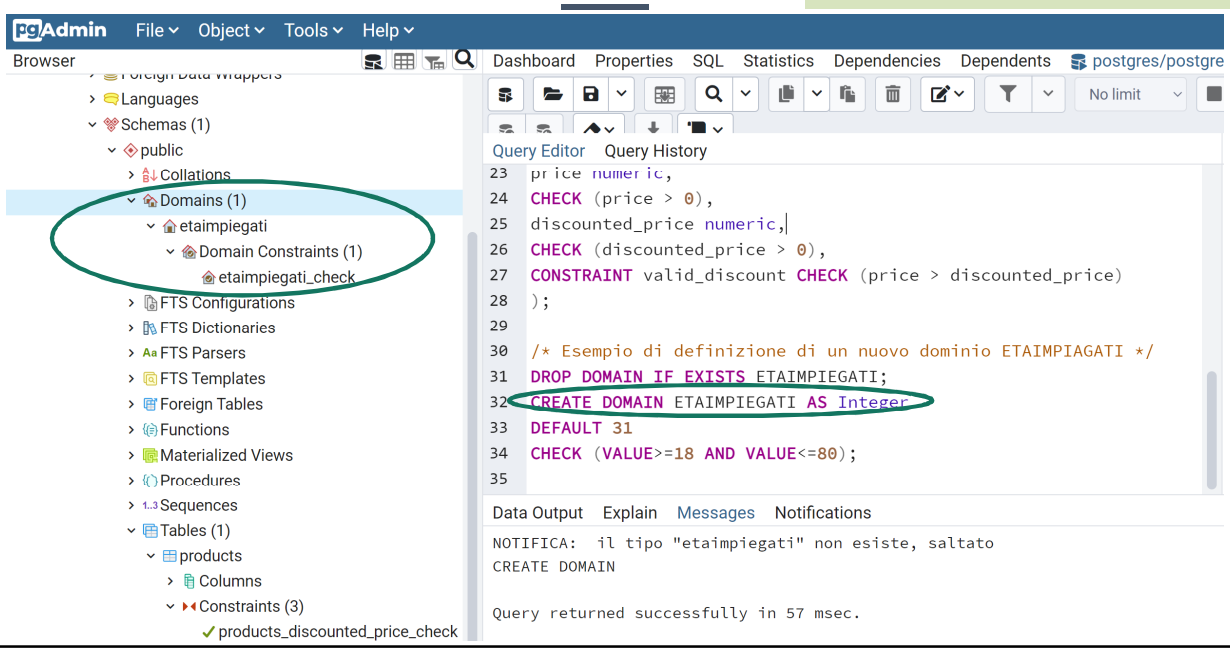
  dove;
  - ✓ **NomeDominio** è il nome del nuovo dominio;
  - ✓ **Tipodidato** è un dominio elementare predefinito (INT, CHAR, ecc.) oppure precedentemente definito dall'utente;
  - ✓ **DEFAULT** per impostare il valore di default del nuovo dominio
  - ✓ **CHECK** esprime le condizioni che i valori del nuovo dominio devono rispettare.

➢ ***Esempio di definizione di un nuovo dominio ETAIMPIEGATI***

CREATE DOMAIN ETAIMPIEGATI AS INTEGER
DEFAULT 31
CHECK (VALUE>=18 AND VALUE<=80)

❖ Value è una parola chiave utilizzata per indicare il valore del dominio in fase di definizione.

Nota: La dichiarazione di nuovi domini permette di associare un insieme di vincoli ad un dominio utilizzato in una tabella. Tale possibilità è utile quando si deve ripetere la stessa definizione di attributo in diverse tabelle. In tal modo si rende anche la definizione più facilmente modificabile e propagabile automaticamente a tutte le tabelle in cui il dominio in questione viene usato.



BASI DI DATI

Materiale utilizzato e bibliografia

- ***Le slide utilizzate dai docenti per le attività frontali sono in gran parte riconducibili e riprese dalle slide originali (con alcuni spunti parziali ripresi dai libri indicati) realizzate da:***
  - ✓ *autori del libro Basi di Dati (Atzeni e altri) testo di riferimento del corso Basi di Dati e sono reperibili su internet su molteplici link oltre che laddove indicato dagli stessi autori del libro;*
  - ✓ *Prof.ssa Tiziana Catarci e dal dott. Ing. Francesco Leotta – corso di Basi di Dati dell'Università degli Studi La Sapienza di Roma al seguente link ed altri: http://www.dis.uniroma1.it/~catarci/basidatGEST.html (molto Interessanti anche le lezioni su YouTube).*
  - ✓ *Proff. Luca Allulli e Umberto Nanni, Libro Fondamenti di basi di dati, editore HOEPLI (testo di facile lettura ed efficace).*
- ***Diverse slide su specifici argomenti utilizzate dai docenti per le attività frontali sono anche in parte riconducibili e riprese dalle slide originali – facilmente reperibili e accessibili su internet - realizzate da:***

Prof.ssa Roberta Aiello – corso Basi di Dati dell'Università di Salerno
Prof. Dario Maio - corso Basi di Dati dell'Università di Bologna al seguente link ed altri: http://bias.csr.unibo.it/maio
Prof. Marco Di Felice - corso Basi di Dati dell'Università di Bologna al seguente link ed altri: http://www.cs.unibo.it/difelice/dbsi/
Prof Marco Maggini e prof Franco Scarselli - corso Basi di Dati dell'Università di Siena ai seguenti link ed altri: http://staff.icar.cnr.it/pontieri/didattica/LabSI/lezioni/_preliminari-DB1%20(Maggini).pdf
Prof.ssa Raffaella Gentilini - corso Basi di Dati dell'Università di Perugia al seguente link ed altri: http://www.dmi.unipg.it/raffaella.gentilini/BD.htm
Prof. Enrico Giunchiglia - corso Basi di Dati dell'Università di Genova al seguente link ed altri: http://www.star.dist.unige.it/~enrico/BasiDiDati/
Prof. Maurizio Lenzerini - corso Basi di Dati dell'Università degli Studi La Sapienza di Roma al seguente link ed altri http://didatticainfo.altervista.org/Quinta/Database2.pdf

- The PostgreSQL Global Development Group - PostgreSQL nn.xx Documentation
- PostgreSQL (appendice - scaricabile dal sito del libro (area studenti) e www.postgresql.org