

SQL

Esercitazioni pratiche

- Per SQL è possibile (e fondamentale) svolgere esercitazioni pratiche
- Verranno anche richieste come condizione per svolgere le prove parziali
- Soprattutto sono utilissime
- Si può utilizzare qualunque DBMS
 - IBM DB2, Microsoft SQL Server, Oracle, PostgreSQL o anche un servizio online (Sqliteonline)
- A lezione utilizziamo PostgreSQL e Sqliteonline

Nota

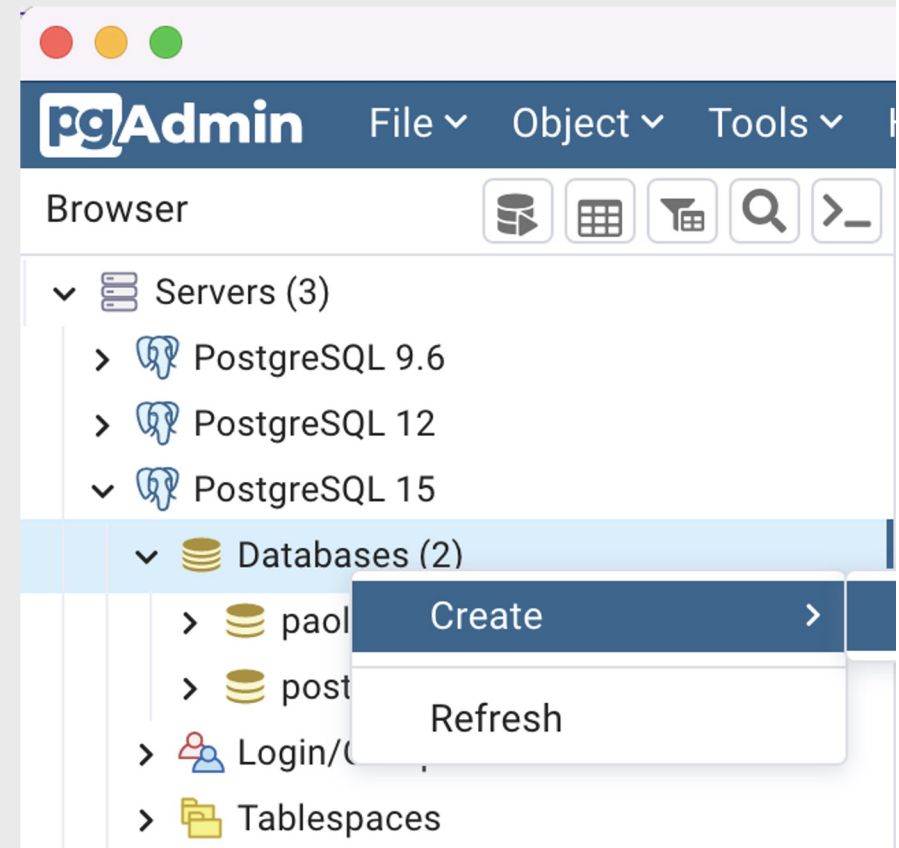
- Un dbms è un programma eseguito in background (un "demone"), cioè senza che sia sotto il controllo dell'utente
- Rimane in ascolto su una porta TCP, attraverso la quale interagisce con uno o più client
 - Tipicamente, un DBMS riceve in input comandi SQL e produce in output dati relazionali

Postgres

- Usiamo il server dbms (postgres) e un client (PgAdmin)
- Scaricare Postgres <https://www.postgresql.org/download/>
- L'installer include
 - il server (postgres)
 - un client (PgAdmin), che usiamo per interagire con il server (mandare istruzioni SQL, visualizzare i risultati)
- La porta TCP di default usata Postgres è la 5432
 - Manteniamo il default
- Durante l'installazione, il server ci chiede di impostare uno username (e relativa password): è l'utente autorizzato ad interagire con il server
 - Consiglio: username=postgres, password=postgres

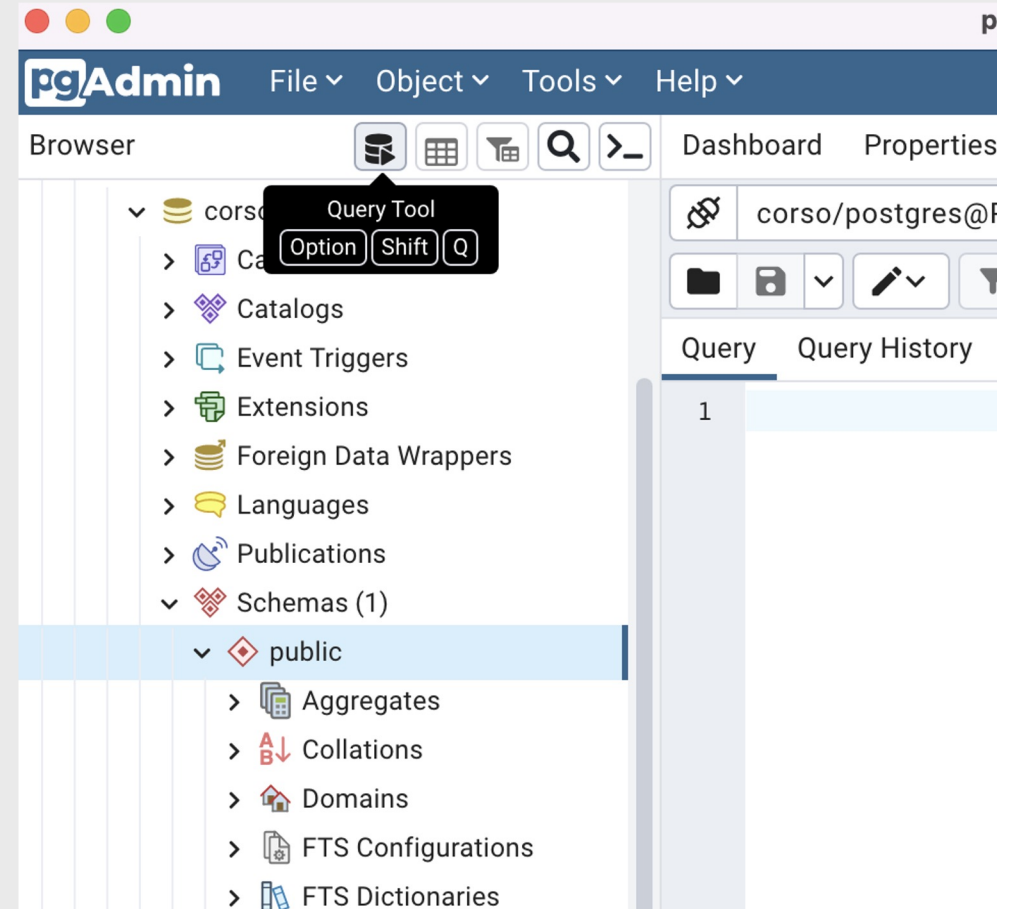
Come usare PgAdmin (per interagire con Postgres)

- Lanciare pgAdmin
- Creare un database (ad esempio: corso)



Come usare PgAdmin (per interagire con Postgres)

- Espandere l'albero a sinistra fino a "Schemas"
- Possiamo creare uno schema o usare lo schema pubblico (noi usiamo lo schema pubblico)
- E poi
 - lavorare sugli elementi dello schema
 - oppure lanciare "Query tool" (SQL interattivo): scelta consigliata



Come usare Sqliteonline

<https://sqliteonline.com/>

CREATE TABLE, esempi

```
CREATE TABLE corsi (  
    codice numeric NOT NULL PRIMARY KEY,  
    titolo character(20) NOT NULL,  
    cfu numeric NOT NULL);
```

```
CREATE TABLE esami (  
    corso numeric REFERENCES corsi (codice),  
    studente numeric REFERENCES studenti,  
    data date NOT NULL,  
    voto numeric NOT NULL,  
    PRIMARY KEY (corso, studente));
```

La chiave primaria viene definita come NOT NULL anche se non lo specifichiamo (in Postgres)

Creazione di tabelle

- In molti sistemi si possono utilizzare strumenti diversi dal codice SQL per definire lo schema della base di dati
 - Ad esempio, in PgAdmin si può creare una tabella attraverso l'interfaccia grafica
- Generalmente, è meglio usare il codice SQL
 - è più veloce la scrittura
 - possiamo riutilizzarlo

Nota pratica: per i nomi delle tabelle e degli attributi usare solo il minuscolo, non usare lettere accentate, per nomi composti usare l'underscore come separatore (es. `citta_nascita`)

CREATE TABLE: Sintassi

```
CREATE TABLE tabella (  
    attributo1 tipo vincoli,  
    attributo2 tipo vincoli,  
    ...  
    attributon tipo vincoli);
```

```
CREATE TABLE persone (  
    nome text NOT NULL PRIMARY KEY,  
    eta numeric NOT NULL,  
    reddito numeric NOT NULL);
```

```
CREATE TABLE maternita (  
    figlio text REFERENCES persone (nome) NOT NULL  
    PRIMARY KEY,  
    madre text REFERENCES persone (nome));
```

SQL, operazioni sui dati

- interrogazione:
 - **SELECT**
- modifica:
 - **INSERT, DELETE, UPDATE**

Inserimento

(necessario per gli esercizi)

```
INSERT INTO Tabella [ ( Attributi ) ]  
VALUES( Valori )
```

oppure

```
INSERT INTO Tabella [ ( Attributi ) ]  
SELECT ...  
(vedremo più avanti)
```

```
INSERT INTO persone VALUES ('Mario',25,52)
```

```
INSERT INTO persone(nome, reddito, eta)  
VALUES('Pino',52,23)
```

```
INSERT INTO persone(nome, reddito)  
VALUES('Lino',55)
```

maternità

madre	<u>figlio</u>
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

paternità

padre	<u>figlio</u>
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

persone

<u>nome</u>	eta	reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Esercizi

- Definire la base di dati per gli esercizi
 - installare un sistema
 - creare lo schema
 - creare le relazioni (CREATE TABLE)
 - inserire i dati
- eseguire le interrogazioni
 - suggerimento, per chi usa Postgres: usare database diversi

```
create table persone (  
    nome char (10) not null primary key,  
    eta numeric not null,  
    reddito numeric not null);  
create table paternita (  
    padre char (10) references persone,  
    figlio char (10) primary key references persone);
```

```
...  
insert into Persone values('Andrea',27,21);
```

```
...  
insert into Paternita values('Sergio','Franco');
```

...
Vedere file:

http://dia.uniroma3.it/~atzeni/didattica/BDN/20222023/protected/BD-04-esempiSQL2022_persone.sql

Vediamo gli esempi

- con Relax

<http://dbis-uibk.github.io/relax/calc/gist/1362a9bb84dd2e4052561b714613b1de>

- con Sqliteonline

<https://sqliteonline.com/>

con i dati

http://dia.uniroma3.it/~atzeni/didattica/BDN/20222023/protected/BD-04-esempiSQL2022_persone.sql

L'espressione più semplice

- In Relax potevamo scrivere un'espressione composta da una sola relazione

 Paternita

- In SQL dobbiamo scrivere
 SELECT Padre, Figlio

 FROM Paternita

- Oppure

 SELECT *

 FROM Paternita

Operatori insiemistici

- Unione
- Intersezione
- Differenza

- Nell'algebra:
 - sugli stessi schemi
- Ma in Relax ...
 - ... e pure in SQL

Unione

R(A,B) S(A,B)

```
select A, B  
from R  
union  
select A , B  
from S
```

Unione

Unione "normale"

R(A,B) S(A,B)

```
select A, B  
from R  
union  
select A , B  
from S
```

Unione "forzata"

R(A,B) T(C,D)

```
select A, B  
from R  
union  
select C, D  
from T
```

Richiede ennuple "compatibili"
Quali nomi per il risultato?

Approccio posizionale

```
select padre, figlio  
from paternita  
union  
select madre, figlio  
from maternita
```

Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Operazione non commutativa (in molti sistemi)

```
select padre, figlio  
from paternita  
union
```

```
select madre, figlio  
from maternita
```

```
select madre, figlio  
from maternita  
union
```

```
select padre, figlio  
from paternita
```

Approccio posizionale!

```
select padre, figlio  
from paternita  
union  
select madre, figlio  
from maternita
```

OK (o quasi)

Approccio posizionale, 2

```
select padre, figlio  
from paternita  
union  
select figlio, madre  
from maternita
```

NO!

Funziona, ma produce
un risultato
indesiderabile

```
select padre, figlio  
from paternita  
union  
select madre, figlio  
from maternita
```

OK

Ridenominazione

R(A,B) A1, B1 nomi diversi

```
select A as A1, B as B1  
from R
```

Unione con ridenominazione

```
select padre as genitore, figlio  
from paternita  
union  
select madre as genitore, figlio  
from maternita
```

Notazione posizionale, 3

- Anche con le ridenominazioni il problema resta

```
select padre as genitore, figlio  
from paternita  
union
```

```
select figlio, madre as genitore  
from maternita
```

- Corretta:

```
select padre as genitore, figlio  
from paternita  
union
```

```
select madre as genitore, figlio  
from maternita
```

Le tabelle SQL non sono insiemi!

```
select A, B  
from R  
union  
select A , B  
from S
```

```
select A, B  
from R  
union all  
select A , B  
from S
```

Le tabelle SQL non sono insiemi!

```
select *  
from paternita  
union  
select *  
from paternita
```

```
select *  
from paternita  
union all  
select *  
from paternita
```

Differenza

```
select A, B  
from R  
except  
select A , B  
from S
```

Intersezione

```
select A, B  
from R  
intersect  
select A , B  
from S
```

Selezione

- Nome, età e reddito delle persone con meno di trenta anni

$SEL_{Eta < 30}(Persone)$

```
select *  
from persone  
where eta < 30
```

Condizione complessa

```
select *  
from persone  
where reddito > 25  
      and (eta < 30 or eta > 60)
```

Proiezione

- Nome e reddito di tutte le persone

$\text{PROJ}_{\text{Nome, Reddito}}(\text{Persone})$

```
select nome, reddito  
from persone
```

Selezione e proiezione

- Nome e reddito delle persone con meno di trenta anni

$\text{PROJ}_{\text{Nome, Reddito}}(\text{SEL}_{\text{Eta} < 30}(\text{Persone}))$

```
select nome, reddito  
from persone  
where eta < 30
```

Proiezione, con ridenominazione

- Nome e reddito di tutte le persone

$REN_{Anni \leftarrow Eta}(PROJ_{Nome, Eta}(Persone))$

```
select nome, eta as anni  
from persone
```

Proiezione, attenzione

```
select  
    madre  
from maternita
```

```
select distinct  
    madre  
from maternita
```

Ancora l'anomalia degli operatori insiemistici

```
select padre  
from paternita  
union  
select padre  
from paternita
```

```
select padre  
from paternita  
union all  
select padre  
from paternita
```