4 Manipuler des SGBD SQL avec Python

4.1 Le module Python sqlite3

A faire vous même 1.

Après avoir créé un répertoire "projet_bd". Créez, à l'aide de spyder, un fichier Python (à vous de choisir le nom) puis saisissez et exécutez le programme suivant :

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()

cur.execute("CREATE TABLE LIVRES(id INT, titre TEXT, auteur TXT,
ann_publi INT, note INT)")

conn.commit()

cur.close()
conn.close()
```

Analysons le programme ci-dessus :

Ce programme va vous permettre de vous "connecter" à une base de données (si cette dernière n'existe pas, elle sera créée). Ensuite nous créons une table (une relation) nommée LIVRES.

Analysons le programme ligne par ligne :

```
import sqlite3
```

Nous commençons par importer la bibliothèque sqlite3. Cette bibliothèque va nous permettre d'effectuer des requêtes SQL sur une base de données. Nous utiliserons le SGBD SQLite.

```
conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()
```

Nous créons un objet de type "connection" (conn) qui va nous permettre d'interagir avec la base de données "baseDonnees.db". Vous devriez donc avoir un fichier "baseDonnees.db" dans le même répertoire que votre fichier Python.

Nous créons ensuite un objet de type "cursor" à partir de l'objet de type "connection". Cet objet de type "cursor" va nous permettre de manipuler la base de données et d'obtenir des résultats lorsque nous effectuerons des requêtes.

```
cur.execute("CREATE TABLE LIVRES(id INT, titre TEXT, auteur TXT,
ann_publi INT, note INT)")
```

La méthode "execute" de notre objet de type "cursor" nous permet d'effectuer une requête SQL. Cette requête SQL est en tout point identique à ce que nous avons vu dans le cours sur les bases de données.

```
conn.commit()
```

Pour véritablement exécuter les requêtes, il est nécessaire d'appliquer la méthode "commit" à l'objet de type "connection".

```
cur.close()
conn.close()
```

Avant de terminer le programme, il nécessaire de "fermer" l'objet de type "cursor" et l'objet de type "connection".

Nous allons systématiquement retrouver cette structure dans nos futurs programmes :

- création d'un objet de type "connection"
- création d'un objet de type "cursor"
- préparation d'une ou plusieurs requête(s) (méthode "execute" sur l'objet de type "cursor")
- exécution réelle des requêtes (méthode "commit" sur l'objet de type "connection")
- "fermeture" de l'objet de type "cursor" puis de l'objet de type "connection"

Si vous exécutez une deuxième fois le programme proposé au "À faire vous-même 1", vous aurez droit à une erreur : "OperationalError: table LIVRES already exists". Afin d'éviter ce genre de problème, il est possible de modifier le programme afin que la requête de création de la table LIVRES ne se fasse pas si la table LIVRES existe déjà :

A faire vous même 2.

Après avoir effacé le fichier "baseDonnees.db", saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS LIVRES(id INT, titre TEXT,
auteur TXT, ann_publi INT, note INT)")

conn.commit()

cur.close()
conn.close()
```

A faire vous même 3.

Saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS LIVRES(id INT, titre TEXT,
auteur TXT, ann_publi INT, note INT)")
cur.execute("INSERT INTO LIVRES(id,titre,auteur,ann_publi,note)
VALUES(1,'1984','Orwell',1949,10)")

conn.commit()

cur.close()
conn.close()
```

Rien de particulier à signaler, la requête INSERT est identique à ce qui a été vu dans le cours sur les bases de données.

A faire vous même 4.

Après avoir effacé le fichier "baseDonnees.db", saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()

data = (1,'1984','Orwell',1949,10)

cur.execute("CREATE TABLE IF NOT EXISTS LIVRES(id INT, titre TEXT,
auteur TXT, ann_publi INT, note INT)")
cur.execute("INSERT INTO LIVRES(id,titre,auteur,ann_publi,note)
VALUES(?, ?, ?, ?, ?)", data)
conn.commit()

cur.close()
conn.close()
```

Nous avons créé un tuple (data) contenant toutes les informations. En effet, la méthode "execute" peut prendre un deuxième paramètre un tuple contenant les données à insérer. Les points d'interrogation présents dans la requête indiquent l'emplacement des données à insérer. Le premier ? sera remplacé par le premier élément du tuple (dans notre cas 1), le deuxième ? sera remplacé par le deuxième élément du tuple (dans notre cas '1984') et ainsi de suite...

A faire vous même 5.

Si l'on désire insérer plusieurs données, il est possible de regrouper toutes les données à insérer dans un tableau et d'utiliser la méthode "executemany" à la place de la méthode "execute".

Après avoir effacé le fichier "baseDonnees.db", saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()

datas = [
        (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)
    ]

    cur.execute("CREATE TABLE IF NOT EXISTS LIVRES(id INT, titre TEXT,
auteur TXT, ann_publi INT, note INT)")
    cur.executemany("INSERT INTO LIVRES(id,titre,auteur,ann_publi,note)
VALUES(?, ?, ?, ?, ?)", datas)
    conn.commit()

    cur.close()
    conn.close()
```

A faire vous même 6.

Il n'est pas très pratique d'avoir à gérer l'id (clé primaire). En effet, si je veux ajouter un nouveau livre, il faudra que je connaisse l'id du précédent (incrémentation de l'id). Heureusement, il est possible d'automatiser cette incrémentation.

Après avoir effacé le fichier "baseDonnees.db", saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()

datas = [
        ('1984','Orwell',1949,10),
        ('Dune','Herbert',1965,8),
        ('Fondation','Asimov',1951,9),
        ('Le meilleur des mondes','Huxley',1931,7),
        ('Fahrenheit 451','Bradbury',1953,7),
        ('Ubik','K.Dick',1969,9),
        ('Chroniques martiennes','Bradbury',1950,8),
        ('La nuit des temps','Barjavel',1968,7),
        ('Blade Runner','K.Dick',1968,8),
        ('Les Robots','Asimov',1950,9),
        ('La Planète des singes','Boulle',1963,8),
        ('Ravage','Barjavel',1943,8),
        ('Le Maître du Haut Château','K.Dick',1962,8),
        ('Le monde des Ã','Van Vogt',1945,7),
        ('La Fin de l'éternité','Asimov',1955,8),
        ('De la Terre à la Lune','Verne',1865,10)
    ]

    cur.execute("CREATE TABLE IF NOT EXISTS LIVRES(id INTEGER PRIMARY KEY
AUTOINCREMENT UNIQUE, titre TEXT, auteur TXT, ann_publi INT, note INT)")
    cur.executemany("INSERT INTO LIVRES(titre,auteur,ann_publi,note)
VALUES(?, ?, ?, ?)", datas)
    conn.commit()

    cur.close()
    conn.close()
```

Ouvrez le fichier "baseDonnees.db" à l'aide du logiciel "DB Browser for SQLite" et vérifiez que les données ont bien été ajoutées à la table LIVRES.

Vous pouvez constater que nous avons bien l'attribut "id", même si ce dernier n'a pas été renseigné dans les données (absence d'id dans le tableau datas). Désormais l'id sera incrémenté automatiquement grâce au "id INTEGER PRIMARY KEY

AUTOINCREMENT UNIQUE" (attention il est nécessaire d'utiliser INTEGER à la place du INT habituel) présent dans la requête de création de la table LIVRES. Attention, de bien penser à supprimer un ? dans la requête d'insertion (chaque tuple contient maintenant 4 éléments (nous en avions 5 quand l'id n'était pas géré automatiquement)).

A faire vous même 7. ☐

Il est tout à fait possible de rajouter une nouvelle donnée :

Saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()
nvx_data = ('Hypérion','Simmons',1989,8)

cur.execute("INSERT INTO LIVRES(titre,auteur,ann_publi,note) VALUES(?, ?, ?, ?)", nvx_data)
conn.commit()

cur.close()
conn.close()
```

A faire vous même 8. ☐

Il est possible de modifier des données déjà présentes dans la table.

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()

modif = (7, 'Hypérion')
cur.execute('UPDATE LIVRES SET note = ? WHERE titre = ?', modif)
conn.commit()

cur.close()
conn.close()
```

A faire vous même 9. ☐

Il est aussi possible de supprimer une donnée :

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()

suppr = ('Hypérion',)
cur.execute('DELETE FROM LIVRES WHERE titre = ?', suppr)
conn.commit()

cur.close()
conn.close()
```

A faire vous même 10. ☐

Enfin, pour terminer cette introduction sur l'utilisation de sqlite en Python, nous devons nous intéresser aux requêtes de type "SELECT" :

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')
```

```
cur = conn.cursor()

cur.execute('SELECT * FROM LIVRES')
conn.commit()

liste = cur.fetchall()

cur.close()
conn.close()
```

À l'aide de la console, déterminez la valeur référencée par la variable liste

A faire vous même 11. ☐

Il est possible d'utiliser les points d'interrogation dans une requête de type SELECT :

Saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()

recherche = (1960, 8)

cur.execute('SELECT titre FROM LIVRES WHERE ann_publi < ? AND note
> ?', recherche)
conn.commit()

liste = cur.fetchall()

cur.close()
conn.close()
```

À l'aide de la console, déterminez la valeur référencée par la variable liste

4.2 Mini-projet : Annuaire téléphonique

Projet à faire par groupe de 3 élèves.

Voici un script permettant d'écrire et de rechercher dans un fichier texte qui sert d'annuaire téléphonique :

```
def ecriture() :
    with open('fichier.txt','w') as f :
        nom = 'tyty'
        while nom != '0':
            nom=input('Entrez un nom (0 pour terminer) : ')
            if nom != '0' and nom != '' :
                telephone = input("Entrez numéro de téléphone de "+nom+" : ")
                f.write(nom+ " : "+telephone+'\n')

def recherche() :
    with open('fichier.txt','r') as f :
        ligne='tyty'
        trouve= False
        nom=input('Entrez un nom à rechercher : ')
        while ligne != '' :
            ligne=f.readline()
            if ligne.startswith(nom) :
                print(ligne)
                trouve = True
        if not trouve :
            print('Inconnu')
```

```
#MENU PRINCIPAL
print("ANNUAIRE TELEPHONIQUE - DEBUT DE PROGRAMME")
choix = 'toto'
while choix != '0' :
    choix = input("""
0-quitter
1-écrire dans le répertoire
2-rechercher dans le répertoire
Votre choix ?
    """)
    if choix=='1' :
        ecriture()
    elif choix=='2' :
        recherche()
    elif choix != '0' :
        print("Réponse non valable")
print("ANNUAIRE TELEPHONIQUE - FIN DE PROGRAMME")
```

Copiez ce script et faites-le fonctionner.

L'objectif est de créer un script Python similaire à celui qui se trouve ci-dessus mais qui enregistre les données dans une SGBD Sqlite.

- Vous pouvez reprendre la structure globale
- En plus de *Nom* et *Numéro de téléphone*, ajoutez *Prénom*, *Date de naissance* et *Adresse*.
- Il doit être possible d'effectuer une recherche exacte sur n'importe quel attribut.
- Il doit être possible d'effectuer une recherche partielle avec une partie d'attribut (recherche sur upo doit donner Dupont, Doupoir, ...) sur n'importe quel attribut.
- Il doit être possible d'effectuer une recherche partielle qui va chercher dans tous les attributs.
- BONUS : Soignez votre affichage, utilisez des couleurs et des effets, notamment pour mettre en évidence vos critères dans les résultats de recherche

Testez ceci dans la console :

```
>>> print('\033[31m' + 'ce texte est rouge')
```

Plus d'info :

https://chamilo.univ-grenoble-alpes.fr/courses/IUT1RT1M2109/document/1718-Sokoban/build/sequences_ansi.html

Term NSI **GRILLE D'EVALUATION :** **Noms :** **Note :** **/10**	Rien d'écrit	Ne fonctionne pas	Fonctionne avec aide	Fonctionne sans aide	Fonctionne de manière optimale
Structure globale (2 points)	l				
Ecriture dans SGBD (2 points)					
Recherche exacte (2 points)					
Recherche partielle sur un attribut (2 points)					
Recherche partielle sur tous les attributs (2 points)					