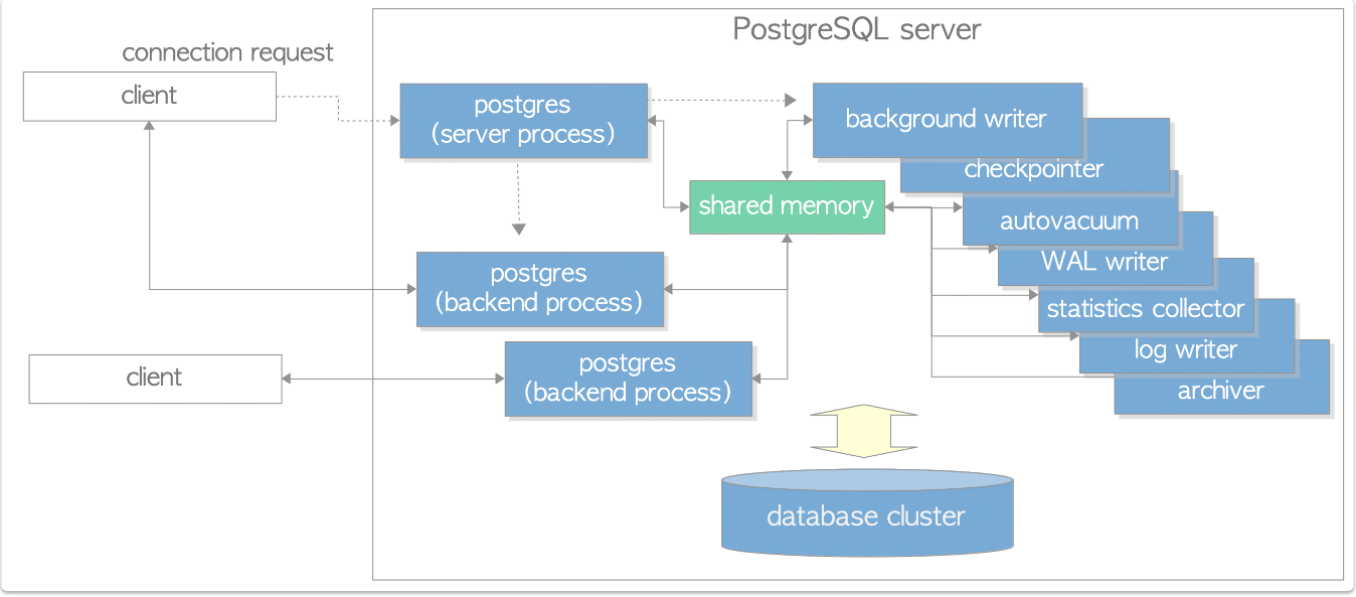
SQL

#sql #pratica #comandi-sql

Per `psql`, parlando del comando, esistono diversi *worker* all'interno della nostra macchina linux che vengono avviati all'istanza. Il seguente comando li mostra in command line.

```
ps aux | grep mysql
```

- Il nostro server si prende la libertà di memorizzare le pagine all'interno della RAM, ma si occupa anche di possibili fallimenti, il *logging collector* è uno dei processi;
- *stats collector* ogni tanto lancia (possiamo farlo manualmente) per acquisire info sul database;
- *autovacuum launcher*, nelle mie tabelle ogni tanto faccio eliminazioni che sono logiche (alcune n-uple diventano invalide), aiuta a recuperare spazio liberato compattando le tabelle;
- *logical replication launcher* per replicare (come altro server) i contenuti del server corrente (master & slave);
- il *background writer* scrive pagine nella shared memory lentamente verso la memoria persistente;
- *WAL writer* che trasferisce dati WAL su memoria persistente;
- *archiver* che archivia il log eseguito.



Interfaccia grafica

Using `pgAdmin`.
Nell'interfaccia grafica possiamo fare JOIN tra *schemi* (collezione di tabelle), possiamo vedere gli utenti con accesso al DB, possiamo vedere le relazioni. Le operazioni sono molteplici ma equivalgono alle stesse operazioni possibili tramite linea di comando.

> **Note**
>
> Personalmente non uso GUI perche' faccio prima da Visual Studio

Esempio SQL su tabella ISCRIZIONI UNIVERSITARIE

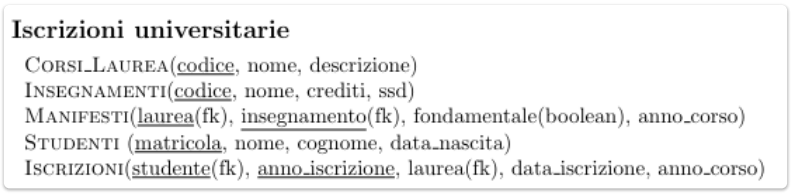
schemi_db

corsi_laurea

Creiamo all'interno del nostro client `sql` un database che possiamo chiamare come vogliamo:

```
CREATE DATABASE universita;
USE universita;
```

Al suo interno creiamo una tabella, `corsi_laurea`, che e' la stessa tabella negli esempi SQL visti in precedenza:



Iscrizioni universitarie

CORSI_LAUREA(codice, nome, descrizione)
INSEGNAMENTI(codice, nome, crediti, ssd)
MANIFESTI(laurea(fk), insegnamento(fk), fondamentale(boolean), anno_corso)
STUDENTI (matricola, nome, cognome, data_nascita)
ISCRIZIONI(studente(fk), anno_iscrizione, laurea(fk), data_iscrizione, anno_corso)

```
CREATE TABLE corsi_laurea (
    codice integer NOT NULL PRIMARY KEY,
    nome VARCHAR(200) NOT NULL,
    descrizione VARCHAR(200) NOT NULL

--  alternative
--  PRIMARY KEY (codice)
--  creiamo i nostri vincoli
--  CONSTRAINT codice_non_nullo CHECK(codice IS NOT NULL)
--  CONSRAINT codice_pk PRIMARY KEY (codice)
);
```

La tabella puo' essere modificata con DML `INSERT INTO`:

```
INSERT INTO corsi_laurea (codice, nome, descrizione)
VALUES (1, 'Informatica', 'Corso di Laurea triennale Informatica');

INSERT INTO corsi_laurea (codice, nome, descrizione)
VALUES (2, 'Matematica', 'Corso di Laurea triennale Matematica');
```

Possiamo aggiungere specifiche alla tabella, alterandola con `ALTER TABLE`:

```
ALTER TABLE corsi_laurea
ADD UNIQUE(nome);
```

---

lezione: 2022-10-07