Base de données: langage SQL:

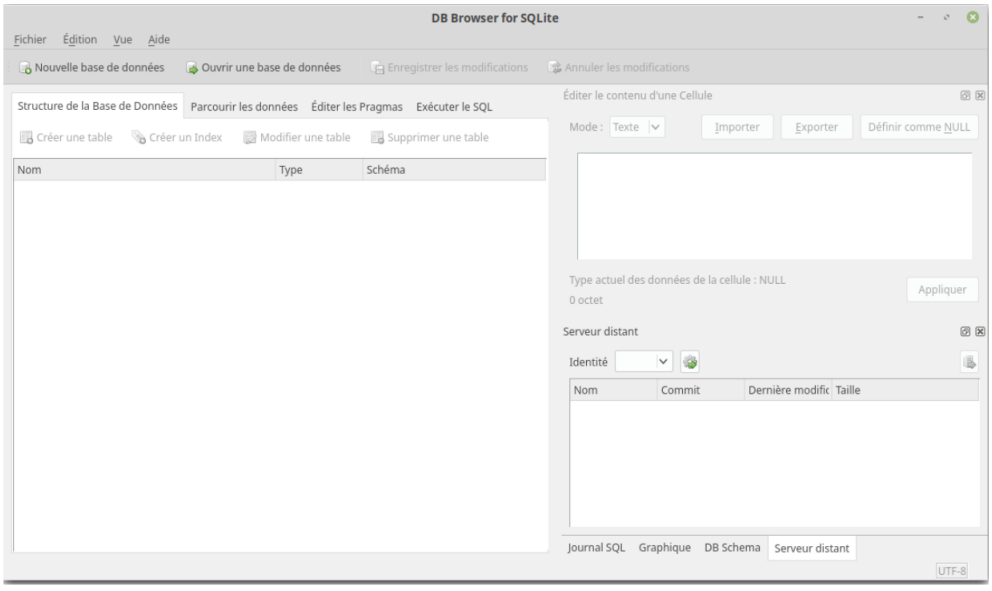
Nous avons eu l'occasion d'étudier la structure d'une base de données relationnelle, nous allons maintenant apprendre à réaliser des requêtes, c'est-à-dire que nous allons apprendre à créer une base de données, créer des attributs, ajouter de données, modifier des données et enfin, nous allons surtout apprendre à 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.

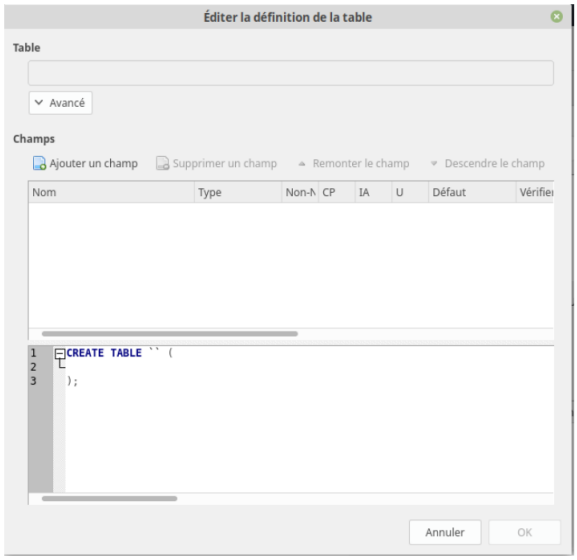
Pour créer une base de données et effectuer des requêtes sur cette dernière, nous allons utiliser le logiciel "DB Browser for SQLite": `https://sqlitebrowser.org/`.

SQLite est un système de gestion de base de données relationnelle très répandu. Noter qu'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. Nous allons commencer par créer notre base de données.

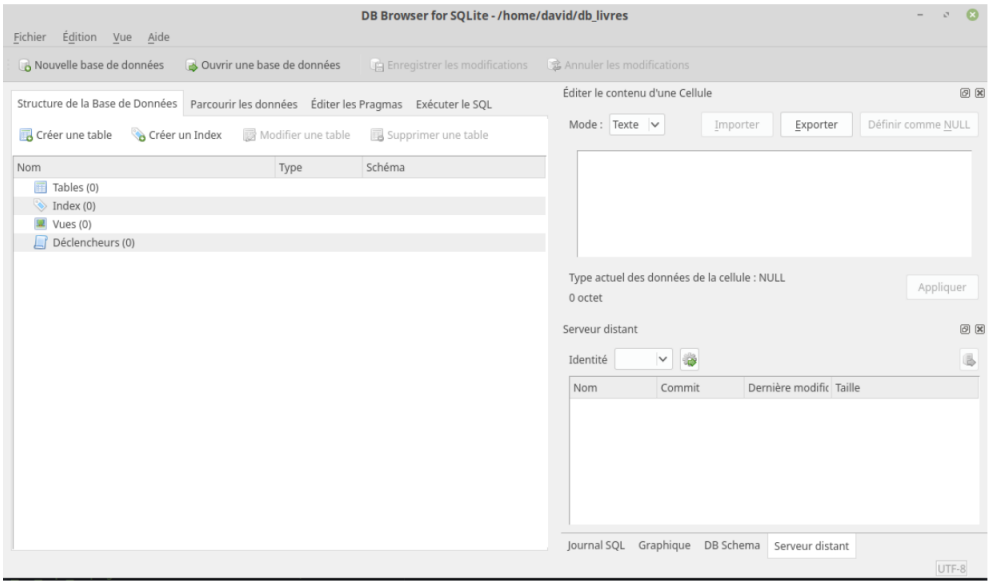
Exercice 1: Après avoir lancé le logiciel "DB Browser for SQLite", vous devriez obtenir ceci:



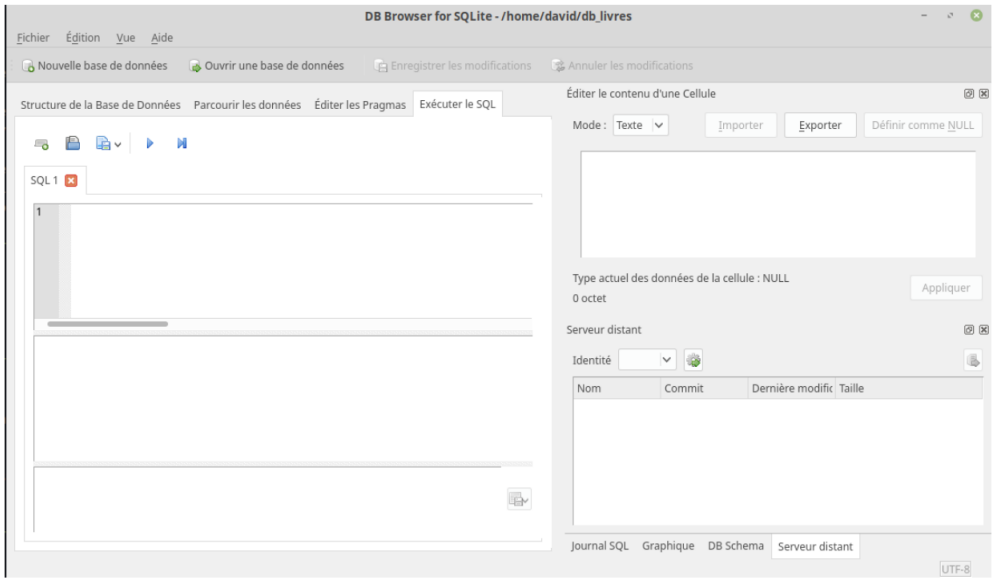
Cliquez sur Nouvelle base de données. Après avoir choisi un nom pour votre base de données (par exemple "db_livres.db"), vous devriez avoir la fenêtre suivante:



Cliquez alors sur Annuler. Notre base de données a été créée:



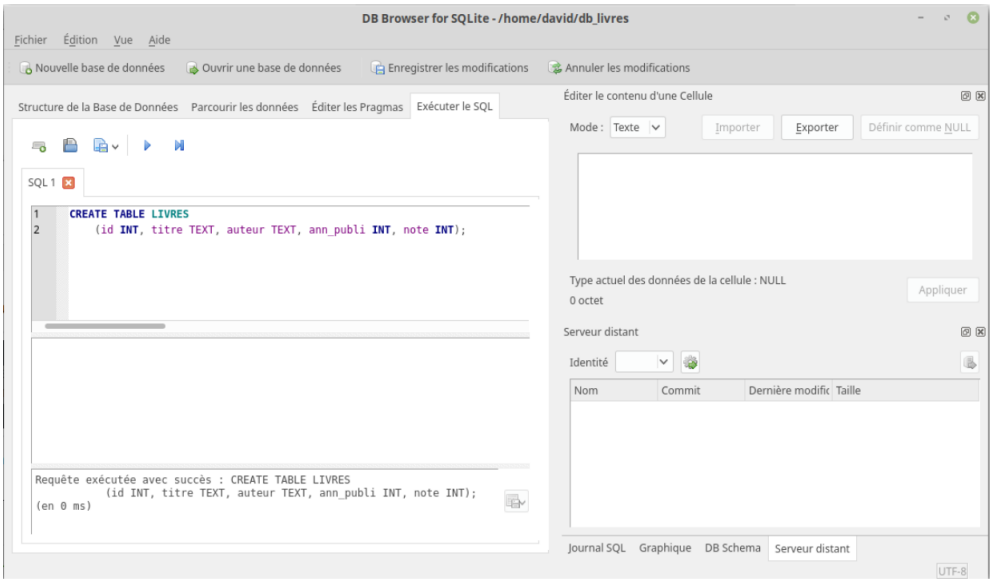
Mais pour l'instant elle ne contient aucune table (aucune relation), pour créer une table, cliquez sur l'onglet "Exécuter le SQL". On obtient alors:



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);
```

Cliquez ensuite sur le petit triangle situé au-dessus de la fenêtre SQL 1 (ou appuyez sur F5), vous devriez avoir ceci:



Comme indiqué dans la fenêtre, "Requête exécutée avec succès"! Vous venez de créer votre première table. Revenons sur cette première requête:

Le CREATE TABLE LIVRES ne devrait pas vous poser de problème: nous créons une nouvelle table nommée "LIVRES". La suite est à peine plus complexe. Nous créons ensuite les attributs:

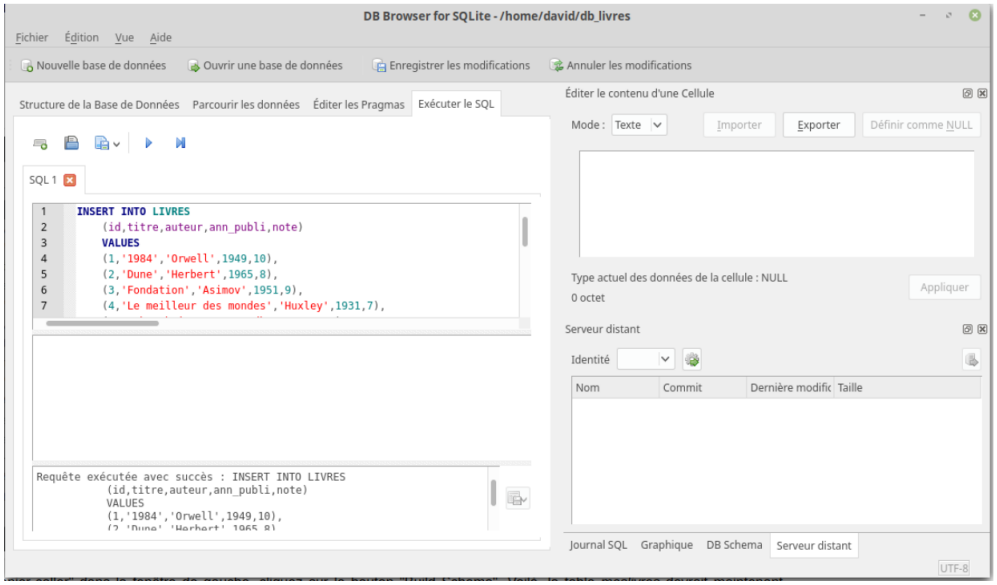
- id
- titre
- auteur
- ann_publi
- note

Comme vous l'avez sans doute remarqué, nous avons pour chaque attribut précisé son domaine: id: entier (INT), titre: chaîne de caractères (TEXT), auteur: chaîne de caractères, ann_publi: entier et note: entier. Nous allons maintenant ajouter des données:

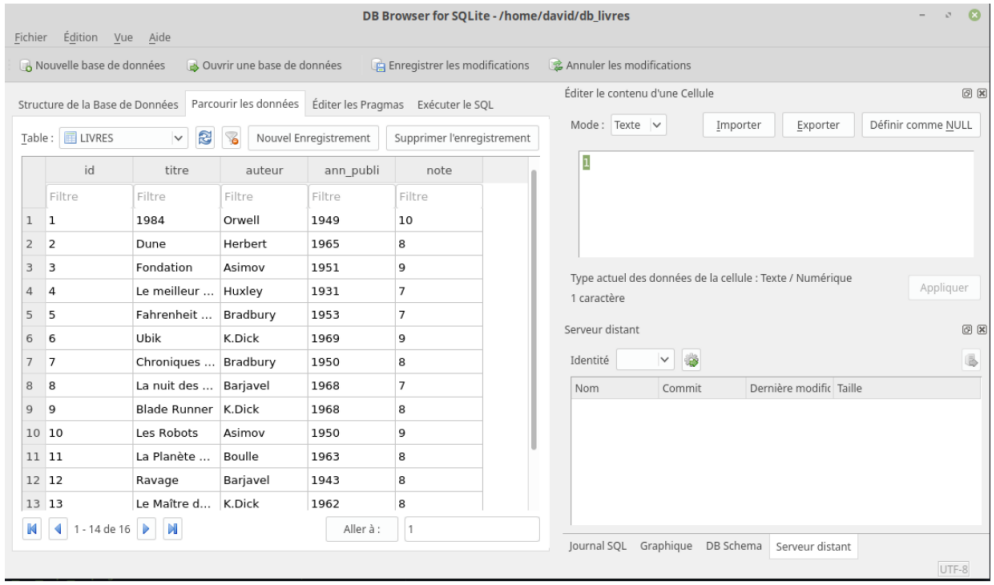
Exercice 2: Toujours dans l'onglet "Exécuter le SQL", après avoir effacé la fenêtre SQL 1, copiez-collez dans cette même fenêtre la requête ci-dessous:

```
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 Ã','Van Vogt',1945,7),
(15,'La Fin de l'éternité','Asimov',1955,8),
(16,'De la Terre à la Lune','Verne',1865,10);
```

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



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

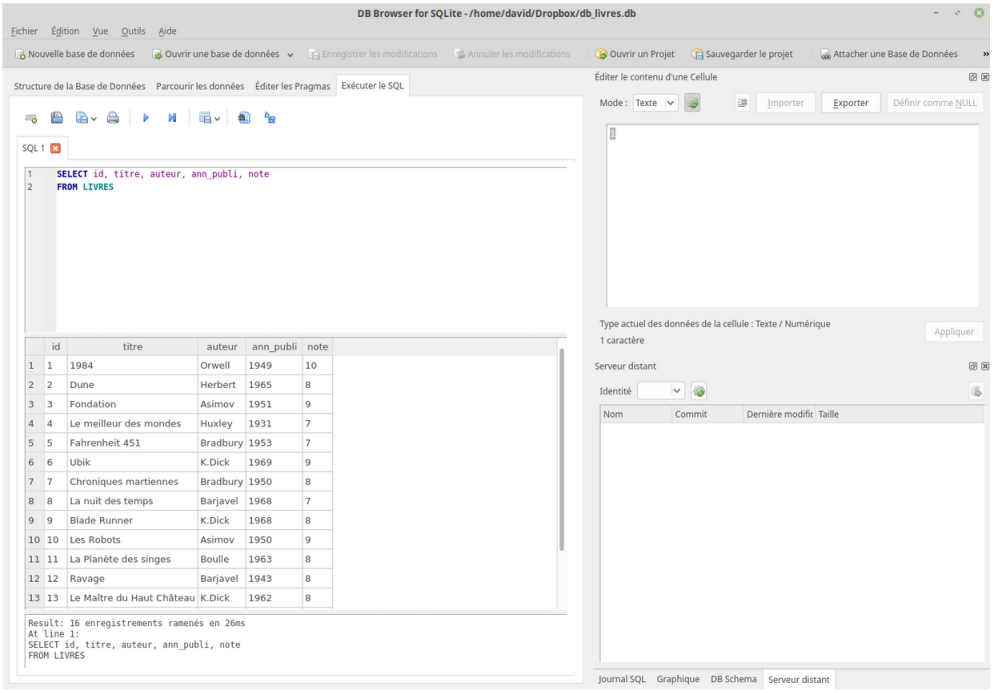


Nous allons apprendre à effectuer des requêtes d'interrogation sur la base de données que nous venons de créer. Toutes les requêtes se feront dans la fenêtre SQL 1 de l'onglet "Exécuter le SQL"

Exercice 3: Saisissez la requête SQL suivante puis appuyez sur le triangle (ou la touche F5)

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

Après un temps plus ou moins long, vous devriez voir s'afficher ceci:



Comme vous pouvez le constater, notre requête SQL a permis d'afficher tous les livres. Nous avons ici 2 mots clés du langage SQL SELECT qui permet de sélectionner les attributs qui devront être "affichés" (je mets "affichés" entre guillemets, car le but d'une requête sql n'est pas forcément d'afficher les données) et FROM qui indique la table qui doit être utilisée. Il est évidemment possible d'afficher seulement certains attributs (ou même un seul):

Exercice 4: Saisissez la requête SQL suivante et vérifiez que vous obtenez bien uniquement les titres et les auteurs des livres.

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

Exercice 5: Écrivez et testez une requête permettant d'obtenir uniquement les titres des livres.

NB: Si vous désirez sélectionner tous les attributs, vous pouvez écrire:

```
SELECT *
FROM LIVRES
```

à la place de:

```
SELECT id, titre, auteur, ann_publi, note
FROM meslivres
```

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. La condition doit suivre le mot-clé WHERE:

Exercice 6: Saisissez et testez la requête SQL suivante. Vérifiez que vous obtenez bien uniquement les livres écrits par Isaac Asimov.

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

Exercice 7: Écrivez et testez une requête permettant d'obtenir uniquement les titres des livres écrits par Philip K.Dick.

Il est possible de combiner les conditions à l'aide d'un OR ou d'un AND

Exercice 8: Saisissez et testez la requête SQL suivante. Vérifiez que nous obtenons bien le livre écrit par Asimov publié après 1953 (comme vous l'avez sans doute remarqué, il est possible d'utiliser les opérateurs d'inégalités).

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

Exercice 9: D'après vous, quel est le résultat de cette requête?

```
SELECT titre
FROM LIVRES
WHERE auteur='K.Dick' OR note>=8
```

Exercice 10: Écrire une requête permettant d'obtenir les titres livres publiés après 1945 qui ont une note supérieure ou égale à 9.

Il est aussi possible de rajouter la clause SQL ORDER BY afin d'obtenir les résultats classés dans un ordre précis.

Exercice 11: Saisissez et testez la requête SQL suivante. Nous obtenons les livres de K.Dick classés du plus ancien ou plus récent.

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

Il est possible d'obtenir un classement en sens inverse à l'aide de la clause DESC.

Exercice 12: Saisissez et testez la requête SQL suivante. Nous obtenons les livres de K.Dick classés du plus récent au plus ancien.

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

Exercice 13: Que se passe-t-il quand la clause ORDER BY porte sur un attribut de type TEXT?

Vous pouvez constater qu'une requête du type suivant affiche plusieurs fois certains auteurs (les auteurs qui ont écrit plusieurs livres présents dans la base de données).

```
SELECT auteur
FROM LIVRES
```

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

Exercice 14: Saisissez et testez la requête SQL suivante:

```
SELECT DISTINCT auteur
FROM LIVRES
```

Nous avons vu précédemment qu'une base de données peut contenir plusieurs relations (plusieurs tables).

Exercice 15: Créez une nouvelle base de données que vous nommerez par exemple db_livres_auteurs.db

Exercice 16: 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);
```

Exercice 17: Créez une table LIVRES à l'aide de la requête SQL suivante:

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

Exercice 18: 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');
```

Exercice 19: 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 Ã',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.

Exercice 20: Saisissez et testez la requête SQL suivante:

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

Le "FROM LIVRES INNER 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 de la ligne A soit égal à l'attribut id_auteur 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.

Cette notion de jointure n'est pas évidente, prenez votre temps pour bien réfléchir et surtout n'hésitez pas à poser des questions.

Exercice 21: Saisissez et testez la requête SQL suivante. Comme vous pouvez le constater, le résultat est différent, cette fois-ci **ce sont les lignes de la table LIVRES qui viennent se greffer sur la table AUTEURS**.

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

Dans le cas d'une jointure, il est tout à fait possible de sélectionner certains attributs et pas d'autres:

Exercice 22: Saisissez et testez la requête SQL suivante:

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

Exercice 23: Saisissez et testez la requête SQL suivante:

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

Si un même nom d'attribut est présent dans les 2 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)

Exercice 24: Saisissez et testez la requête SQL suivante:

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

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

Exercice 25: Saisissez et testez la requête SQL suivante:

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

Enfin, pour terminer avec les jointures, vous devez savoir que nous avons abordé la jointure la plus simple (INNER JOIN). Il existe des jointures plus complexes (CROSS JOIN, LEFT JOIN, RIGHT JOIN), ces autres jointures ne seront pas abordées ici.

Nous en avons terminé avec les requêtes d'interrogation, intéressons-nous maintenant aux requêtes de mise à jour (INSERT, UPDATE, DELETE). Nous allons repartir avec une base de données qui contient une seule table:

Exercice 26: Créez une nouvelle base de données que vous nommerez par exemple db_livres.db

Exercice 27: Créez une table LIVRES à l'aide de la requête SQL suivante:

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

Exercice 28: Ajoutez des données à la table LIVRES à l'aide de la requête SQL suivante:

```
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 Ã','Van Vogt',1945,7),
(15,'La Fin de l'éternité','Asimov',1955,8),
(16,'De la Terre à la Lune','Verne',1865,10);
```

Nous avons déjà eu l'occasion de voir la requête permettant d'ajouter une entrée (utilisation d'INSERT)

Exercice 29: Que va faire la requête suivante? Vérifiez votre réponse en l'exécutant et en faisant une requête "SELECT * FROM LIVRES".

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

Exercice 30: Écrivez et testez une requête permettant d'ajouter le livre de votre choix à la table LIVRES.

"UPDATE" va permettre de modifier une ou des entrées. Nous utiliserons "WHERE", comme dans le cas d'un "SELECT", pour spécifier les entrées à modifier. Voici un exemple de modification:

Exercice 31: 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'
```

Exercice 32: Écrivez une requête permettant d'attribuer la note de 10 à tous les livres écrits par Asimov publiés après 1950. Testez cette requête.

"DELETE" est utilisée pour effectuer la suppression d'une (ou de plusieurs) entrée(s). Ici aussi c'est le "WHERE" qui permettra de sélectionner les entrées à supprimer.

Exercice 33: 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'
```

Exercice 34: Écrivez une requête permettant de supprimer les livres publiés avant 1945. Testez cette requête.