TP 2 : SQL

Nous allons travailler sur une base de données `biblio` décrivant décrivant des emprunts de livres au sein d'un réseau de bibliothèques.

Le schéma relationnel de la base est le suivant :

- **Membre** (IdMembre, nomMembre, adrMembre, cpMembre)
- **Biblio** (IdBiblio, nomBiblio, adrBiblio, cpBiblio)
- **Livre** (IdLivre,titreLivre, auteurLivre, categLivre, *IdBiblio*)
- **Emprunt** (IdLivre, IdMembre, dateEmprunt, dureeEmprunt)

avec :

- `IdMembre` : numéro du membre
- `nomMembre` : nom du membre
- `adrMembre` : adresse du membre
- `cpMembre` : code postal associé à l'adresse du membre
- `IdBiblio` : code de la bibliothèque
- `nomBiblio` : nom de la bibliothèque
- `adrBiblio` : adresse de la bibliothèque
- `cpBiblio` : code postal associé à l'adresse de la bibliothèque
- `IdLivre` : numéro du livre
- `titreLivre` : titre du livre
- `auteurLivre` : auteur du livre
- `categLivre` : catégorie du livre
- `dateEmprunt`: date de début de l'emprunt
- `dureeEmprunt` : durée de l'emprunt (en nombre de jours)

Les attributs ou groupes d'attributs soulignés (resp. en italique) correspondent aux clés primaires (resp. étrangères) des relations.

Type des attributs :

- `IdMembre`, `categLivre`, `dureeEmprunt`, `IdBiblio`: entier
- `categLivre` : smallint
- `dateEmprunt`: date
- `prixLivreJour` : réel
- `nomMembre`, `adrMembre`, `cpMembre`, `nomBiblio`, `adrBiblio`, `cpBiblio`, `IdLivre`, `titreLivre`, `auteurLivre`, `IdLivre`: chaînes de caractères

Partie 1

Executer sqlite3

Pour travailler sur sqlite3 et y créer une base de données `biblio.db` , ouvrez un terminal, déplacez-vous dans votre dossier de travail et tapez:

```
sqlite3 biblio.db
```

Fichier source

Le fichier creat_biblio.sql contient les commandes SQL pour créer et remplir la base de données.

Téléchargez ce fichier, et mettez-le dans votre dossier de travail.

Dans un premier temps, ouvrez-le sous geany pour regarder son contenu. Il commence par 4 commandes `DROP TABLE` sur chacune des tables permettant de supprimer une table, `DROP TABLE IF EXISTS` permet de supprimer une table si elle existe. Cela nous permet de relancer le fichier en exécution même si on l'avait déjà lancé sans déclencher d'erreur.

Ensuite nous avons les 4 `CREATE TABLE` puis différentes insertions dans ces tables.

Lecture du fichier source

Revenez dans le terminal :

Lancer sqlite3 et tapez `.read creat_biblio.sql`

Si tout se passe bien, sans erreur :

```
sqlite>.read creat_biblio.sql
sqlite>
```

Schéma de la base

Maintenant vous pouvez verifier que les tables ont été créées avec la commande `.tables`.

```
sqlite>.tables
```

Vous pouvez également vérifier la structure des tables avec `.schema`:

```
sqlite>.schema Livre
```

Requêtes et commandes d'entrée/sortie

L'interpréteur permet d'exécuter des requêtes et de voir leur résultat dans le terminal.

Pour afficher le contenu d'une table, exécutez la requête

```
sqlite> SELECT * FROM  Livre ;
```

- Les requêtes SQL se terminent par un point virgule `;`
- Les commandes d'entrée/sortie commencent par un point `.` et ne nécessitent pas de point-virgule.

- Recommencez après avoir exécuté la commande `.header ON`. Que se passe-t-il?
- Faites de même avec les commandes :
  - `.header OFF`
  - `.mode line`
  - `.mode column`

- `.width` permet de donner une taille d'affichage à chaque colonne
- `.output resultats.txt` permet de ne pas faire un affichage écran mais de recupérer les résultats dans le fichier « `resultats.txt` »
- `.output stdout` permet de revenir au mode écran.

Insertion de valeurs

Les commandes d'insertion suivantes contiennent des erreurs. Essayez de les exécuter, puis corrigez-les si besoin.

```
INSERT INTO Membre VALUES (2, 'Topeck', 'Marseille', '13012');
```

```
INSERT INTO Livre VALUES ('056G667X','La planète des sages', 'Jul&Pepin', 6, 208);
```

```
INSERT INTO Livre VALUES ('0056561U','Mort sur le Nil', 'Agatha Christie', 1, 77);
```

```
INSERT INTO Biblio VALUES (77, NULL, 'Marseille', '13005');
```

Partie 2

Voici pour vous entraîner une série de requêtes à effectuer sur la base. Il est recommandé de mettre ces requêtes dans un fichier `requetes.sql` puis de lancer leur exécution via la commande `.read requetes.sql`

1. Liste des auteurs présents dans la base.

2. Liste des livres de Victor Hugo.

3. Liste des livres de Victor Hugo appartenant à la catégorie 5 (Théâtre).

4. Liste des livres de Jacques Prévert ou de Gilbert Sinoué.

5. Liste des membres n’habitant pas à Marseille.

6. Liste des livres (titre, auteur) empruntés au moins une fois.

7. Liste des livres jamais empruntés.

8. Liste des membres qui ont emprunté Ruy Blas.

9. Liste des livres empruntés par des habitants du 8ème arrondissement de Marseille.

10. Quels sont les livres qui ont été empruntés au cours de la période allant du 16/01/10 au 15/06/10?

11. Donner la liste des livres se trouvant à St Charles ou à République.

12. Quels sont les numéros des membres qui ont emprunté au moins un livre à Marseille pour une durée supérieure ou égale à 7 jours ?

13. Donner la liste des membres (nomMembre) qui n'ont pas emprunté 'Avicenne'.

14. Donner la liste des membres (nomMembre) qui ont emprunté à la fois ‘Contes pour les enfants pas sages’ et ‘Avicenne’.

15. Liste des membres (nomMembre) qui n'ont emprunté ni ‘Contes pour les enfants pas sages’, ni ‘Avicenne’.

16. Quels sont les numéro et noms des membres qui ont emprunté au moins un livre écrit par Victor Hugo mais qui n’ont jamais emprunté de livre écrit par Jacques Prévert ?

17. Donner le nombre de livres par auteur.

18. Pour chaque catégorie de livre, donner le nombre de livres empruntés au moins une fois.

19. Donner pour chaque livre (titreLivre, auteurLivre) le nombre de fois où il a été emprunté?

20. Donner la liste des livres (titreLivre, auteurLivre) qui ont été empruntés au moins *deux* fois?

21. Donner les noms des membres qui ont emprunté plus de deux livres.

22. Quels sont les nom et adresse complète des membres qui ont empruntés tous les livres de Victor Hugo ?

Partie 3

L'utilisation des bases de données dans un programme Python nécessite de faire appel à des librairies spécialisées. Nous utiliserons ici :

- La librairie `sqlite3` qui permet de connecter le programme à la base de données créée précédemment.
- La librairie `pandas` qui sert à la mise en forme et à l'analyse des données.

Nous allons maintenant essayer de "connecter" un projet Python à la base de données. Pour cela,

nous allons créer un nouveau projet ("TP2") à l'aide de l'éditeur Pycharm.

Commencez par importer les librairies mentionnées :

```
import sqlite3
import pandas
```

Déplacez le fichier `biblio.db` dans votre dossier de projet Python.

Essayez maintenant le commande d'ouverture suivante :

```
db = sqlite3.connect('biblio.db')
```

Attention, si le fichier `biblio.db` n'est pas présent dans le dossier, l'interpréteur ne produit pas d'erreur et crée un fichier `biblio.db` contenant une base vide!

Pour pouvoir exécuter des requêtes dans la base, il faut créer un "curseur" :

```
c = db.cursor()
```

Vous pouvez maintenant exécuter des requêtes à l'aide de ce curseur:

Il est conseillé d'encadrer les requêtes avec une commande de gestion des exceptions pour éviter un arrêt brutal en cas d'erreur de syntaxe SQL :

```
try:
    c.execute("SELECT * FROM Livre")
except sqlite3.OperationalError:
    print("Erreur SQL!");
```

Une fois la requête exécutée sans erreur, les réponses sont disponibles à l'aide de la commande `fetchall`:

```
reponses = c.fetchall()
for r in reponses:
    print r
```

Mise en forme avec pandas

Les réponses apparaissent sous forme de tuples, ce qui manque un peu de lisibilité. Pour améliorer la mise en forme des résultats, nous utiliserons la librairie `pandas`:

```
with db:
    requete = "SELECT * FROM Livre"
    reponse = pandas.io.sql.read_sql(query, db)
    print (reponse)
```

A faire

Reprenez les requêtes de la partie précédente, exécutez-les et affichez leur résultat à l'aide du programme Python.