SAPIENZA
UNIVERSITÀ DI ROMA

Basi di Dati, Modulo 2

Sapienza Università di Roma
Facoltà di Ing. dell'Informazione, Informatica e Statistica
Laurea in Informatica
Prof. Toni Mancini
http://tmancini.di.uniroma1.it

Database B.2.4.1 (DB.B.2.4.1)

Basi di dati

Sistemi di Gestione di Basi di Dati Relazionali

Database

Il Database Accademia

Versione 2024-05-20

1 Schema Relazionale

Il database Accademia è definito sul seguente insieme di domini e sul seguente schema relazionale con vincoli.

Definizione dei domini

- **Strutturato**
  enum ('Ricercatore', 'Professore Associato', 'Professore Ordinario')
- **LavoroProgetto**
  enum ('Ricerca e Sviluppo', 'Dimostrazione', 'Management', 'Altro')
- **LavoroNonProgettuale**
  enum ('Didattica', 'Ricerca', 'Missione', 'Incontro Dipartimentale', 'Incontro Accademico', 'Altro')
- **CausaAssenza**
  enum ('Chiusura Universitaria', 'Maternita', 'Malattia')
- **PosInteger**
  integer ≥ 0
- **StringaM**
  varchar(100)
- **NumeroOre**
  integer tra 0 e 8
- **Denaro**
  real ≥ 0

Schema relazionale con vincoli della base dati

Persona (id: PosInteger, nome: StringaM, cognome: StringaM, posizione: Strutturato, stipendio: Denaro)

Progetto (id: PosInteger, nome: StringaM, inizio: date, fine: date, budget: Denaro)

[VincoloDB.1] *altra chiave:* (nome)
[VincoloDB.2] *ennupla:* inizio $<$ fine

WP (progetto: PosInteger, id: PosInteger, nome: StringaM, inizio: date, fine: date)

[VincoloDB.3] *ennupla:* inizio $<$ fine
[VincoloDB.4] *altra chiave:* (progetto, nome)
[VincoloDB.5] *foreign key:* progetto references Progetto(id)

AttivitaProgetto (id: PosInteger, persona: PosInteger, progetto: PosInteger, wp: PosInteger, giorno: date, tipo: LavoroProgetto, oreDurata: NumeroOre)

[VincoloDB.6] *foreign key:* persona references Persona(id)
[VincoloDB.7] *foreign key:* (progetto, wp) references WP(progetto, id)

AttivitaNonProgettuale (id: PosInteger, persona: PosInteger, tipo: LavoroNonProgettuale, giorno: date, oreDurata: NumeroOre)

[VincoloDB.8] *foreign key:* persona references Persona(id)

Assenza (id: PosInteger, persona: PosInteger, tipo: CausaAssenza, giorno: date)

[VincoloDB.9] *altra chiave:* persona, giorno
[VincoloDB.10] *foreign key:* persona references Persona(id)

2 Importazione del Database in PostgreSQL

I file `.sql` allegati definiscono i domini, lo schema relazionale e i vincoli esterni del database Accademia e lo popolano con istanze di esempio.

Per importare il database in PostgreSQL, è possibile sia utilizzare la shell `psql` del DBMS accessibile dal container Docker di PostgreSQL, sia operare graficamente tramite `pgAdmin`.

I file vanno importati nel seguente ordine:

1. `domains-tables.sql`
2. `constraints.sql`
3. `data.sql`.

2.1 Importazione tramite la shell `psql`

1. Assicurarsi che il container nel quale è installato PostgreSQL sia attivo tramite il comando `docker ps -a`, che visualizza lo stato di tutti i container.
2. Copiare i file `.sql` nella cartella `userData/`, che viene montata anche all'intero del container.
3. Eseguire la shell `psql` all'interno del container di PostgreSQL con il comando:

   ```
   docker exec -it postgres_container psql -U postgres
   ```

   dove `postgres_container` è il nome del container, `postgres` è il nome utente di default per accedere al DBMS (se non è stato cambiato durante l'installazione).
4. Al momento dell'ingresso nella shell `psql`, il database corrente è `postgres` (il DB di servizio di PostgreSQL, che non va mai modificato). Eseguire i comandi:

```
Type "help" for help
postgres=#CREATE DATABASE Accademia;
postgres=#\c Accademia
accademia=#\i domains-tables.sql
accademia=#\i constraints.sql
accademia=#\i data.sql
accademia=#exit
```

I comandi, nell'ordine: creano il DB (comando SQL standard); cambiano il DB corrente in Accademia (comando della shell); eseguono il codice in un file `.sql` (comando della shell). L'ultimo comando termina la shell.

2.2 Importazione tramite l'interfaccia di pgAdmin

1. Puntare un browser su `localhost:5000` (o altra porta, se modificata nel file `.env`) e fare login con le credenziali riportare nelle istruzioni di installazione.
2. Creare il database Accademia mediante il menu Object/Create.
3. Selezionare il nuovo database Accademia nell'albero dei database presenti nel server (potrebbe essere necessario ricaricare la pagina per visualizzarlo).
4. Caricare i tre file `.sql` all'interno del container tramite il menu Tools/Storage manager.
5. Sempre assicurandosi di aver selezionato, dall'albero dei database, il database Accademia, aprire dal menu Tools un Query tool, ovvero un'area di testo dove poter inviare query ed altri comandi SQL al DBMS. Assicurarsi che il nome della scheda del query tool mostri che è collegato effettivamente al database Accademia.
6. Usare il pulsante Open file per aprire i file `.sql` nell'ordine descritto alla sezione precedente e lanciare il codice mediante il pulsante Execute.

Data la taglia molto ridotta dei file `.sql`, in alternativa è possibile evitare di caricare i tre file all'interno del container, ma di usare le funzionalità di copia/incolla nel Query tool. Attenzione però: tale approccio non funzionerà correttamente con file `.sql` di grandi dimensioni.