3_Le_langage_Sql

September 3, 2020

1 TD base de données

Nous avons eu l'occasion d'étudier la structure d'une base de données relationnelle, nous allons maintenant apprendre à réaliser des requêtes.

c'est-à-dire que nous allons apprendre à :

- Créer une base de données,
- Créer des attributs,
- Ajouter de données,
- Modifier des données
- Interroger une base de données afin d'obtenir des informations.

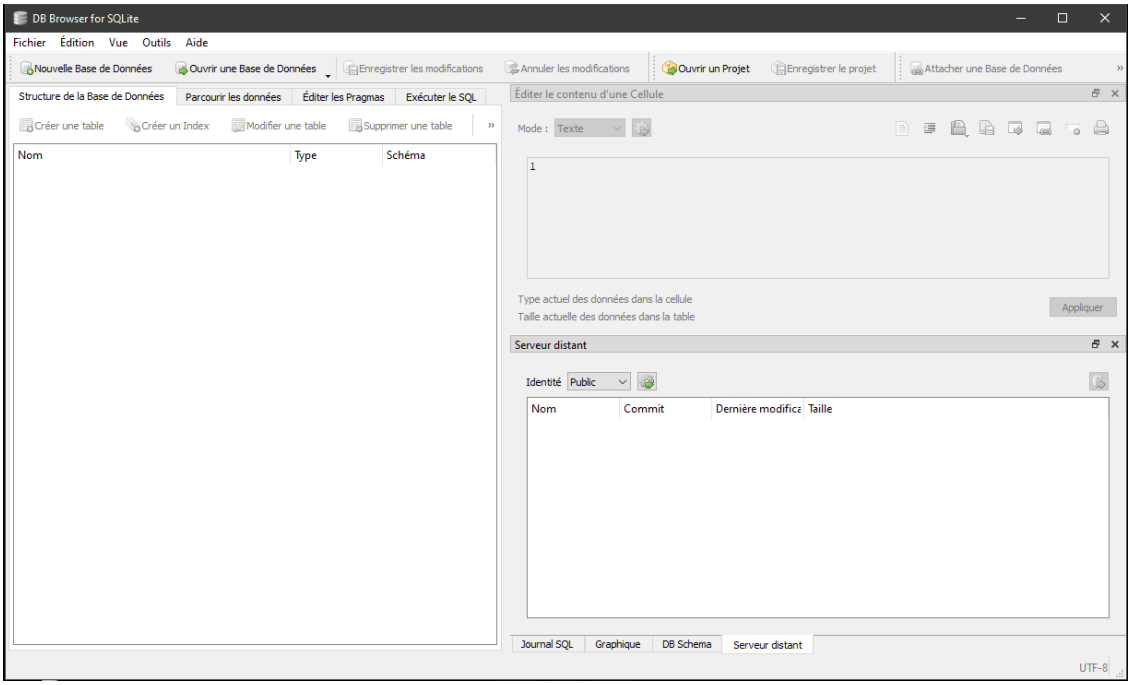
Pour réaliser toutes ces requêtes, nous allons devoir apprendre un langage de requêtes : SQL (Structured Query Language). SQL est propre aux bases de données relationnelles, les autres types de bases de données utilisent d'autres langages pour effectuer des requêtes.

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/.

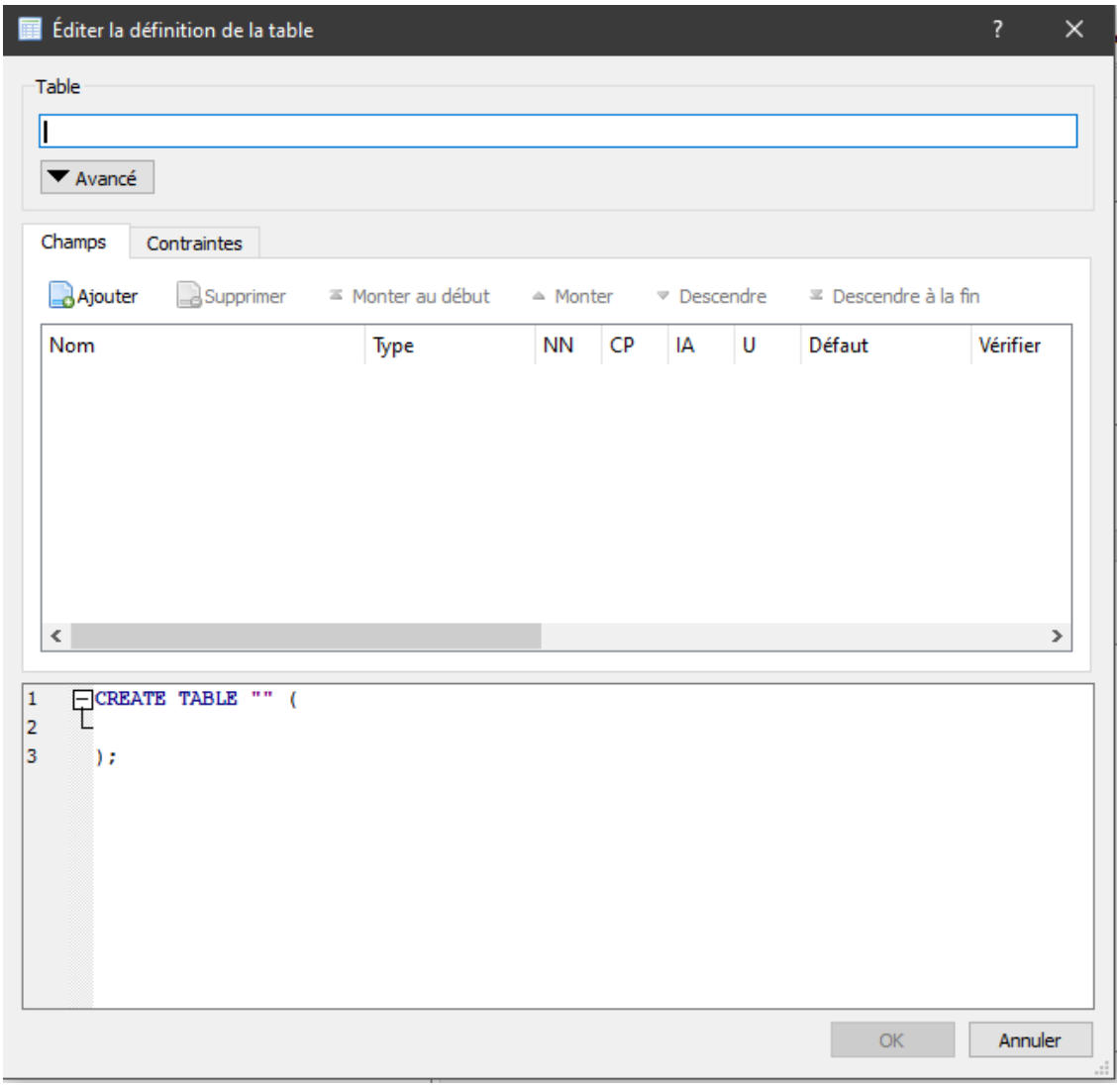
SQLite est un système de gestion de base de données relationnelle très répandu. Il existe d'autres systèmes de gestion de base de données relationnelle comme MySQL ou PostgreSQL.Dans tous les cas, le langage de requête utilisé est le SQL (même si parfois on peut noter quelques petites différences).Ce qui sera vu ici avec SQLite pourra, à quelques petites modifications près, être utilisé avec, par exemple, MySQL.

2 Création de la base de données

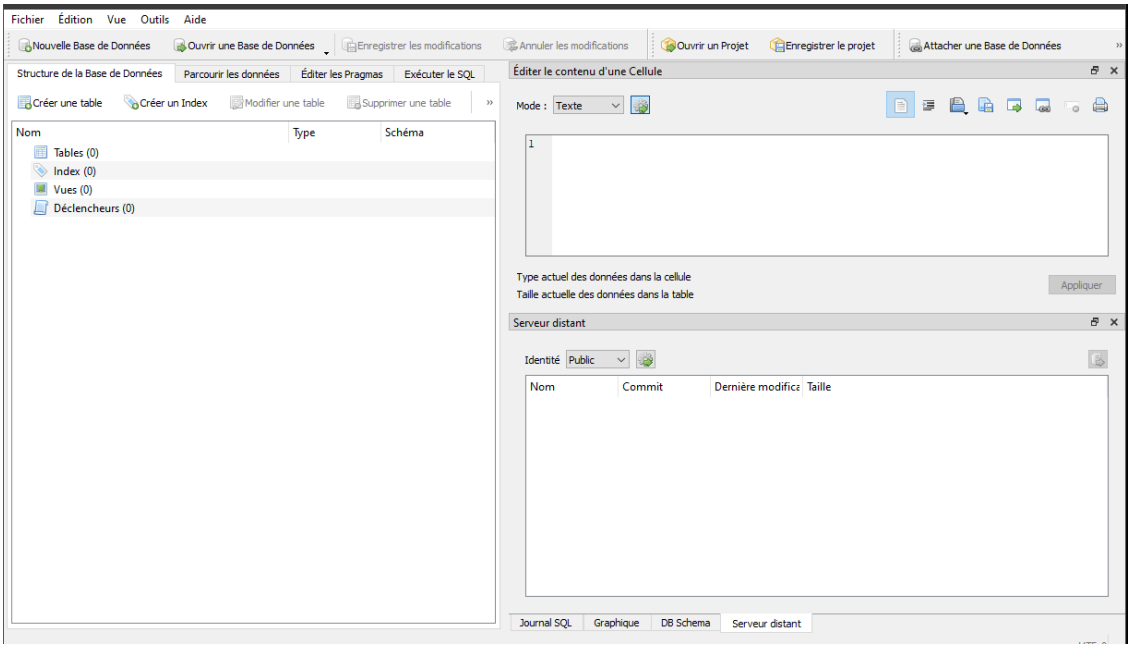
Après avoir lancé le logiciel "DB Browser for SQLite", vous obtenez ceci :



Cliquez sur Nouvelle base de données. Vous nommerez votre base de données “db_livres.db” et choissirez de l’enregistrer sur le bureau, vous devriez avoir la fenêtre ci-dessous que vous fermerez :



Notre base de données a été créée :

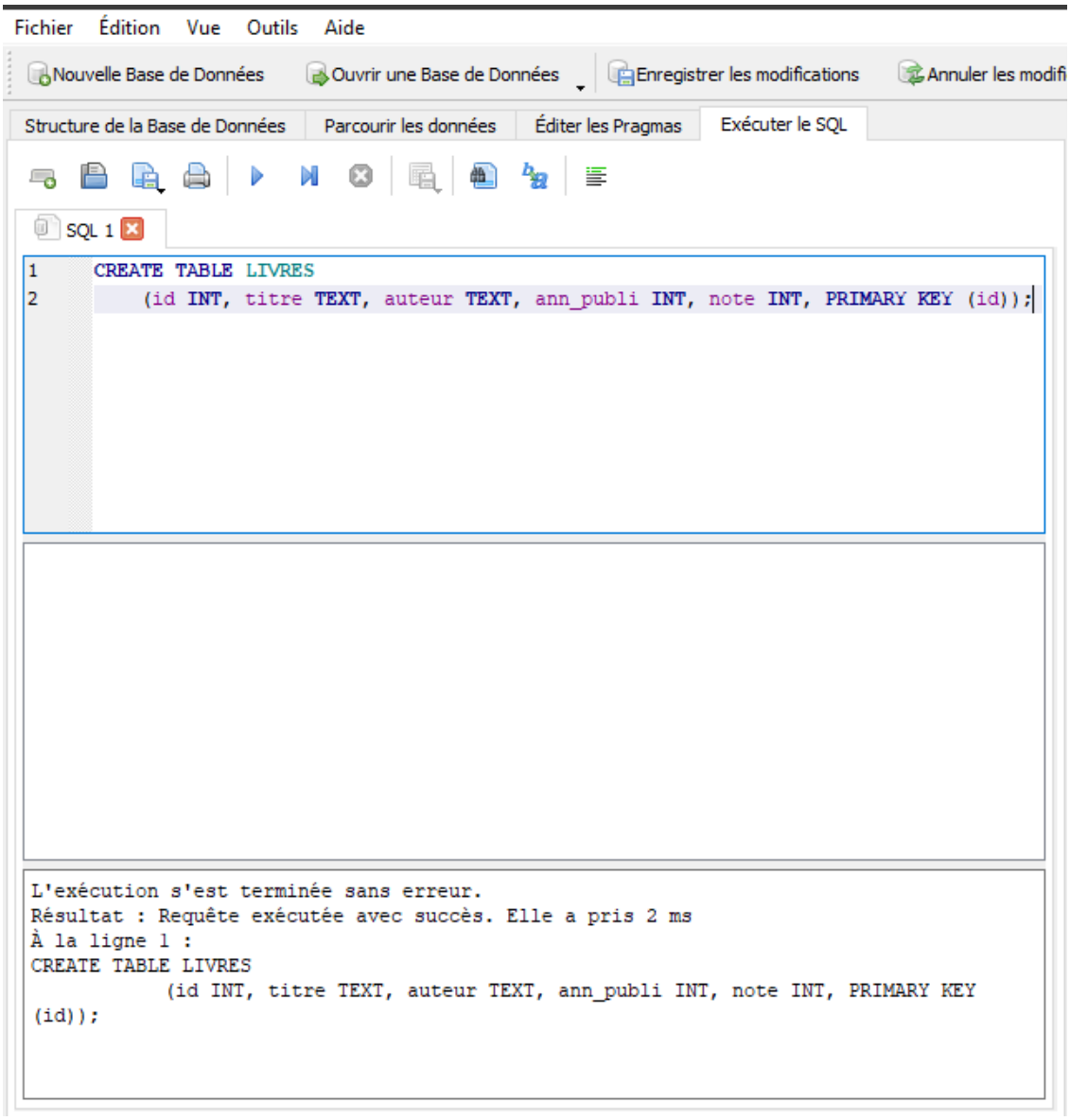


3 Création des attributs

Pour l'instant notre base est vide et ne contient aucune table (aucune relation), pour créer une table, cliquez sur l'onglet "Exécuter le SQL".

Copiez-collez le texte ci dessous dans la fenêtre "SQL 1": CREATE TABLE LIVRES (id INT, titre TEXT, auteur TEXT, ann_publi INT, note INT, PRIMARY KEY (id));

Puis exécuter la commande et cliquant sur la flèche bleu.



Comme indiqué en bas dans la fenêtre, “Requête exécutée avec succès” !

Vous venez de créer votre première table avec les attributs :

- id
- titre
- auteur
- ann_pulbi
- note

Nous avons pour chaque attribut précisé son domaine :

- id : entier (INT)
- titre : chaîne de caractères (TEXT)
- auteur : chaîne de caractères
- ann_publi : entier

- note : entier

L'attribut "id" va jouer ici le rôle de clé primaire. Par souci de sécurité (afin d'éviter que l'on utilise 2 fois la même valeur pour l'attribut "id"), l'instruction SQL "PRIMARY KEY (id)" précise que l'attribut "id" est bien notre clé primaire.

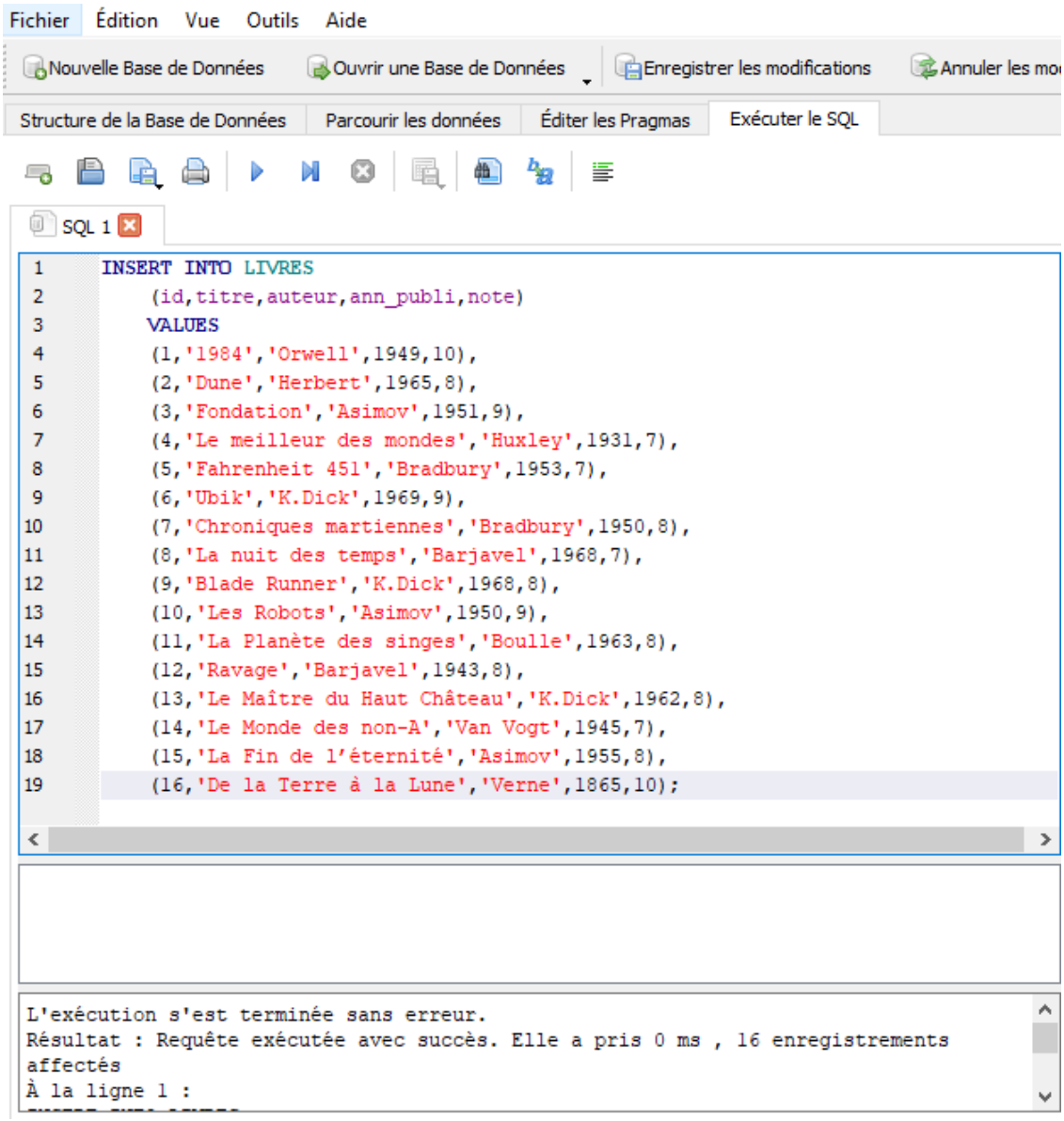
4 Ajout de données

Maintenant que la table est crée, nous allons ajouter des données :

Copiez -collez la requête ci dessous dans l'onglet "Exécuter le SQL".

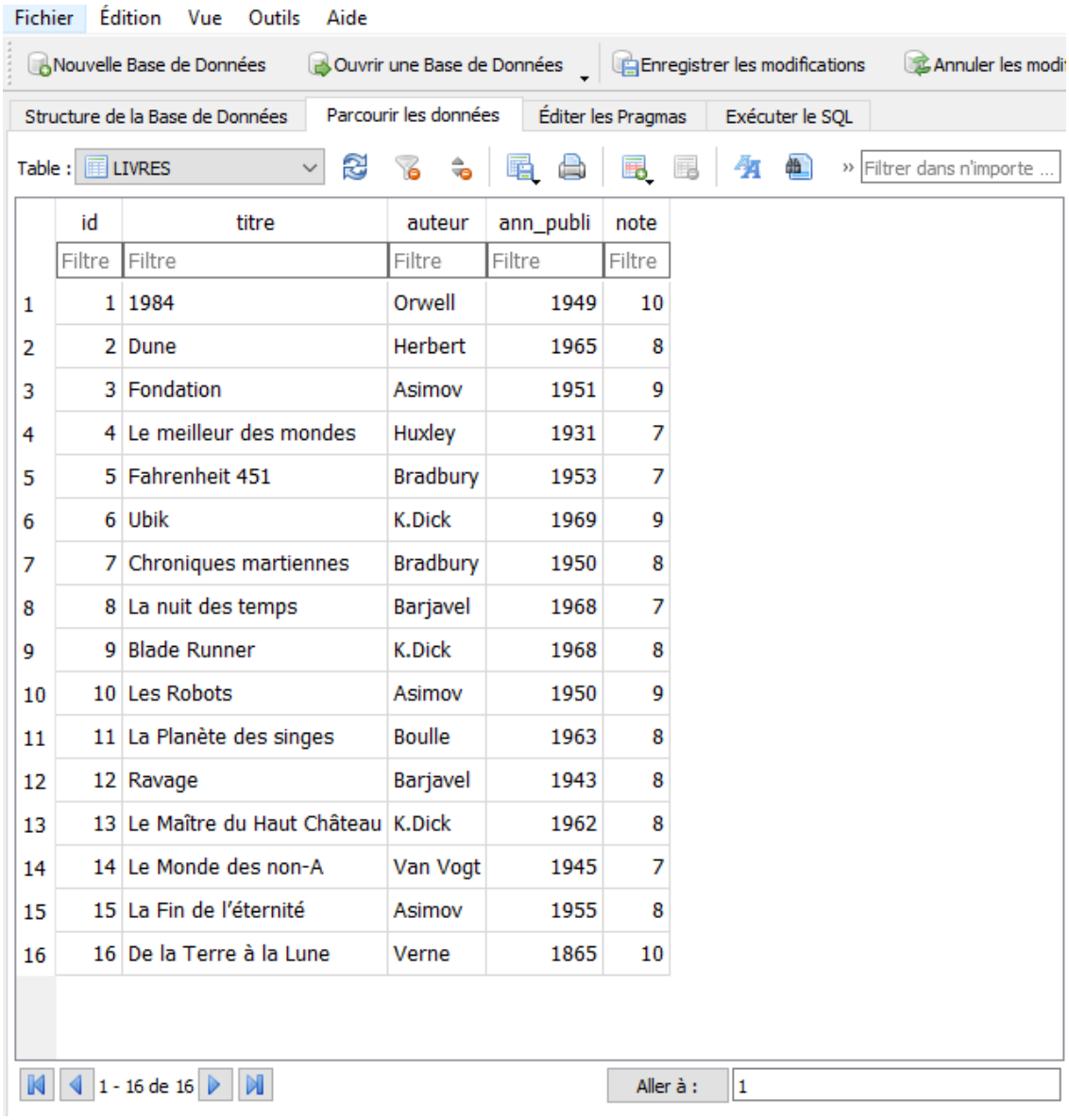
```
INSERT INTO LIVRES
    (id,titre,auteur,ann_publi,note)
VALUES
    (1,'1984','Orwell',1949,10),
    (2,'Dune','Herbert',1965,8),
    (3,'Fondation','Asimov',1951,9),
    (4,'Le meilleur des mondes','Huxley',1931,7),
    (5,'Fahrenheit 451','Bradbury',1953,7),
    (6,'Ubik','K.Dick',1969,9),
    (7,'Chroniques 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 non-A','Van Vogt',1945,7),
    (15,'La Fin de l'éternité','Asimov',1955,8),
    (16,'De la Terre à la Lune','Verne',1865,10);
```

et comme précédemment, cliquez sur la flèche bleu pour éxécuter la commande.



Ici aussi, aucun problème, la requête a bien été exécutée.

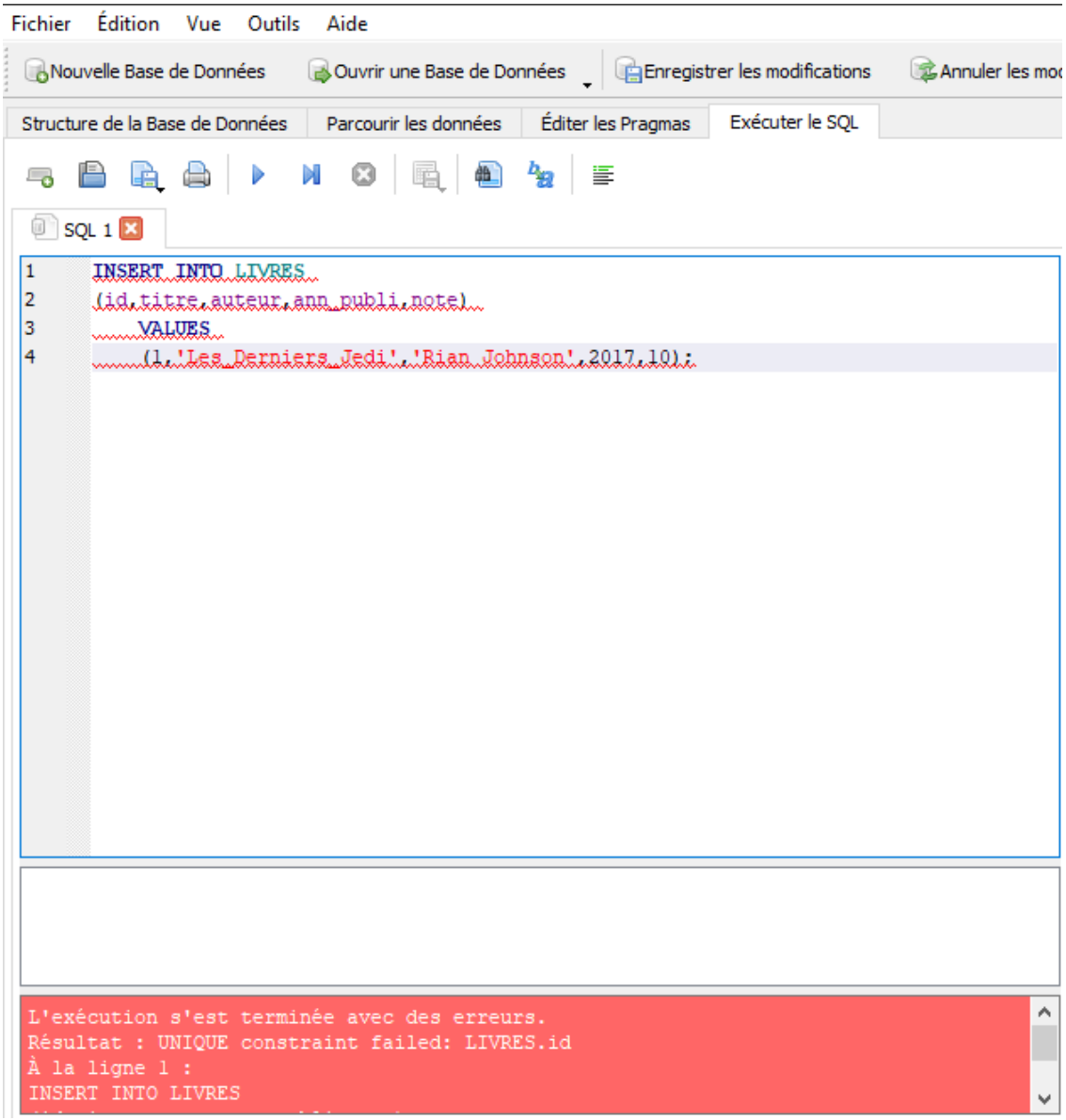
La table “LIVRES” contient bien les données souhaitées (onglet “Parcourir les données”) :



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

Essayons l'ajout du t-uplet suivant suivant la même méthode, que remarquez-vous ?

```
INSERT INTO LIVRES
    (id,titre,auteur,ann_publi,note)
VALUES
    (1,'Les_Derniers_Jedi','Rian Johnson',2017,10);
```



5 Manipulation des données

- Saisissez et exécutez la requête SQL suivante :

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

```
![Select1](img/select1.PNG)
```

```
Comme vous pouvez le constater, la requête SQL a permis d'afficher tous les livres.<br>
Nous avons ici 2 mots clés du langage SQL :

* **SELECT** qui permet de sélectionner les attributs
* **FROM** qui indique la table qui doit être utilisée.

Il est évidemment possible d'afficher seulement certains attributs (ou même un seul), et même c

la requête aurait pu être remplacer par :

```sql
SELECT
    *
FROM
    LIVRES;
```

- Essayez

Pour l'instant nos requêtes affichent tous les livres, il est possible d'utiliser la clause **WHERE** afin d'imposer une (ou des) condition(s) permettant de sélectionner uniquement certaines lignes.

- testez la requête SQL suivante :

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

Vérifiez que vous obtenez bien uniquement les livres écrits par Isaac Asimov.

- testez la requête SQL suivante :

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

*Vérifiez que nous obtenons bien le livre écrit par Asimov publié après 1953.

D'après vous, quel est le résultat de cette requête :

```sql
SELECT
    titre
FROM
    LIVRES
```
```

```
WHERE
    auteur='K.Dick'
    OR note>=8;
```

Écrire une requête permettant d'obtenir les titres livres publiés après 1945 qui ont une note supérieure ou égale à 9.

- testez la requête SQL suivante :

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

Nous remarquons une nouvelle clause SQL **ORDER BY** qui permet d'obtenir les résultats classés dans un ordre précis.

Nous obtenons les livres de K.Dick classés du plus ancien ou plus récent.

- testez la requête SQL suivante :

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

Que remarquez vous ?

- testez la requête SQL suivante :

```
SELECT
    auteur
FROM
    LIVRES ;
```

Cette requête affiche plusieurs fois certains auteurs (les auteurs qui ont écrit plusieurs livres présents dans la base de données).

Il est possible d'éviter les doublons grâce à la clause **DISTINCT**.

- testez la requête SQL suivante :

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

5.1 Jointure sur deux tables

- Créez une nouvelle base de données que vous nommerez par exemple db_livres_auteurs.db
- Créez une table AUTEURS à l'aide de la requête SQL suivante :

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

- Ajoutez des données à la table AUTEURS à l'aide de la requête SQL 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');
'''

* Créez une table LIVRES à l'aide de la requête SQL suivante :

```sql
CREATE TABLE LIVRES
    (id INT, titre TEXT, id_auteur INT, ann_publi INT, note INT);
```

- Ajoutez des données à la table LIVRES à l'aide de la requête SQL 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);
```

Nous avons 2 tables, grâce aux jointures nous allons pouvoir associer ces 2 tables dans une même requête.

En général, les jointures consistent à associer des lignes de 2 tables. Elles permettent d'établir un lien entre 2 tables. Qui dit lien entre 2 tables dit souvent clef étrangère et clef primaire.

- Comparez les résultats entre les deux requêtes suivantes:

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

et

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

- testez la requête SQL suivante :

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

puis ajouter un tri par "ann_publi" croissant.

- Que va faire cette requête ?

Vérifiez votre réponse en l'exécutant et en faisant une requête "SELECT * FROM LIVRES;".

Puis

```
INSERT INTO
    LIVRES
        (id,titre,auteur,ann_publi,note)
VALUES
    (17,'Hypérion','Simmons',1989,8);
```

- Que va faire cette requête ?

Vérifiez votre réponse en l'exécutant et en faisant une requête "SELECT * FROM LIVRES;".

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

- Que va faire cette requête ?

Vérifiez votre réponse en l'exécutant et en faisant une requête "SELECT * FROM LIVRES;".

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

[ ]: