Le langage SQL

1. Introduction

Le langage le plus communément utilisé pour dialoguer (effectuer des requêtes) avec le SGBD est le **SQL** (Structured Query Language : langage de requête structurée).
Il permet de créer des tables, ajouter, supprimer, mettre à jour des données.
Il a été créé en 1979 et normalisé en1986.

2. SQLite

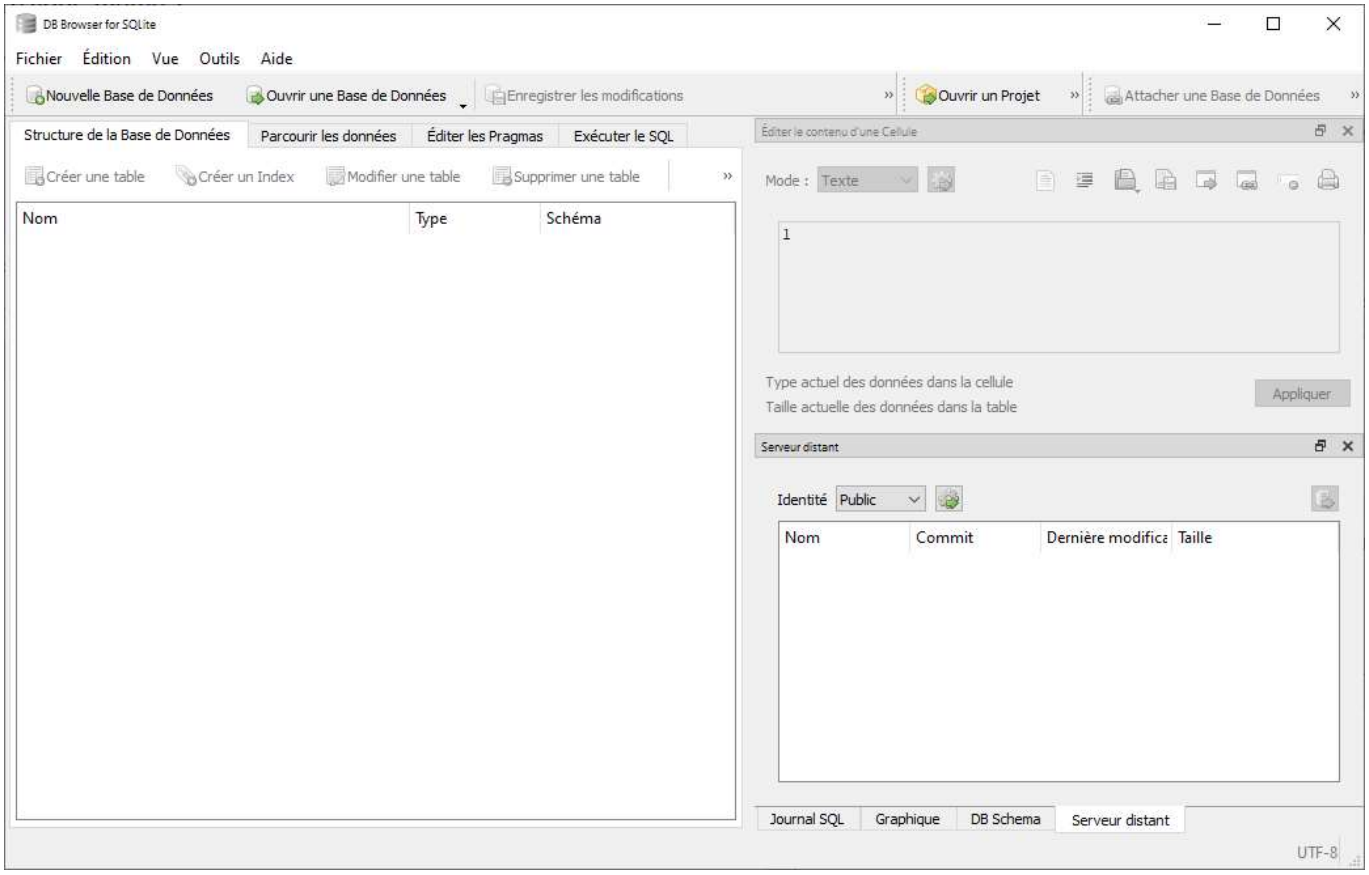
La plupart de SGBD fonctionnement sur des machines distantes selon le modèle client-serveur bien adapté à un langage de requêtes. SQLite, comme son nom l'indique, est un SGBD qui a un fonctionnement plus simple. SQLite est un système de gestion de base de données relationnelle très répandu.

- SQLite est disponible sur tous les systèmes d'exploitation. Il existe une bibliothèque Python, **`sqlite3`**, qui permet de gérer la BDD à partir de commandes écrites en Python.
- On peut aussi l'utiliser avec l'outil graphique DB Browser for SQLite (https://sqlitebrowser.org/).

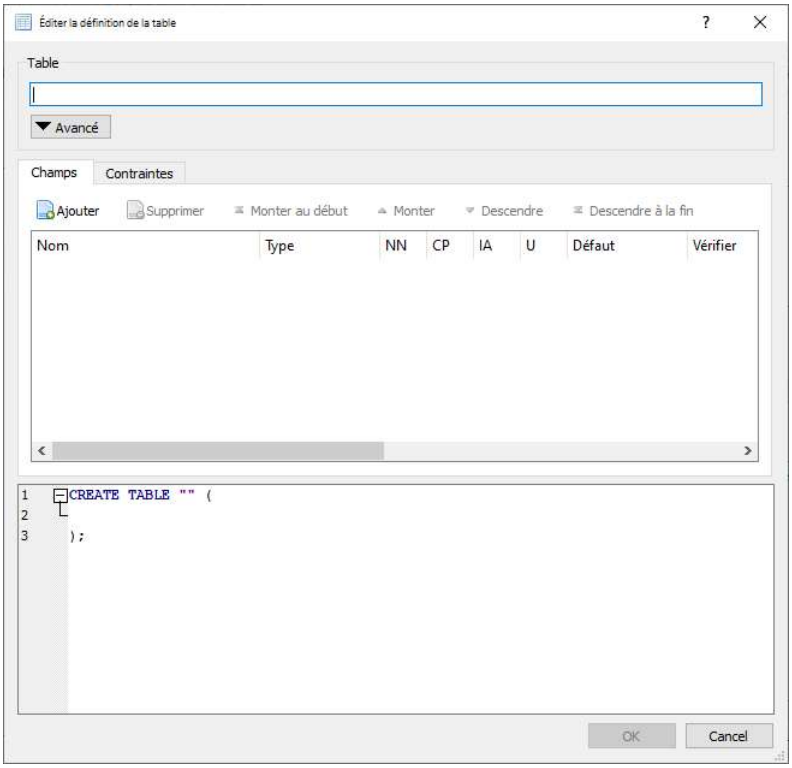
3. Création d'une base de données

Pour créer une base de données et effectuer des requêtes sur cette dernière, nous allons utiliser le logiciel DB Browser for SQLite : https://sqlitebrowser.org/

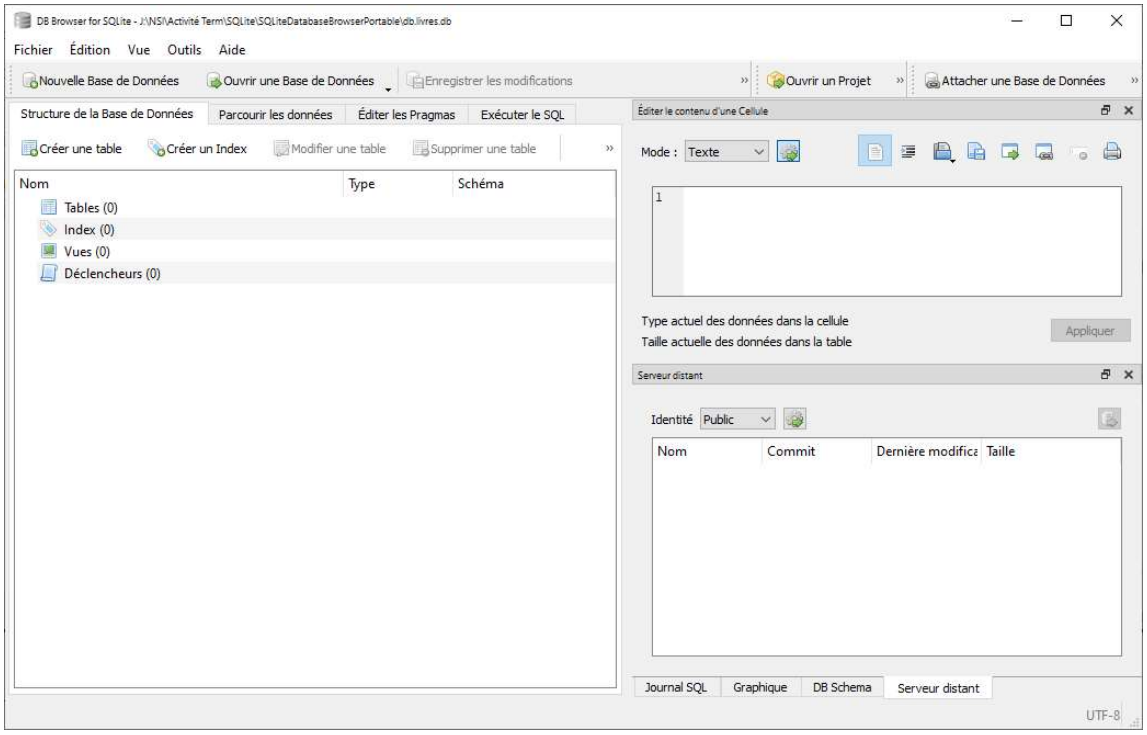
✍ Après avoir installé la version portable sur votre clé USB, lancez le logiciel, vous devriez obtenir ceci :



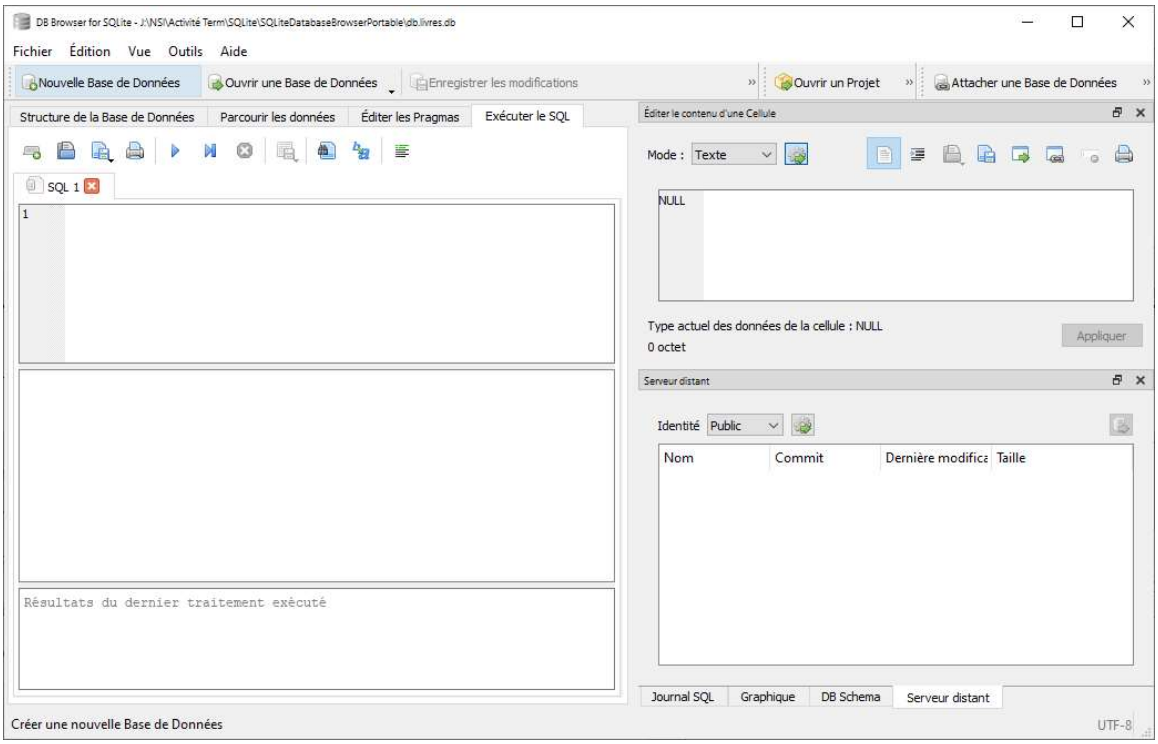
✍ Cliquer sur **Nouvelle Base de Données**. Choisir un nom pour votre base de données (**db_livres.db**).
✍ La fenêtre ci-dessous apparaît, cliquer sur **Cancel**.



La base de données est créée mais elle ne contient aucune table (relation).



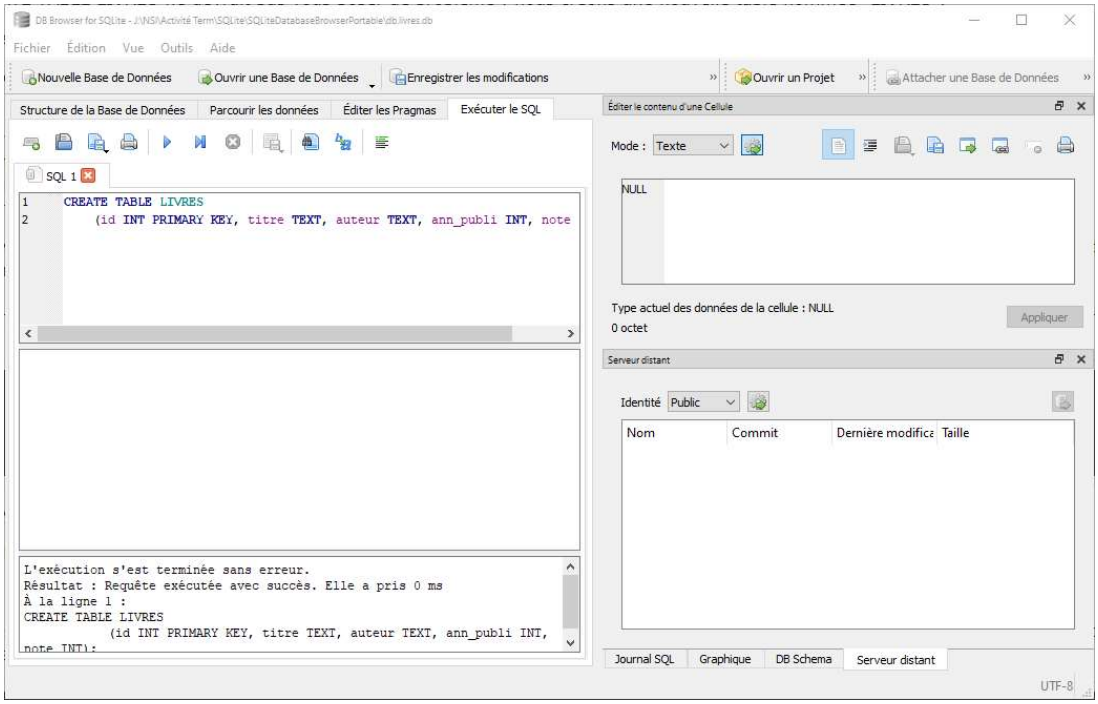
✍ Pour créer une table, cliquer sur l'onglet **Exécuter le SQL** pour obtenir ceci :



✍ Dans la fenêtre SQL 1, écrire le texte ci-dessous :

```
CREATE TABLE LIVRES
      (id INT PRIMARY KEY, titre TEXT, auteur TEXT, ann_publi INT, note INT);
```

✍ Puis cliquer sur le triangle bleu (ou touche F5) afin d'effectuer la requête.
Le triangle bleu (MAJ + F5) avec une barre permet d'effectuer seulement la requête courante (où se situe le curseur) et par conséquent de garder écrite les requêtes précédentes et d'enregistrer un seul fichier SQL.

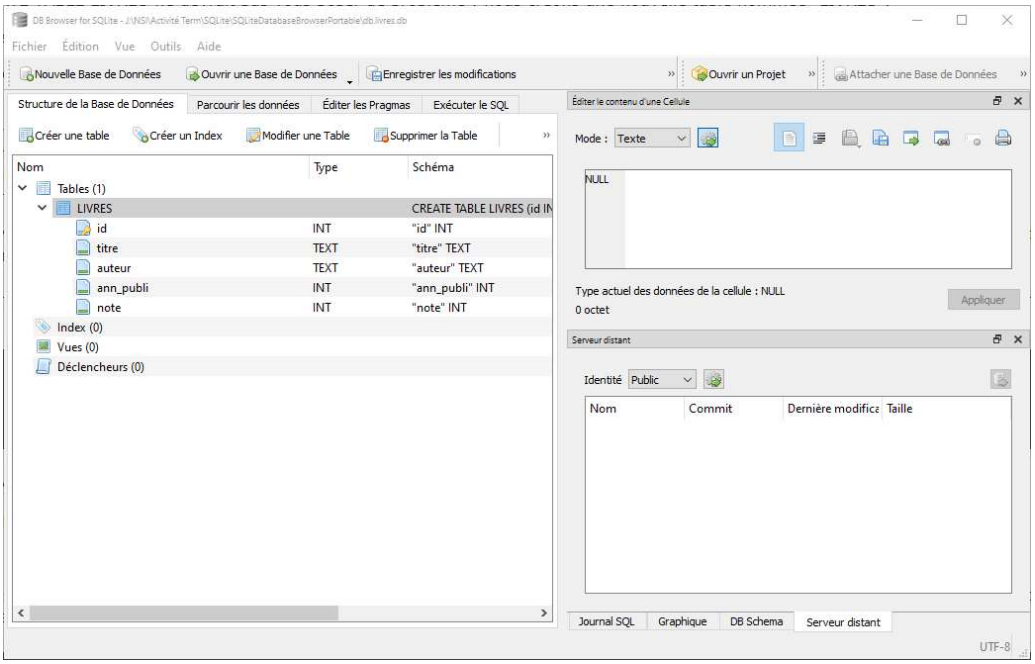


L'instruction **`CREATE TABLE LIVRES`** a permis de créer une table nommée **`LIVRES`**.
L'instruction **`id INT PRIMARY KEY, titre TEXT, auteur TEXT, ann_publi INT, note INT`** a permis de créer les attributs :

- id
- titre,
- auteur
- ann_publi,
- note

Pour chaque attribut le domaine est précisé : **`id`** entier (**`INT`**), **`titre`** : chaîne de caractères (**`TEXT`**)…
L'attribut **`id`** est aussi la clé primaire (**`PRIMARY KEY`**). Ainsi l'attribut **`id`** ne pourra pas comporter deux fois la même valeur.

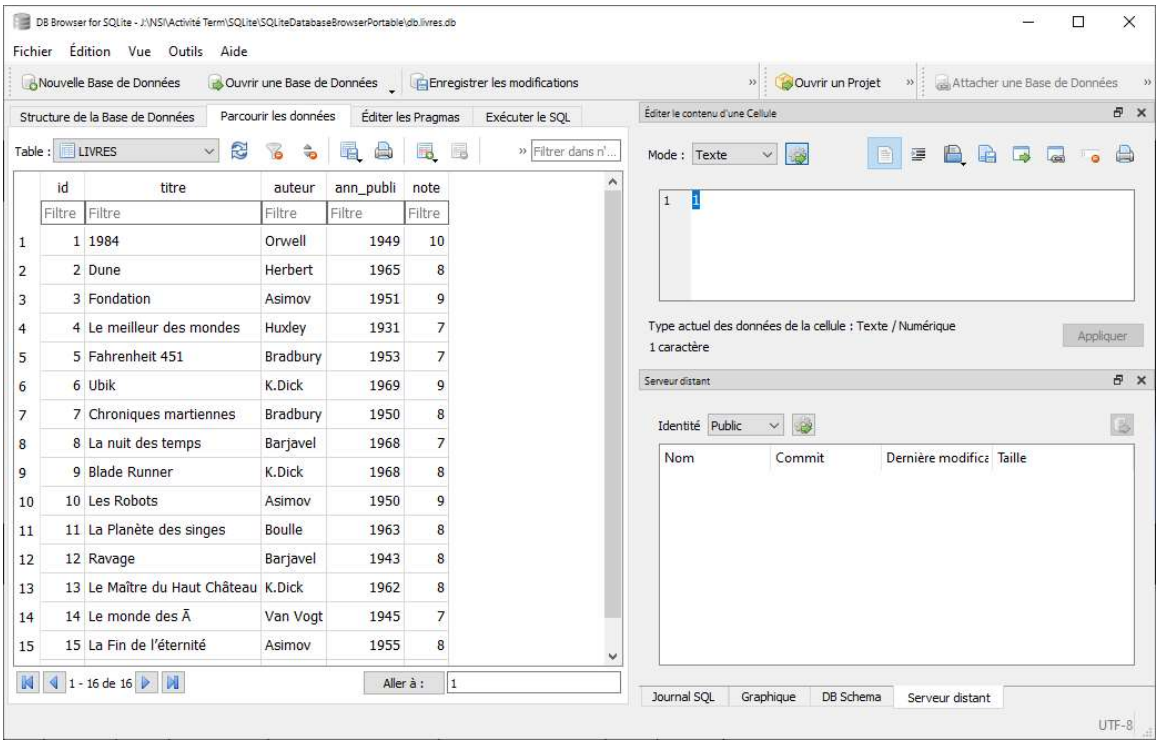
✍ L'onglet **Structure de la Base de Données** permet d'observer le détail de la table **LIVRES**.



✍ Compléter la table **LIVRES** en exécutant la requête suivante (onglet **Exécuter le SQL**).

```
INSERT INTO LIVRES
      (id,titre,auteur,ann_publi,note)
      VALUES
      (1,'1984','Orwell',1949,10),
      (2,'Dune','Herbert',1965,8),
      (3,'Fondation','Asimov',1951,9),
      (4,'Le meilleur des mondes','Huxley',1931,7),
      (5,'Fahrenheit 451','Bradbury',1953,7),
      (6,'Ubik','K.Dick',1969,9),
      (7,'Chroniques martiennes','Bradbury',1950,8),
      (8,'La nuit des temps','Barjavel',1968,7),
      (9,'Blade Runner','K.Dick',1968,8),
      (10,'Les Robots','Asimov',1950,9),
      (11,'La Planète des singes','Boulle',1963,8),
      (12,'Ravage','Barjavel',1943,8),
      (13,'Le Maître du Haut Château','K.Dick',1962,8),
      (14,'Le monde des Ā','Van Vogt',1945,7),
      (15,'La Fin de l'éternité','Asimov',1955,8),
      (16,'De la Terre à la Lune','Verne',1865,10);
```

✍ Les données de la table **LIVRES** sont visibles dans l'onglet **Parcourir les données**.

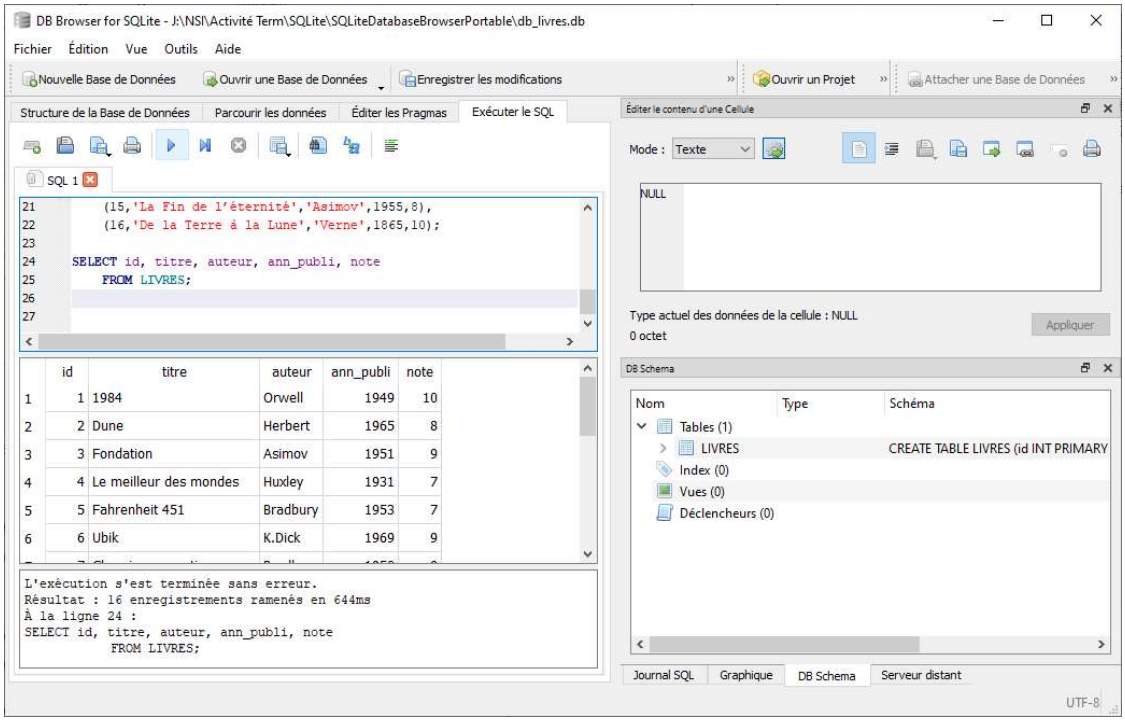


4. Interroger une base de données

✍ Saisir la requête SQL suivante :

```
SELECT id, titre, auteur, ann_publi, note
      FROM LIVRES;
```

Le résultat doit être le suivant :



La requête SQL a permis d'afficher tous les livres.
SELECT permet de sélectionner les attributs et **FROM** indique la table qui doit être utilisée.

✍ En saisissant la requête suivante, vérifier que seulement les attributs titre et auteur sont affichés.

```
SELECT titre, auteur
     FROM LIVRES;
```

✍ Pour afficher tous les attributs, comme précédemment, on peut écrire :

```
SELECT *
     FROM LIVRES;
```

Actuellement les requêtes affichent tous les livres, il est possible d'utiliser la clause WHERE afin d'imposer une ou plusieurs conditions permettant de sélectionner uniquement certaines lignes.
La condition doit suivre le mot clé WHERE.
✍ Saisir et tester la requête suivante :

```
SELECT titre, ann_publi
     FROM LIVRES
     WHERE auteur='Asimov';
```

✍ Vérifier que seul les livres écrit par Isaac Asimov sont affichés.

Il est possible de combiner les conditions à l'aide d'un OR ou d'un AND.
✍ Saisir et tester la requête suivante :

```
SELECT titre, ann_publi
     FROM LIVRES
     WHERE auteur='Asimov' AND ann_publi>1953;
```

Seul le livre écrit après 1953 est affiché. Les opérateurs d'inégalités peuvent aussi être employés (**ann_publi>1953**).

Il est aussi possible de rajouter la clause SQL ORDER BY afin d'obtenir les résultats classés dans un ordre précis.
✍ Saisir et tester la requête suivante :

```
SELECT titre
     FROM LIVRES
     WHERE auteur='K.Dick' ORDER BY ann_publi;
```

Les livres de K.Dick sont classés du plus ancien au plus récent.

Il est possible d'obtenir un classement en sens inverse à l'aide de la clause DESC.
✍ Saisir et tester la requête suivante :

```
SELECT titre
     FROM LIVRES
     WHERE auteur='K.Dick' ORDER BY ann_publi DESC;
```

Lorsque l'attribut est du type TEXT, le classement se fait par ordre alphabétique.
✍ Saisir et tester la requête suivante :

```
SELECT titre
     FROM LIVRES
     WHERE auteur='K.Dick' ORDER BY titre;
```

✍ Saisir et tester la requête suivante :

```
SELECT auteur
     FROM LIVRES;
```

La requête précédente affiche plusieurs fois certains auteurs (les auteurs qui ont écrit plusieurs livres présents sans la base de données).
Il est possible d'éviter les doublons grâce à la clause DISTINCT.
✍ Saisir et tester la requête suivante :

```
SELECT DISTINCT auteur
    FROM LIVRES;
```

5. Association de deux tables

✍ Créer une nouvelle base de données nommée **db_livres_auteurs.db**.

✍ Créer une table AUTEURS à l'aide de la requête suivante :

```
CREATE TABLE AUTEURS
    (id INT PRIMARY KEY, nom TEXT, prenom TEXT, ann_naissance INT, langue_ecriture TEXT);
```

✍ Créer la table LIVRES à l'aide de la requête suivante :

```
CREATE TABLE LIVRES
    (id INT PRIMARY KEY, titre TEXT, id_auteur INT, ann_publi INT, note INT, FOREIGN KEY
    (id_auteur) REFERENCES AUTEURS(id));
```

id_auteur est une clé étrangère.

✍ Ajouter des données à la table AUTEURS à l'aide la requête suivante :

```
INSERT INTO AUTEURS
    (id,nom,prenom,ann_naissance,langue_ecriture)
    VALUES
        (1,'Orwell','George',1903,'anglais'),
        (2,'Herbert','Frank',1920,'anglais'),
        (3,'Asimov','Isaac',1920,'anglais'),
        (4,'Huxley','Aldous',1894,'anglais'),
        (5,'Bradbury','Ray',1920,'anglais'),
        (6,'K.Dick','Philip',1928,'anglais'),
        (7,'Barjavel','René',1911,'français'),
        (8,'Boulle','Pierre',1912,'français'),
        (9,'Van Vogt','Alfred Elton',1912,'anglais'),
        (10,'Verne','Jules',1828,'français');
```

✍ Ajouter des données à la table LIVRES à l'aide la requête suivante :

```
INSERT INTO LIVRES
    (id,titre,id_auteur,ann_publi,note)
    VALUES
        (1,'1984',1,1949,10),
        (2,'Dune',2,1965,8),
        (3,'Fondation',3,1951,9),
        (4,'Le meilleur des mondes',4,1931,7),
        (5,'Fahrenheit 451',5,1953,7),
        (6,'Ubik',6,1969,9),
        (7,'Chroniques martiennes',5,1950,8),
        (8,'La nuit des temps',7,1968,7),
        (9,'Blade Runner',6,1968,8),
        (10,'Les Robots',3,1950,9),
        (11,'La Planète des singes',8,1963,8),
        (12,'Ravage',7,1943,8),
        (13,'Le Maître du Haut Château',6,1962,8),
        (14,'Le monde des Ā',9,1945,7),
        (15,'La Fin de l'éternité',3,1955,8),
        (16,'De la Terre à la Lune',10,1865,10);
```

Grâce aux jointures, il est possible d'associer ces deux tables dans une même requête. Les jointures consistent à associer des lignes de deux tables.

✍ Saisir et tester la requête suivante :

```
SELECT *
      FROM LIVRES
      JOIN AUTEURS ON LIVRES.id_auteur = AUTEURS.id;
```

Le **FROM LIVRES JOIN AUTEURS** permet de créer une jointure entre les tables LIVRES et AUTEURS (rassembler les tables LIVRES et AUTEURS en une seule grande table). Le **ON LIVRES.id_auteur = AUTEURS.id** signifie qu'une ligne quelconque A de la table LIVRES devra être fusionnée avec la ligne B de la table AUTEURS à condition que l'attribut **id_auteur** de la ligne A soit égal à l'attribut **id** de la ligne B.

Par exemple, la ligne 1 (**id**=1) de la table LIVRES (que l'on nommera dans la suite ligne A) sera fusionnée avec la ligne 1 (**id**=1) de la table AUTEURS (que l'on nommera dans la suite B) car l'attribut **id_auteur** de la ligne A est égal à 1 et l'attribut **id** de la ligne B est aussi égal à 1.

Autre exemple, la ligne 1 (**id**=1) de la table LIVRES (que l'on nommera dans la suite ligne A) ne sera pas fusionnée avec la ligne 2 (**id**=2) de la table AUTEURS (que l'on nommera dans la suite B') car l'attribut **id_auteur** de la ligne A est égal à 1 alors que l'attribut **id** de la ligne B' est égal à 2.

✍ Saisir et tester la requête suivante :

```
SELECT *
      FROM AUTEURS
      JOIN LIVRES ON LIVRES.id_auteur = AUTEURS.id;
```

Le résultat est différent, cette fois-ci ce sont les lignes de la table LIVRES qui viennent se greffer sur la table AUTEURS.

✍ Saisir et tester la requête suivante :

```
SELECT nom, prenom, titre
      FROM AUTEURS
      JOIN LIVRES ON LIVRES.id_auteur = AUTEURS.id;
```

✍ Saisir et tester la requête suivante :

```
SELECT titre, nom, prenom
      FROM LIVRES
      JOIN AUTEURS ON LIVRES.id_auteur = AUTEURS.id;
```

Si un même nom d'attribut est présent dans les deux tables (par exemple ici l'attribut **id**), il est nécessaire d'ajouter le nom de la table devant afin de pouvoir les distinguer (**AUTEURS.id** et **LIVRES.id**).

✍ Saisir et tester la requête suivante :

```
SELECT titre, AUTEURS.id, nom, prenom
      FROM LIVRES
      JOIN AUTEURS ON LIVRES.id_auteur = AUTEURS.id;
```

Il est possible d'utiliser la clause WHERE dans le cas d'une jointure.

✍ Saisir et tester la requête suivante :

```
SELECT titre, nom, prenom
      FROM LIVRES
      JOIN AUTEURS ON LIVRES.id_auteur = AUTEURS.id
      WHERE ann_publi > 1950;
```

6. Modifier une base de données

6.1. Insérer un nouvel enregistrement

✍ Insérer un nouvel enregistrement dans la table **LIVRES** avec l'instruction ci-dessous et vérifier sa présence dans l'onglet **Parcourir les données**. Rajouter son auteur si besoin.

```
INSERT INTO LIVRES
      (id, titre, id_auteur, ann_publi, note)
      VALUES
      (17,'Hypérion',11,1991,10);
```

6.2. Modifier un enregistrement

✍ Modifier la note pour le titre **Hypérion** et vérifier la modification.

```
UPDATE LIVRES
      SET note=7
      WHERE titre = 'Hypérion';
```

Ici les instructions permettent de mettre (**SET**) la note à 7 (**note = 7**) où (**WHERE**) se situe l'enregistrement qui contient le titre Hyperion (**titre = 'Hypérion'**). La mise à jour (**UPDATE**) s'effectue dans la table **LIVRES**.

6.3. Effacer un enregistrement

✍ Effacer l'enregistrement **Hypérion** et vérifier la modification.

```
DELETE FROM LIVRES
      WHERE titre='Hypérion';
```

Ici les instructions permettent d'effacer (**DELETE**) dans la table **LIVRES** la ligne qui contient le titre Hyspérion (**titre='Hypérion'**).

7. Filtrer

L'instruction **LIKE** permet de réaliser un filtre sur un motif de chaîne recherché.

- Un _ pour un caractère quelconque et **%** pour aucun ou plusieurs.

Il est ainsi possible d'afficher tous les titres qui contiennent le mot **monde**.

✍ Saisir et tester la requête suivante :

```
SELECT titre
      FROM LIVRES
      WHERE titre LIKE '%monde%';
```

Il est possible de définir un intervalle de recherche avec **BETWEEN**.

✍ Saisir et tester la requête suivante afin d'afficher tous les livres parus entre 1950 et 1959 :

```
SELECT titre
      FROM LIVRES
      WHERE ann_publi BETWEEN 1950 AND 1959;
```

8. Compter

Parfois lors d'une requête le nombre de résultats peut être important et il est souvent intéressant de demander combien il y a de réponses sans les regarder.

✍ Saisir et tester la requête suivante afin de savoir le nombre de livres qui ont une note de 7 :

```
SELECT COUNT(titre)
      FROM LIVRES
      WHERE note = 7;
```

Source : https://pixees.fr/informatiquelycee/n_site/nsi_term_bd_sql.html