Licence Ingénierie Logicielle et Système d'Information

TP N°3 Base de données en python avec sqlite3

Python et SQL

Le système de gestion de bases de données relationnelles (SGBDR) que nous utiliserons préférentiellement s'appelle SQLite ; il présente l'avantage d'être présent dans la bibliothèque standard de Python. Cela signifie donc que vous pouvez écrire en Python une application contenant son propre SGBDR intégré à l'aide du module sqlite3.

1. Installation de SQLite

Cette section vous guide à l'installation de SQLite, si vous souhaitez l'installer sur vos machines personnelles. Si vous avez déjà SQLite installé, vous pouvez aller directement à la section 2.

1.1 Installation sous Windows

– Aller sur la page de téléchargement de SQLite : http://www.sqlite.org/download.html

– Double-cliquer sur sqlite3.exe.

2. Lancement de requêtes SQL

2.1. Créer manuellement une base de données '**LUSILSI2020.db**' depuis DB Browser for SQLite avec le schéma suivant :

- Film(*idFilm* entier, titre texte)
- Acteur(**idActeur** entier, nom texte, prenom texte)
- Filmographie(*idActeur* entier, idFilm entier)

Nb : Les attributs en italique sont des clés primaires.

Avant de créer une table T, verrier qu'elle n'existe pas comme suit : drop table if existe T;

2.2. Ecrire une requête pour créer la table Films avec la colonne idFilm de type entier (en tant que clé primaire) et la colonne titre de type texte (non null).

Ecrire un programme de connexion à la base de données 'LUSILSI2020.db' et afficher les requêtes suivantes :

2.3. Ajouter à la table Films les titres de films suivants, avec leurs clés respectives : {(1,“Les évadés”), (2,“Le parrain”), (3,“La vie de Pi”)}

2.4. Ecrire une requête pour afficher tous les éléments de la table Films. Voilà le résultat que vous devez voir apparaitre :

1|Les évadés
2|Le parrain
3|La vie de Pi

2.5. Ecrire une requête pour ajouter les titres de films suivants {(4, “Chocolat”), (5, “Scarface”), (6,“Rango”)}.

2.6. Ecrire une requête pour afficher tous les ´éléments de la table Films. Voilà le résultat que vous devez voir apparaitre :

1|Les évadés
2|Le parrain
3|La vie de Pi
4|Chocolat
5|Scarface
6|Rango

2.7. Ecrire une requête pour afficher tous les titres de films. Voilà le résultat que vous devez voir apparaitre :

Les évadés
Le parrain
La vie de Pi
Chocolat
Scarface
Rango

2.8. Ecrire une requête pour créer la table Acteurs avec la colonne idActeur de type entier (en tant que clé primaire) et les colonnes nom et prenom de types texte (non null).

2.9. Ecrire une requête pour ajouter les acteurs suivants : {Johnny Deep, Al Pacino, Suraj Sharma}.

2.10. Ecrire une requête qui permet de lister le nom des acteurs. Voilà le résultat que vous devez voir apparaitre :

Deep
Pacino
Sharma

2.11. Ecrire une requête pour créer la table Filmographie. Ajouter les contraintes d'intégrité : idActeur et idFilm sont des clés étrangères correspondant aux attributs Acteur(idActeur) et Film(idFilm).

RECAP

La première chose à faire c'est d'importer le bon module import sqlite3 Comme dans le cas de la ligne de commande, il faut ensuite se connecter à une base (si elle n'existe pas, elle sera créée) connexion = sqlite3.connect('ma_base.db')

A la fin de votre programme, si vous voulez que les changements apportés à la base soient enregistrés : connexion.commit()

Et puis n'oubliez pas de fermer proprement connexion.close()

Gestion des requêtes

On ne réalise pas des requêtes directement sur une connexion mais sur un objet qu'on appel curseur et qui peut contenir plusieurs lignes d'une table, quelque soit son schéma.

Pour créer un curseur : curseur = connexion.cursor()

Pour utiliser un curseur, on utilise la primitive execute.

Par exemple : curseur.execute("SELECT * FROM hotel WHERE ville='Nice'")

On trouve alors 0, 1 ou plusieurs lignes dans le curseur.

Pour récupérer la première ligne (et 'déplacer' le curseur sur la ligne suivante), on peut écrire : ligne = **curseur.fetchone()**

Vous pouvez aussi récupérer toutes les lignes renvoyés sous forme d'une liste (ce qui vide le curseur) : lignes = **curseur.fetchall()**

A ce moment, il est possible de tester si aucune ligne a été renvoyée en regardant la taille de la liste.

Bien entendu, les boucles sont vos amis : For ligne in curseur.execute('SELECT * FROM hotel WHERE etoile = 3') print ligne

Quand on récupère une ligne dans une variable, elle se comporte comme un iterable : curseur.execute('SELECT nom, prenom FROM Client') ligne = curseur.fetchone() nom = ligne[0] prenom = ligne[1]

Vous pouvez bien entendu utiliser des variables :

prenom = 'Jayme' nom = 'Labédé'

curseur.execute('SELECT numclient FROM Client where prenom= ? AND nom = ?', (prenom,nom))

Attention, même si ça peut éventuellement marcher, il est très déconseillé de faire ça comme ça : curseur.execute("SELECT numclient FROM Client where prenom='" + prenom + "' AND nom = '" + nom + "'"