Basi di dati - Laboratorio

Corso di Laurea in Bioinformatica

Docente: Barbara Oliboni

Lezione 1

Contenuto della lezione

- Definizione di dati in SQL
  - Istruzione CREATE TABLE
- Domini SQL
- Vincoli intrarelazionali
- Inserimento di dati in una base di dati
  - Istruzione INSERT
- PostgreSQL
  - Connessione alla base di dati
  - Creazione tabelle e inserimento dati

SQL

- Structured Query Language
- SQL è stato definito nel 1973 ed è oggi il linguaggio più diffuso per i DBMS relazionali
- Il linguaggio SQL è composto di diverse parti:
  - Definizione delle strutture dati e dei vincoli di integrità
  - Linguaggio per modificare dati (inserimento, aggiornamento e cancellazione)
  - Linguaggio per interrogare la base di dati (Query language)

Definizione Dati in SQL

- Istruzione CREATE TABLE:
  - Definisce uno schema di relazione e ne crea un'istanza vuota
  - Specifica attributi, domini, vincoli

CREATE TABLE NomeTabella
(Attributo Tipo [Valore Default][Vincolo Attributo]
{, Attributo Tipo [Valore Default][Vincolo Attributo]}
{, Vincolo Tabella})

Domini

- Domini elementari (predefiniti):
  - Carattere: singoli caratteri o stringhe anche di lunghezza variabile
  - Bit: singoli booleani (flag) o stringhe
  - Numerici, esatti e approssimati
  - Data, ora, intervalli di tempo
- Domini definiti dall'utente

Dominio CARATTERE

- Permette di rappresentare singoli caratteri oppure stringhe
- La lunghezza delle stringhe può essere fissa o variabile

  character [varying][(Lunghezza)]
- Forme abbreviate:

  character → CHAR

  character varying (20) → VARCHAR(20)

Dominio BIT

- Tipicamente usato per rappresentare attributi, detti FLAG, che specificano se l'oggetto rappresentato da una tupla possiede o meno quella proprietà
- Si può anche definire un dominio "stringa di bit"

bit [varying][(Lunghezza)]

Dominio TIPI NUMERICI ESATTI

- Permette di rappresentare valori interi o valori decimali in virgola fissa
- SQL mette a disposizione 4 diversi tipi:
  - NUMERIC
  - DECIMAL

    numeric [(Precisione [, Scala])]

    decimal [(Precisione [, Scala])]

    Numeric(4,2) → 4 cifre significative, 2 cifre dopo la virgola
  - INTEGER
  - SMALLINT

Dominio
TIPI NUMERICI APPROSSIMATI

- Permette di rappresentare valori numerici approssimati mediante l'approccio in virgola mobile
- SQL mette a disposizione 3 diversi tipi numerici approssimati:
  - REAL

    (in postgres: "range of at least 1E-37 to 1E+37 with a precision of at least 6 decimal digits")
  - DOUBLE PRECISION

    (in postgres: "range of around 1E-307 to 1E+308 with a precision of at least 15 digits")
  - SQL-standard notazione:

    float [(Precisione)] → (Precisione = cifre mantissa)

Domini per il TEMPO

- Permette di rappresentare istanti di tempo
  - DATE: (year, month, day)
  - TIME: (hour, minute, second)
  - TIMESTAMP: date ∪ time

    time [(Precisione)][with time zone]

    timestamp [(Precisione)][with time zone]

    Precisione = numero di cifre decimali usate per rappresentare le frazioni di secondo

    with time zone = se specificato risultano disponibili due campi in più: timezone_hour e timezone_minute che rappresentano la differenza con l'ora di Greenwich

CREATE TABLE: Esempio

```
CREATE TABLE Impiegato
(    Matricola    CHAR(6),
     Nome         VARCHAR(20),
     Cognome      VARCHAR(20),
     Qualifica    VARCHAR(20),
     Stipendio    FLOAT       )
```

Vincoli intrarelazionali

Vincoli di integrità: sono proprietà che devono essere soddisfatte da ogni istanza della base di dati

Vincoli di integrità intrarelazionali: riguardano proprietà che si riferiscono a singole relazioni della base di dati:

- NOT NULL: attributo non nullo
- UNIQUE: definisce chiavi
- PRIMARY KEY: definisce la chiave primaria (una sola, implica NOT NULL)
- CHECK(espressione): vincolo generico

NOT NULL

- Implica che il valore nullo non sia ammesso come valore dell'attributo
  - Il valore dell'attributo deve essere specificato in fase di inserimento

```
Nome VARCHAR(20) NOT NULL
```

UNIQUE

- Impone che i valori di un attributo (o di un insieme di attributi) siano una superchiave, quindi tuple differenti della tabella non possono avere gli stessi valori
- Si può definire su:
  - un solo attributo
  - un insieme di attributi

```
Matricola       CHAR(6) UNIQUE

Nome            VARCHAR(20),
Cognome         VARCHAR(20),
UNIQUE(Nome, Cognome)
```

UNIQUE su più attributi: attenzione!

```
Nome        VARCHAR(20) NOT NULL,
Cognome     VARCHAR(20) NOT NULL,
UNIQUE(Nome,Cognome)
```

Impone che non ci siano due righe che abbiano uguali sia il nome che il cognome

```
Nome        VARCHAR(20) NOT NULL UNIQUE,
Cognome     VARCHAR(20) NOT NULL UNIQUE,
```

Impone che non ci siano due righe che abbiano lo stesso nome o lo stesso cognome

PRIMARY KEY

- Specifica la chiave primaria della relazione
  - Si usa una sola volta per tabella
  - Implica una definizione di NOT NULL
- Due forme:
  - Nella definizione di un attributo, se forma da solo la chiave primaria

    ```
    Matricola       CHAR(6) PRIMARY KEY
    ```

  - Come definizione separata a livello di tabella (necessario quanto la chiave primaria è composta da più attributi)

    ```
    Nome            VARCHAR(20),
    Cognome         VARCHAR(20),
    PRIMARY KEY(Nome, Cognome)
    ```

CREATE TABLE
Esempio DEFAULT

```
CREATE TABLE Impiegato
(    Matricola     CHAR(6)  PRIMARY KEY,
     Nome          VARCHAR(20) NOT NULL,
     Cognome       VARCHAR(20) NOT NULL,
     Qualifica     VARCHAR(20),
     Stipendio     FLOAT  DEFAULT 0.0,
     UNIQUE(Cognome, Nome)                )
```

Il valore che deve assumere l'attributo quando viene inserita una riga nella tabella senza che sia specificato un valore per l'attributo stesso. Se non specificato, si assume come valore di default NULL

CREATE TABLE
Esempio CHECK

```
CREATE TABLE Impiegato
(    Matricola     CHAR(6)  PRIMARY KEY,
     Nome VARCHAR(20) NOT NULL,
     Cognome       VARCHAR(20) NOT NULL,
     Qualifica     VARCHAR(20),
     Stipendio     FLOAT  DEFAULT 100.0,
     UNIQUE(Cognome, Nome),
     CHECK (Stipendio>= 100) )
```

INSERT

- Come popolare una tabella (inserimento righe):

```
INSERT INTO NomeTabella [(ElencoAttributi)]
VALUES (Elenco di Valori)
```

```
INSERT INTO Impiegato
(Matricola,Nome,Cognome)
VALUES ('A00001', 'Mario', 'Rossi')
```

PostgreSQL

http://www.postgresql.org/

PostgreSQL

- Sistema di gestione di basi di dati relazionali ad oggetti sviluppato in varie fasi fin dal 1977
- È largamente considerato come il sistema di gestione di basi di dati *Open Source* più avanzato al mondo e fornisce molte funzionalità che si vedono tradizionalmente soltanto in prodotti commerciali per aziende

PostgreSQL

- PostgreSQL è un DBMS relazionale ad oggetti
- Software multipiattaforma di pubblico dominio
- L'interazione tra un utente (programmatore della base di dati, o utente finale) e la base di dati considerata avviene secondo il modello client-server
  - L'architettura impiegata da PostgreSQL riserva un processo per ogni utente
  - Esiste un processo principale che si biforca, allo scopo di fornire una connessione aggiuntiva per ogni client che cerca di connettersi

PostgreSQL

Per ogni connessione stabilita vengono coinvolti tre processi UNIX:

- il postmaster: un processo deamon con funzione di supervisione (gestisce le basi di dati presenti sul server)
  - Non è concepito per essere un'interfaccia diretta all'utente ma può essere connesso a moltissimi client
  - In fase di avvio, il processo postmaster viene messo in background, in ascolto sulla porta TCP/IP caratteristica delle connessioni con i client
- L'applicazione frontend dell'utente
  - Esistono numerose interfacce attraverso le quali i client si possono connettere con il processo postmaster
    - il client più diffuso è *psql*
    - Ogni utente che si connette lancia questo programma
- un backend database server (per ogni connessione)

Uso locale di PostgreSQL

- Il server **sqlserver** è anche un server PostgreSQL
  - Sono disponibili tante basi di dati quanti sono gli utenti
  - Ogni utente accede alla propria base di dati
    - *dblabXXX* è la base di dati dell'utente *userlabXXX*
- Come ci si connette?
  - **export PGUSER=userlabXXX**
  - **psql** –h <nome server> –d <nome database>
    psql –h sqlserver –d dblabXXX

Esempio di connessione

```
oliboni@feisty:~$ export PGUSER=userlab200
oliboni@feisty:~$ psql -h sqlserver -d dblab200
Password:
Benvenuto in psql 7.4.13, il terminale interattivo per
   PostgreSQL.

Digita:  \copyright per i termini di distribuzione
       \h per gli aiuti sui comandi SQL
       \? per gli aiuti sui comandi interni
       \g oppure punto e virgola per eseguire una query
       \q per uscire

dblab200=>
```

Psql

- Subito dopo l'avvio di *psql* appare un breve sommario di quattro *comandi slash*
  - **\h** per l'aiuto su SQL
  - **\?** per l'aiuto sui comandi *psql*
  - **\g** per l'esecuzione delle query
  - **\q** per uscire da *psql* una volta terminato

Psql

- **`\c[onnect] [nomedb| - [utente]]`**
  si connette ad un nuovo database
- **`\C <titolo>`**
  titolo della tabella
- **`\d <table>`**
  descrive la tabella (o la vista)
- **`\d{t|i|s|v}`**
  elenca tabelle/indici/sequenze/viste
- **`\d{p|S|l}`**
  elenca permessi/tabelle di sistema/large object
- **`\da`**
  elenca gli aggregati

Psql

- **`\dd [oggetto]`**
  elenca i commenti per tabella, tipo, funzione o operatore
- **`\df`**
  elenca le funzioni
- **`\do`**
  elenca gli operatori
- **`\dT`**
  elenca i tipi di dati
- **`\e [file]`**
  permette la modifica del buffer della query attuale o di [file] con un editor esterno
- **`\echo <testo>`**
  scrive il testo sullo stdout
- **`\f <sep>`**
  cambia il separatore dei campi

Psql

- **`\g [file]`**
  invia la query al backend (e I risultati in [file] o |pipe)
- **`\h [cmd]`**
  aiuto sulla sintassi dei comandi sql; * per tutti i comandi
- **`\H`**
  attiva o disattiva la modalità HTML (attualmente disinserita)
- **`\i <file>`**
  legge ed esegue la query da file
- **`\l`**
  elenca tutti i database
- **`\o [file]`**
  invia tutti i risultati della query nel [file] o | pipe
- **`\p`**
  mostra il contenuto del buffer della query attuale

Psql

- **`\q`**
  esce da psql
- **`\r`**
  cancella il buffer della query
- **`\s [file]`**
  stampa lo storico dei comandi (history) e lo salva in [file]
- **`\set <var> <valore>`**
  imposta una variabile interna
- **`\t`**
  visualizza solo le righe
- **`\T <tags>`**
  imposta gli attributi dei tag HTML per le tabelle

Psql

- `\unset <var>`
  elimina una variabile interna
- `\w <file>`
  scrive il buffer della query attuale su <file>
- `\x`
  attiva e disattiva l'ouptut esteso
- `\z`
  elenca i permessi di accesso alle tabelle
- `\! [cmd]`
  esce sulla shell o esegue un comando

Uso locale di PostgreSQL

- Il nome della tabella è univoco in una base di dati
  - Non possono essere create due tabelle con lo stesso nome
- Terminare ogni comando SQL con il carattere

;

Creazione Tabelle e inserimento dati

```
CREATE TABLE Persona (
CodiceFiscale char(16) PRIMARY KEY,
Nome          varchar(20),
Cognome       varchar(20),
Indirizzo     varchar(50),
Email         varchar(30) );

INSERT INTO Persona VALUES
('MRARSS70D10L781T', 'Mario', 'Rossi',
'via Dante, 3, ROMA','mario.rossi@gmail.com');
```

Creazione Tabelle e inserimento dati

- Due modi:
  - Da prompt
    - Si scrive l'istruzione SQL direttamente da prompt
  - Da file
    - Si usa il comando slash `\i` di *psql* per richiedere a *psql* la lettura e l'interpretazione di un file di input situato sul proprio filesystem locale

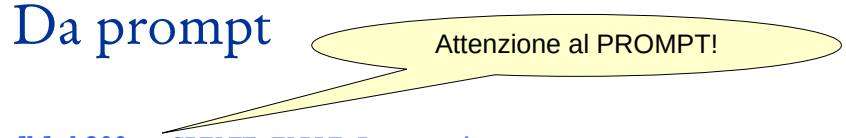
Da prompt

- Da *psql* connettersi alla propria base di dati

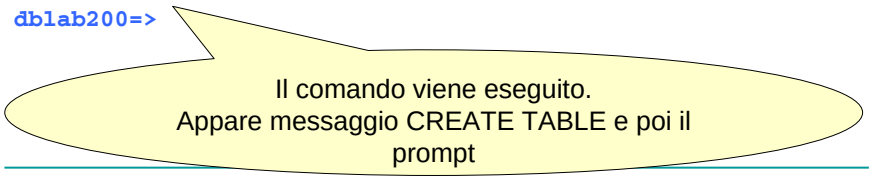
  `dblabXXX =>`

- Digitare l'istruzione
  - tutto ciò che viene digitato sarà accodato finché non verrà digitato il carattere `;`
  - l'istruzione può quindi essere suddivisa su righe diverse
  - Nota Bene: prestare attenzione al prompt!

Da prompt




```
dblab200=> CREATE TABLE Persona(
dblab200(> CodiceFiscale char(16) PRIMARY KEY,
dblab200(> Nome varchar(20),
dblab200(> Cognome varchar(20),
dblab200(> Indirizzo varchar(50),
dblab200(> Email varchar(30)
dblab200(> );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit
   index "persona_pkey" for table "persona"
CREATE TABLE
dblab200=>
```



Lista delle relazioni: \d

```
dblab200=> \d
           Lista delle relazioni
 Schema |  Nome   |  Tipo   | Proprietario
--------+---------+---------+--------------
 public | persona | tabella | userlab200
(1 riga)
```

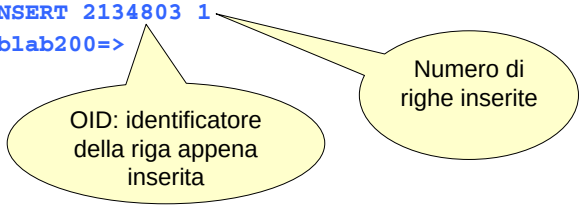
Descrizione relazione: \d NomeRelazione

```
dblab200=>
dblab200=> \d persona
              Tabella "public.persona"
    Colonna    |         Tipo          | Modificatori
---------------+-----------------------+--------------
 codicefiscale | character(16)         | not null
 nome          | character varying(20) |
 cognome       | character varying(20) |
 indirizzo     | character varying(50) |
 email         | character varying(30) |
Indici:
    "persona_pkey" chiave primaria, btree (codicefiscale)

dblab200=>
```

Da prompt: inserimento

```
dblab200=>
dblab200=> INSERT INTO Persona VALUES
dblab200-> (
dblab200(> 'MRARSS70D10L781T',
dblab200(> 'Mario','Rossi',
dblab200(> 'Via Dante, 3, ROMA',
dblab200(> 'mario.rossi@gmail.com')
dblab200-> ;
INSERT 2134803 1
dblab200=>
```



Da file: creazione e inserimento

- File CreaCitta.txt

```
CREATE TABLE Citta (
        Codice    char(5) PRIMARY KEY,
        Nome varchar(20));
INSERT INTO Citta VALUES ('VR001', 'Verona');
INSERT INTO Citta VALUES ('MI002', 'Milano');
```

```
dblab200=> \i CreaCitta.txt
psql:CreaCitta.txt:1: NOTICE:  CREATE TABLE / PRIMARY KEY
  will create implicit index "citta_pkey" for table
  "citta"
CREATE TABLE
INSERT 2134814 1
INSERT 2134815 1
dblab200=>
```

Da file: creazione e inserimento

```
dblab200=> \d
           Lista delle relazioni
 Schema |  Nome   |  Tipo   | Proprietario
--------+---------+---------+--------------
 public | citta   | tabella | userlab200
 public | persona | tabella | userlab200
(2 righe)

dblab200=>
```

Da file: creazione e inserimento

```
dblab200=> \d Citta
             Tabella "public.citta"
 Colonna |         Tipo          | Modificatori
---------+-----------------------+--------------
 codice  | character(5)          | not null
 nome    | character varying(20) |
Indici:
    "citta_pkey" chiave primaria, btree (codice)

dblab200=>
```

Visualizzare quanto inserito

```
dblab200=> SELECT * FROM persona;
  codicefiscale   | nome  | cognome |     indirizzo      |         email
------------------+-------+---------+--------------------+-----------------------
 MRARSS70D10L781T | Mario | Rossi   | Via Dante, 3, ROMA | mario.rossi@gmail.com
(1 riga)

dblab200=>
dblab200=> SELECT * FROM Citta;
 codice |  nome
--------+--------
 VR001  | Verona
 MI002  | Milano
(2 righe)

dblab200=>
```

Tipi di Dati principali in PostgreSQL

- **varchar(n)** stringa di lunghezza variabile minore o uguale a "n"
- **char** carattere singolo
- **char(n)** stringa di lunghezza fissa di "n" caratteri
- **integer** un intero di non più di nove cifre
- **float** un numero in virgola mobile
- **real** numero reale
- **date** data
- **time** l'orario
- **timestamp** data + orario
- **interval** intervallo di tempo