TP - BDD et Film

Thibault PENNING

Présentation

Dans ce TP, nous allons utiliser la base de données d'IMDb et en particulier son top 250 afin de manipuler les données et d'obtenir des statistiques.

Pour être plus précis, nous allons utiliser tous les films ayant déjà été dans ce top 50. Nous utiliserons pour cela les données de ce site.

Les données présentes venant de ce site est d'IMDb ne sont donc pas utilisables en dehors des contrats de licence de ce dernier.

Les données présentées sont sous forme de CSV qui a été scrapé sur les sites Internet.

L'une de vos premières tâches sera donc de créer les tables et d'ajouter les données.

Afin de créer et d'interagir avec une base de données, nous allons utiliser l'API présente de base dans Python de SQLite 3.

SQLite3

Lors du précédent TP, nous avons utilisé Capytal directement afin d'utiliser du SQL. Cela est une utilisation qui ne reflète pas une réalité. Comme nous l'avons vu dans nos cours, le langage SQL permet l'interaction avec un SGBD. Nous allons donc ici utiliser un LGBT intégré dans Python, SQLite. Ce dernier a la particularité, contrairement à la majorité des SGBD, d'interagir directement avec un seul et unique fichier contenant toutes les données de la base de données.

L'une des premières étapes est d'initialiser la base de données :

Pour cela, nous utiliserons la syntaxe suivante :

```
import sqlite3
con = sqlite3.connect("nom.sqlite")
cur = con.cursor()
```

Où `nom.sqlite` est le nom du fichier. Si le fichier n'existe pas, celui-ci sera créé. La variable `cur` est un curseur, et c'est avec lui que nous interagirons. Le but de ce TP est d'utiliser SQLite, mais nous ne rentrerons pas plus dans les subtilités.

Par la suite, 2 commandes seront utiles :

```
cur.execute("REQUETE")
```

Cette commande exécute une requête SQL, par exemple pour créer une table, insérer les données, où effectuer une sélection.

Lors de la modification de la base de données, il vous sera aussi demandé de *commit* Les transactions. En effet, l'API Python de SQLite ne le fait pas automatiquement.

Pour cela, nous utiliserons :

```
con.commit()                                                    (1)
```

(1) Ici nous n'utilisons pas le curseur, mais bien la connexion à la base de données.

Enfin, pour obtenir le résultat d'une requête, notamment d'une sélection, nous utiliserons :

```
res = cur.execute("REQUETE")
res.fetchall()
```

Cela nous retourne une liste de tuples où chaque tuple est un enregistrement de la base de données.

> **Note**
>
> D'autres fonctions existent, vous pouvez aller regarder la documentation à ce sujet.
> On peut notamment citer la fonction :
>
> ```
> res.fetchone()
> ```
>
> Qui se comporte comme la fonction `fetchall` mais qui me retourne non pas une liste, mais un élément de cette dernière. Cela peut être utile lorsque l'on s'attend à obtenir qu'un seul élément.

> **Note**
>
> Il n'est plus possible d'utiliser, comme la dernière fois `.schema`.
> En revanche, vous pouvez au besoin utiliser cette ligne ::!

```
cur.execute("SELECT sql FROM sqlite_schema WHERE type = 'table';").fetchall()
```

En changeant `cur` par votre curseur.

Création de la base de donnée

Création des table

Regardez le fichier CSV, vous y trouverez l'ensemble des clés du dictionnaire sur la première ligne. Cela devrait être normalement :

- Pour `films.csv` : *Titre*, *Année*, *Réalisateur*, *Scénariste*, *Acteur*, *Pays*, *Langue*, *Genre*, *Couleur*, *Note*
- Pour `réalisateur.csv`, `acteur.csv`, `scénariste.scv` : *ID*, *Nom*
- Pour `personage.csv` : *Titre*, *Année*, *Acteur*, *Personnage*

Vous allez donc devoir créer les requêtes afin de créer les tables correspondantes. Et Mais aussi créer leur requête d'insertion pour toutes les valeurs des tableaux.

Ici, vous créez une table par fichier CSV. Toutes les clés de dictionnaire doivent être ajoutées comme attributs dans les relations correspondantes.

Attention, réfléchissez bien aux données pouvant être nulles et les données pouvant servir de clé. En l'occurrence, et assez étonnamment, il manque par exemple certaines fois les informations sur le réalisateur. De plus, dans les CSV, lorsqu'une donnée est manquante, celle-ci est indiquée par non ou dans le fichier une donnée manquante. Cela doit donc se répercuter tant votre base de données.

Je vous conseille de commencer par les fichiers des `réalisateurs`, `acteurs` et `scénaristes` qui sont extrêmement similaires. Puis aux `films`, et lorsque vous serez avancé dans. Le TD de faire la relation `personnage`.

Lorsqu'il existe un *ID*, celui-ci est considéré unique. Remarquer que l'on peut les écrire avec moins de 8 caractères. Réfléchissez aussi aux clés étrangères qui doivent être indiquées.

Insertion des données

La première étape que vous allez réaliser est de créer la base de données. Pour cela, vous allez utiliser les données en CSV disponibles. Pour en simplifier la lecture, vous pouvez utiliser la librairie `csv` incluse dans Python.

La syntaxe est la suivante :

```python
import csv


with open('films.csv', newline='', encoding="utf-8") as csvfile:
    reader = list(csv.DictReader(csvfile))
    for film in reader:
        print(film['Titre'], film['Année'])
```

Aussi, certaines données, comme par exemple le réalisateur, ne sont pas composées d'un seul élément, mais d'une liste d'éléments. Il est de votre ressort de transformer cette chaîne de caractères de type liste en une véritable liste. Grâce à la programmation fonctionnelle, cela se fait extrêmement rapidement. Vous avez ici le droit d'utiliser des built-in comme `split`, car la programmation en Python est ici très secondaire. Dans un premier temps, je vous conseille non pas d'insérer cette dernière en liste, mais d'insérer seulement le premier réalisateur par exemple. Vous pouvez aussi créer n attribut *réalisateur* et *réalisateur principal* d'utiliser le *réalisateur principal* pour l'instant, puis dans un 2nd temps, utiliser l'attribut *réalisateur*.

Tout cela peut paraître de prime abord compliqué, mais ne vous inquiéter pas, cela ne vous prendra que quelques minutes.

Analyse des données

Dans toute la suite, il vous sera demandé de sélectionner les données selon certains critères

Première partie

Pour chaque point, faites une requête retournant :

- ☐ Le nom de chaque film
- ☐ Le nom de chaque réalisateur
- ☐ Toutes les années des films de la sélection
- ☐ Le nom et la note des films, trier par note
- ☐ Tous les films dont on ne connaît pas le réalisateur

Deuxième partie

- ☐ La moyenne des notes des films
- ☐ Pour chaque genre, le nombre de films qui lui appartient
- ☐ Le nombre de films qu'a réalisés chaque réalisateur
- ☐ Le couple composé du nom du film et du réalisateur

- ☐ Tous les acteurs ayant joué avec `Stanley Kubrick`
- ☐ Les films français, par ordre de note descendante

Troisième partie

- ☐ Tous les réalisateurs, triez par nombre de films
- ☐ De même pour les acteurs
- ☐ Les films ayant été réalisés, écrits et jouer par `Charles Chaplin`
- ☐ La moyenne des films, par réalisateur
- ☐ Tous les acteurs ayant joué dans un film contenant le mot
- ☐ Le pays ayant fait le plus de films en noir et blanc
- ☐ Les films français n'étant pas en langue française

Quatrième partie

- ☐ Supprimer tous les films `Avengers`
- ☐ Mettre 10 à tous les films de `Nicolas Winding Refn` et `Lars von Trier`
- ☐ Remplacer toutes les notes des films de `James Cameron` par la note minimale du classement
- ☐ Les personnages de tous les films de `Akira Kurosawa`
- ☐ Tous les personnages des films de `Martin Scorsese` interprétés par `Robert De Niro`
- ☐ De même pour `Leonardo DiCaprio`