



SQL3

*Python*

## **Cos'è un Database (DB) – Paragrafo teorico**

**Un Database, o base di dati, è un insieme organizzato di dati strutturati e correlati tra loro, progettato per essere facilmente accessibile, gestito e aggiornato.**

**I database vengono utilizzati per memorizzare informazioni in modo persistente, consentendo operazioni di inserimento, modifica, cancellazione e interrogazione dei dati.**

**Questi sistemi sono fondamentali in moltissimi ambiti, dal software gestionale ai siti web, passando per applicazioni mobile e sistemi industriali. L'accesso e la gestione dei database sono solitamente affidati a un DBMS (Database Management System), ovvero un software che si occupa dell'interazione tra l'utente e il database.**



## Tipi di Database – Lista con breve descrizione

- **Database relazionali (RDBMS)**

**Basati su tabelle (righe e colonne), utilizzano il linguaggio SQL. Ogni tabella ha relazioni con altre tramite chiavi. Esempi: MySQL, PostgreSQL, Oracle.**

- **Database non relazionali (NoSQL)**

**Progettati per grandi quantità di dati non strutturati o semi-strutturati. Non usano lo schema rigido delle tabelle. Tipologie comuni:**

- **Documentali:** archiviano dati in formato JSON o BSON (es. MongoDB)
- **Key-Value:** semplici coppie chiave-valore (es. Redis)
- **Colonnari:** ottimizzati per letture su grandi volumi (es. Cassandra)
- **A grafo:** rappresentano dati e relazioni con nodi e archi (es. Neo4j)

- **Database in memoria**

**Conservano i dati nella RAM per garantire altissime prestazioni (es. Redis, Memcached). Ideali per cache e dati temporanei.**



## **Cos'è sqlite3 in Python**

**In Python, sqlite3 è un modulo integrato nella libreria standard che permette di interfacciarsi con SQLite, un sistema di database relazionale leggero e senza bisogno di installazione o configurazione esterna.**

**SQLite memorizza i dati in un singolo file .db e supporta tutte le funzionalità base dell'SQL.**

**È ideale per applicazioni locali, test, piccoli progetti o prototipi. Il modulo sqlite3 consente di eseguire comandi SQL direttamente da Python, creando tabelle, inserendo dati, eseguendo query e altro ancora.**



## **Cos'è un CRUD – Paragrafo teorico**

**Il termine CRUD è un acronimo che rappresenta le quattro operazioni fondamentali che possono essere eseguite su un insieme di dati in un sistema informatico: Create, Read, Update e Delete.**

**Queste operazioni sono alla base di qualsiasi interazione con un database o una risorsa dati persistente. "Create" serve a inserire nuovi dati, "Read" a leggere o recuperare informazioni esistenti, "Update" a modificarle, e "Delete" a eliminarle, il modello CRUD è un concetto centrale nello sviluppo software, soprattutto in applicazioni web e sistemi gestionali, ed è spesso implementato attraverso interfacce utente (GUI), API RESTful o comandi SQL nei database relazionali.**

**Progettare un'applicazione CRUD significa costruire una struttura che consenta all'utente di gestire un insieme di dati in modo completo e coerente, spesso con l'ausilio di un backend (come Python, Java o PHP) collegato a un database.**



```
import sqlite3

# Connessione al database
conn = sqlite3.connect('esempio.db')
cur = conn.cursor()

# Query con filtro (voto maggiore di 25)
cur.execute("SELECT nome, voto FROM studenti WHERE voto > ?", (25,))

# Recupero dei risultati
studenti_meritevoli = cur.fetchall()

# Stampa dei risultati filtrati
for nome, voto in studenti_meritevoli:
    print(f"{nome} ha preso {voto}")

# Chiusura della connessione
conn.close()
```

## Cosa fa questo script

- Si connette al database esempio.db.
- Esegue una SELECT con filtro sulla colonna voto.
- Recupera e stampa solo i nomi e voti degli studenti che hanno ottenuto più di 25.



## Metodi principali di sqlite3

- 1. `sqlite3.connect()`

**Aprire una connessione a un file di database SQLite (o lo crea se non esiste). Serve da punto di ingresso per interagire con il database.**

- 2. `Connection.cursor()`

**Ritorna un oggetto cursor che serve per eseguire comandi SQL e gestire query.**



## Metodi principali di sqlite3

- **3. Connection.execute() e Connection.executemany()**

**Permettono di eseguire comandi SQL direttamente dalla connessione. execute() è per singole istruzioni, mentre executemany() è utile per eseguire lo stesso comando su più insiemi di dati (bulk insertion).**

- **4. Cursor.execute()**

**Esegue una singola istruzione SQL tramite il cursore.**





## Metodi principali di sqlite3

- **5. Cursor.executescript()**

**Permette l'esecuzione di più istruzioni SQL in uno script unico.**

- **6. Connection.commit()**

**Salva tutte le modifiche apportate al database dalla connessione (come inserimenti, aggiornamenti o cancellazioni).**

- **7. Connection.rollback()**

**Annulla tutte le modifiche non confermate (non ancora committate).**



## Metodi principali di sqlite3

- **8. Connection.close()**

**Chiude la connessione al database. È importante per liberare risorse.**

- **9. Cursor.fetchone(), Cursor.fetchmany(), Cursor.fetchall()**

**Metodi del cursore per recuperare i risultati di una query:**

- **fetchone()** restituisce la prossima riga (o None se non ci sono più dati).
- **fetchmany()** restituisce un insieme di righe.
- **fetchall()** restituisce tutte le righe rimanenti.

## **10. Connection.total\_changes()**

- **Restituisce il numero totale di righe modificate, inserite o eliminate dal database (dall'inizio della connessione).**



## In breve:

- **Connessione e cursori: `connect()`, `cursor()`, `execute()`, `executemany()`**
- **Gestione transazioni: `commit()`, `rollback()`**
- **Lettura risultati: `fetchone()`, `fetchmany()`, `fetchall()`**
- **Altri utili: `executescript()`, `total_changes()`, `close()`, costanti informative.**



## Esempio semplice con sqlite3

```
1.import sqlite3
2.
3.# Connessione a un database (o creazione se non esiste)
4.conn = sqlite3.connect('esempio.db')
5.
6.# Creazione di un cursore per eseguire comandi SQL
7.cur = conn.cursor()
8.
9.# Creazione di una tabella
10.cur.execute("""
11.     CREATE TABLE IF NOT EXISTS studenti (
12.         id INTEGER PRIMARY KEY,
13.         nome TEXT,
14.         voto INTEGER
15.     )
16.""")
17.
18.# Inserimento di un record
19.cur.execute("INSERT INTO studenti (nome, voto) VALUES (?, ?)",
20.            ("Marco", 28))
21.# Salvataggio delle modifiche
22.conn.commit()
23.
24.# Esecuzione di una query
25.cur.execute("SELECT * FROM studenti")
26.risultati = cur.fetchall()
27.
28.# Stampa dei risultati for riga in risultati:
29.     print(riga)
30.
31.# Chiusura della connessione
32.conn.close()
```

## Spiegazione

- **connect()** apre (o crea) un database SQLite.
- Il cursore (**cursor()**) serve per eseguire i comandi SQL.
- Viene creata una tabella studenti con tre colonne.
- Viene inserito un dato in modo sicuro (parametrico).
- Si esegue una query di selezione e si stampano i risultati.
- Infine, si chiude la connessione.



## Conclusione

**Il modulo sqlite3 in Python rappresenta uno strumento potente e allo stesso tempo estremamente semplice da usare per lavorare con database relazionali locali, senza necessità di installare server esterni.**

**È perfetto per piccole applicazioni, prototipi o esercitazioni didattiche, perché consente di sperimentare tutte le operazioni tipiche di un database (CRUD) utilizzando il linguaggio SQL.**

**La sua integrazione nativa nella libreria standard di Python lo rende immediatamente disponibile e portabile, mentre la sintassi chiara consente anche a chi è alle prime armi di apprendere velocemente i concetti fondamentali.**



**Buon MasterD a tutti**

