BDDR - SQLite - Requêtes dans la console

[Mise à jour le 1/3/2025]

- **Ressources**
  - Le langage SQL
  - How To Download & Install SQLite Tools
  - Command Line Shell For SQLite
- **Lectures connexes**
  - Wiki - BDDR : généralités
  - Wiki - BDDR - MySQL - Requêtes dans la console
  - Wiki - BDDR - SQLite dans VSCode
- **Mots-clés** : SGBDR, base, table, enregistrement, champ, requêtes, SQL.

---

1. Présentation

SQLite est une bibliothèque en **langage C** qui implémente un petit moteur de base de données **SQL** complet, rapide, autonome et très fiable. C'est le moteur de base de données le plus utilisé au monde. Il est intégré à tous les téléphones mobiles et à la plupart des ordinateurs ainsi que dans d'innombrables applications que nous utilisons tous les jours.

Le code source de SQLite est dans le domaine public. Voir le **lien** suivant pour **installer SQLite3**.

2. Prise en main rapide

> **REMARQUE**
>
> Sur un PC sous **Windows**, l'utilisation de la ligne de commande peut se faire dans **n'importe quel répertoire** du moment que le chemin vers le fichier **sqlite3.exe** est dans les **variables d'environnement**.
>
> Sous **Linux** ce réglage est fait par défaut.

2.1 Connexion à une base ou création

- **Ressource** : SQLITE tutorial - Commandes

Le projet SQLite fournit un **programme de ligne de commande simple nommé sqlite3** (ou sqlite3.exe sous Windows) qui permet à l'utilisateur de saisir et d'exécuter manuellement des instructions SQL sur une base de données SQLite ou sur une archive ZIP .

SQLite3, .open et .database

- **Cas 1** : la console SQLite **n'est pas activée** et aucune base n'a été créée
  Démarer la console sqlite3 en entrant : **sqlite3** *nom_base*.**db** pour ouvrir la console et créer une base.
  ATTENTION : si aucun fichier de base de données n'est spécifié sur la ligne de commande, une **base de données temporaire** est créée et **automatiquement supprimée** lorsque le programme « sqlite3 » se ferme.

- **Cas 2** : la console SQLite **n'est pas activée** mais **la base existe**.
  Démarer la console sqlite3 en entrant : **sqlite3**

- **Cas 3** : la console SQLite **est activée**.
  Entrer **.open** *nom_base.db* (si la base n'existe pas elle sera créée).
  Remarque : entrer **.database** pour afficher toutes les bases de données dans la connexion actuelle.

Exemple : création de la base *cine* sur un Raspberry pi et connexion

*.bash

```
login@machine:~/tpbdd $ sqlite3 cine.db
```

*.sql

```
sqlite>.open cine.db
sqlite> .databases -- Résultat sur un Raspberry Pi -> main:
/home/login/tpbdd/cine.db r/w
```

On termine le programme sqlite3 en saisissant **Ctrl-D**.

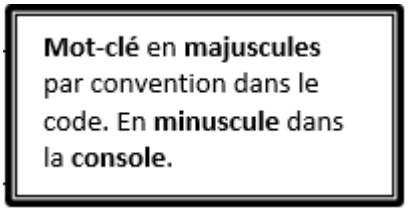
CREATE TABLE

2.2 Création et suppression d'une table

- **Ressource** : SQLITE tutorial - CREATE TABLE
- **Création d'une table**

> Il est nécessaire de définir pour chaque champ,son **nom** et son **type** lors de la création d'une table, .

Exemple : création de la table *acteurs*.



*.sql

```
CREATE TABLE acteurs(
id INTEGER PRIMARY KEY AUTOINCREMENT,
prenom VARCHAR(30),
nom VARCHAR(30),
nationalite VARCHAR(6),
naissance DATE
);
```

> **.schema et .tables**
> La commande **.schema** renvoie le schéma de la table. La commande **.tables** renvoie la liste de toutes les tables présentes dans la base.

- **Suppression d'une table**

*.sql

```
-- Exemple : suppression de la table //acteurs//.
DROP TABLE acteurs;
```

INSERT INTO

2.3 Insertion d'enregistrements avec INSERT INTO

- **Ressource** : SQLITE tutorial - INSERT INTO

> On peut enregistrer une valeur à la fois ou plusieurs séparées par des virgules.

Exemple : ajout d'un acteur dans la table.

*.sql

```
INSERT INTO acteurs (prenom, nom, nationalite, naissance)
VALUES ('Keanu', 'Reeves', 'ca', '1964-07-02');
```

SELECT ... FROM

2.4 Extraction de données avec SELECT

- **Ressource** : SQLITE tutorial - SELECT

> On peut sélectionner une ou plusieurs colonnes, données en paramètre. L'étoile indique que l'on sélectionne toutes les colonnes de la table.

Exemple : consultation de la table *acteur*.

*.sql

```
SELECT * FROM acteurs;

-- Résultat
1|Keanu|Reeves|ca|1964-07-02
```

> **.headers, .mode**
>
> Pour améliorer la présentation, on utilise les commandes:
> **.headers ON** et
> **.mode column**

Exemple

*.sql

```
SELECT * FROM acteurs;

-- Résultat
-- id  prenom  nom     nationalite  naissance
-- --  ------  ------  -----------  ----------
-- 1   Keanu   Reeves  ca           1964-07-02
```

2.5 Sortir de SQLite

- **Ressource** : SQLITE tutorial - Commandes

Entrer la combinaison de touches **Ctrl-C** sous Windows ou **.quit** sous Linux

3. Utiliser un fichier d'extension .sql

- **Ressource** : SQLite Transaction

Fichier d'extension

On peut créer un fichier d'extension sql pour “plus de confort”. Les requêtes sont placées entre
BEGIN TRANSACTION;
et
COMMIT;

Exemple : création du fichier *requetes.sql* dans le même répertoire que *cinema.db* pour ajouter un acteur à la table.

*.sql

```
BEGIN TRANSACTION;
INSERT INTO acteurs (prenom, nom, nationalite, naissance)
VALUES('Christopher','Walken','us','1943-03-31');
COMMIT;
```

> Ce fichier est “appliqué” à la base avec la commande : **sqlite3 cinema.db < requetes.sql**.

On se connecte ensuite à *cinema.db* pour consulter la table *acteur*.

Pour aller plus loin ...

Voir SQL et SQLite dans VSCode et SQLite Tutorial