Fiche de Procédure : Les Bases de SQLite

1. Qu'est-ce que SQLite ?

- SQLite est un système de gestion de base de données relationnelle (SGBDR) open source, léger, autonome et embarqué, qui fonctionne sans serveur externe ni configuration complexe.
- La base de données est stockée dans un fichier unique sur le disque, ce qui facilite la portabilité, l'intégration dans des applications, et l'utilisation sur des appareils à ressources limitées (mobiles, IoT, etc.).
- SQLite est largement utilisé dans les applications mobiles, navigateurs web, logiciels embarqués, et pour le prototypage rapide.

2. Fonctionnalités principales

- Autonome : Pas besoin de serveur, tout fonctionne via une bibliothèque logicielle intégrée à l'application.
- Sans configuration : Prêt à l'emploi, aucune installation ou gestion de serveur nécessaire.
- Stockage léger : Les fichiers de base de données sont petits et efficaces, parfaits pour les environnements à espace limité.
- Standard SQL : Prise en charge d'une grande partie du langage SQL, y compris les requêtes complexes, les jointures, les transactions, les vues, les index et les triggers.
- Portabilité : Le fichier de base de données peut être déplacé et utilisé sur différents systèmes sans conversion.

3. Concepts et structures de base

- Base de données : Un simple fichier (ex : `ma_base.db`) contenant toutes les tables, index, vues et triggers.
- Schéma : Définit la structure de la base (tables, colonnes, types de données, contraintes, relations).
- Table : Structure de stockage des données en lignes et colonnes, identifiée par un nom unique.
- Types de données :
    - `INTEGER` : entier

    - `TEXT` : chaîne de caractères
    - `REAL` : nombre à virgule flottante
    - `BLOB` : données binaires
- Clé primaire : Identifie de façon unique chaque ligne d'une table.
- Clé étrangère : Sert à relier plusieurs tables entre elles, assurant l'intégrité référentielle.

4. Commandes et requêtes SQL de base

- Créer une base de données :

```
sqlite3 ma_base.db
```

- Créer une table :

```
CREATE TABLE utilisateurs (
  id INTEGER PRIMARY KEY,
  nom TEXT NOT NULL,
  email TEXT UNIQUE NOT NULL
);
```

- Insérer des données :

```
INSERT INTO utilisateurs (nom, email) VALUES ('Alice',
'alice@email.com');
```

- Lire des données :

```
SELECT * FROM utilisateurs;
```

- Mettre à jour des données :

```
UPDATE utilisateurs SET nom = 'Alicia' WHERE id = 1;
```

- Supprimer des données :

```
DELETE FROM utilisateurs WHERE id = 1;
```

5. Opérations avancées

- Jointures : Relient des données de plusieurs tables via des clés étrangères.
- GROUP BY, ORDER BY : Pour regrouper ou trier les résultats.
- Vues : Requêtes enregistrées, utilisées comme des tables virtuelles.
- Triggers : Exécutent automatiquement du code SQL lors d'événements (insertion, mise à jour, suppression).

6. Sauvegarde et restauration

- Sauvegarde :
  En ligne de commande, utilisez `.backup` pour copier la base vers un autre fichier.
- Restauration :
  Utilisez `.restore` pour charger une sauvegarde dans une base existante.

7. Bonnes pratiques et astuces

- Utilisez les index sur les colonnes fréquemment recherchées pour accélérer les requêtes.
- Sauvegardez régulièrement votre fichier de base de données.
- Évitez de modifier directement le fichier sur plusieurs systèmes simultanément pour prévenir la corruption.
- Exploitez la portabilité : copiez le fichier où vous voulez, il fonctionnera sur tout système supportant SQLite.

8. Ressources pour progresser

- Documentation officielle SQLite
- Tutoriels en ligne, forums et communautés
- Outils graphiques comme DB Browser for SQLite pour manipuler vos bases facilement

Astuce : SQLite est idéal pour les projets nécessitant une base de données simple, rapide à mettre en œuvre et portable. Testez vos requêtes, explorez les possibilités des transactions et des triggers, et profitez de la simplicité de gestion d'un fichier unique !