Introduction to SQLite3

- **What is SQLite3?**
- SQLite3 is a C library that provides a lightweight, disk-based database that doesn’t require a separate server process. It reads and writes directly to ordinary disk files.

- **Why use SQLite3?**
- **Serverless:** No need for a separate server process.
- **Zero Configuration:** No setup or administration required.
- **Cross-Platform:** Runs on any system with a file system.
- **Self-contained:** A single library.
- **Transactional:** ACID-compliant (Atomicity, Consistency, Isolation, Durability).

Atomicity

- Atomicity ensures that a transaction is treated as a single unit, which either completely succeeds or completely fails. If any part of the transaction fails, the entire transaction is rolled back, and the database remains unchanged.
- **Example:**
- Consider a banking system where you are transferring $100 from Account A to Account B.
- The transaction consists of two steps:
  - Deduct $100 from Account A.
  - Add $100 to Account B.
- Atomicity ensures that both steps occur successfully. If the first step succeeds but the second step fails (e.g., due to a system crash), the entire transaction will be rolled back, and Account A will still have the original $100.

Consistency

- Consistency ensures that a transaction brings the database from one valid state to another valid state. This means that any transaction will leave the database in a valid state, maintaining the defined rules, constraints, and integrity.
- **Example:**
- Using the same banking system example:
  - If Account A and Account B must always have a non-negative balance, the transaction must ensure this rule is not violated.
  - If you attempt to transfer $100 from Account A, which only has $50, the transaction should be aborted, maintaining consistency by not allowing Account A to have a negative balance.

Isolation

- Isolation ensures that transactions are executed in isolation from one another. Intermediate states of a transaction are invisible to other transactions, and transactions that occur concurrently do not affect each other.
- **Example:**
- Suppose two transactions occur simultaneously:
  - Transaction 1 transfers $100 from Account A to Account B.
  - Transaction 2 transfers $50 from Account B to Account A.
- Isolation ensures that these transactions do not interfere with each other, preventing inconsistencies such as one transaction reading intermediate results of another.

Durability

- Durability ensures that once a transaction has been committed, it will remain so, even in the event of a system failure. This means that the results of the committed transaction are permanently recorded in the database.
- **Example:**
- After transferring $100 from Account A to Account B, the transaction is committed.
- Even if the system crashes immediately after the commit, the changes will not be lost, and the database will still reflect the transfer of $100 from Account A to Account B when the system is back up.

Summary of ACID properties

- **Atomicity:** Ensures transactions are all-or-nothing.
- **Consistency:** Ensures transactions bring the database from one valid state to another.
- **Isolation:** Ensures transactions do not interfere with each other.
- **Durability:** Ensures committed transactions are permanent.

Setting Up SQLite3 in Python

- **SQLite3 Installation:**
- SQLite3 comes built-in with Python’s standard library, so no additional installation is needed.

Connecting to an SQLite3 Database

- **Connecting to SQLite3:**
- Create or open an SQLite database file:
- Use python code

CRUD operations

- Standard language for managing and manipulating databases.
- Basic Commands:
- `SELECT` to fetch data
- `INSERT` to add data
- `UPDATE` to modify data
- `DELETE` to remove data