Chapitre 7 - IntegrationSQLite et Python

1. Base de données dans une application
   - Paradigmes SGBD
   - Python - SQLite

Application utilisant un SGBD

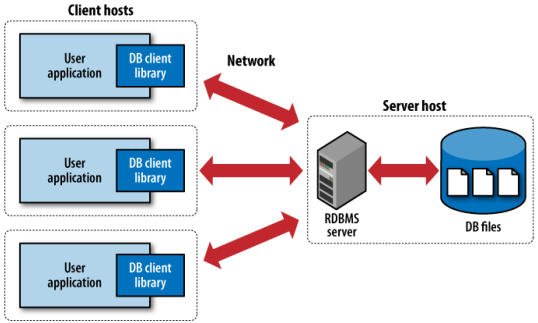
SQL Embarqué

- Le SQL a vocation à être utilisé dans un programme
- Transférer des informations du SGBD vers un programme (navigateur web, R, python, Java ...)

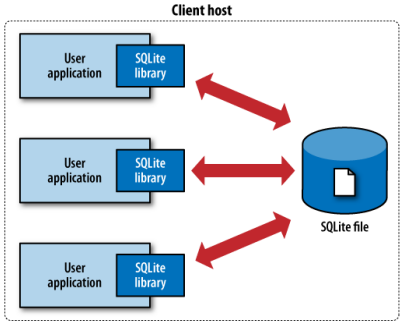
Paradigmes de SGBD

SQLLite vs SGDB traditionnels (Oracle, MySQL, PostgeSQL)

- La plus part des SGBD sont construits selon le paradigme client-serveur
- SQLite, au contraire, est directement intégrée dans l'application



(a) Traditional client-server architecture

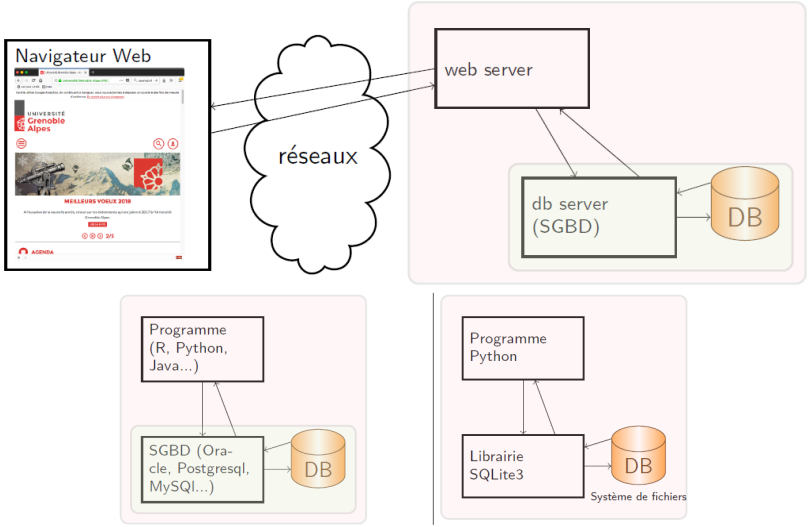


(b) SQLite serverless architecture

[1]

[1]Source: Kreibich, Jay A. 2010. "Using SQLite." O'Reilly Media, Inc., August 10.

Application Web utilisant un SGBD



Situations dont SQLite fonctionne bien et moins bien

[2]

Situations où SQLite fonctionne bien

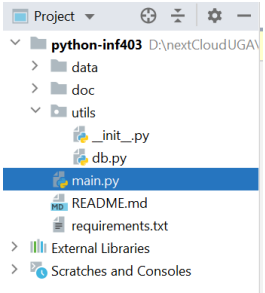
- Appareils embarqués
- Sites Web à trafic faible - moyen (moins de 100 000 visites / jour)
- L'analyse des données
- Remplacement des fichiers de disque *ad hoc*
- Bases de données internes ou temporaires
- ...

Situations où un SGBD client / serveur peut mieux fonctionner

- Applications client / serveur
- Données très volumineux (une BD SQLite est limitée en taille à 281 TB)
- Concurrence élevée

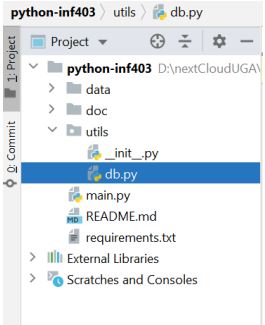
[2] Source: https://sqlite.org/whentouse.html

Python : Main



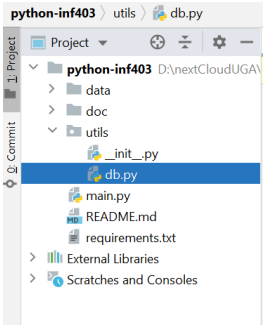

```python
from utils import db

def main():
        # Nom de la BD à créer
        db_file = "data/voile.db"

        # Créer une connexion a la BD
        conn = db.creer_connexion(db_file)

        # Remplir la BD
        print("1. On crée la bd et on l'initialise avec des premières valeurs.")
        db.mise_a_jour_bd(conn, "data/voile_creation.sql")
        db.mise_a_jour_bd(conn, "data/voile_inserts_ok.sql")

        # Lire la BD
        print("2. Liste de tous les bateaux")
        select_tous_les_bateaux(conn)
```

Python : Mise à jour de la BD



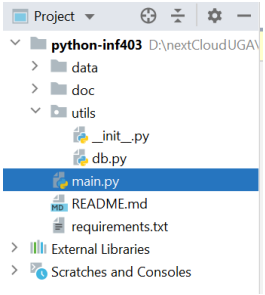

```python
def mise_a_jour_bd(conn: sqlite3.Connection, file: str):

    # Lecture du fichier et placement des requêtes dans un tableau
    sqlQueries = []

    with open(file, 'r') as f:
            createSql = f.read()
            sqlQueries = createSql.split(";")

    # Exécution de toutes les requêtes du tableau
    cursor = conn.cursor()
    for query in sqlQueries:
       cursor.execute(query)

    # Validation des modifications
    conn.commit()
```

Python : Connexion à la BD




```python
def creer_connexion(db_file):

        try:
                conn = sqlite3.connect(db_file)
                # On active les foreign keys
                conn.execute("PRAGMA foreign_keys = 1")
                return conn
        except sqlite3.Error as e:
                print(e)

        return None
```

Python : Python : Exemple SELECT




```python
def select_tous_les_bateaux(conn):

        cur = conn.cursor()
        cur.execute("SELECT * FROM Bateaux")

        rows = cur.fetchall()

        for row in rows:
                print(row)
```

Notion de transactions

- Une ou plusieurs requêtes sur la base de données
- Toutes sont validées, ou aucune
- Une transaction typique :
  1. Connexion à la base de données
  2. Exécutions sans erreur de requêtes (dont certaines sont des mises à jour)
  3. Une erreur ? annulation (rollback)
  4. Aucune erreur ? validation (commit)
  5. Déconnexion de la base de données

Conclusion

Les propriétés des transactions :

- Atomicité : toutes les opérations sont exécutées, ou aucune
- Cohérence : à l'issue de la transaction les données sont cohérentes
- Isolation : les effets en cours d'une transaction ne sont pas visibles par les autres
- Durabilité : les effets de la transactions persistent

Ces propriétés doivent être garanties par le SGBD.