Mini projet Bases de Données – Prendre avant les vacances de Noël

Le but de ce projet est de réaliser une interface homme-machine permettant de satisfaire des requêtes simples en SQL. On utilisera la base de données bdfilms construire lors du TD SQL (la version correctement faite, pas la version initiale !). Les requêtes permettront de sélectionner deux (voire trois) attributs, avec au plus deux (voire trois) conditions, sans ordre. L'utilisateur doit donc pouvoir choisir un ou deux attributs à l'aide de listes déroulantes, éventuellement rajouter une ou deux conditions, à l'aide de listes déroulantes portant sur les attributs et de champs de saisie pour les valeurs portant sur ces conditions. Le programme construit la requête SQL correspondante et renvoie le résultat. Les requêtes les plus compliquées sont donc du type : « SELECT title, release_year FROM films WHERE title LIKE "%a" AND runtime > 150 ».
Comme précisé en cours, ce n'est pas très logique d'avoir une requête sans « ORDER BY ». Vous pouvez l'implémenter par défaut pour que le résultat soit plus « réaliste ». Par exemple sur le premier attribut à afficher, en ordre croissant, pour ne pas avoir à rajouter de choix supplémentaire de la part de l'utilisateur. On peut aussi rajouter « DISTINCT » etc. Vous pouvez faire encore plus compliqué si vous le souhaitez, mais vous n'apprendrez pas grand chose de plus : c'est surtout la construction de la base de ce code qui est intéressante, avec la simple sélection d'un ou deux attributs avec zéro, une ou deux conditions.

Il y a trois versions pour ce projet. La version sans serveur me semble plus facile, même si le code final est plus long. Pour ceux qui sont à l'aise, faites une des deux versions avec serveur (la première a ma préférence mais la deuxième est très bien aussi). Tous les fichiers pour les exemples sont dans l'archive `mini_projet_bd`.

- La version la moins technique a priori : sans serveur, en Python pur. On utilise les bibliothèques :
  - Tkinter pour l'interface graphique ;
  - Sqlite pour l'accès aux bases de données.
  - *Cette version a l'avantage de ne pas demander d'installation de bibliothèques, ce qui pose souvent des problèmes –parfois incompréhensibles–. C'est par contre la version où le code est le plus long (Tkinter…)*

- Première version avec serveur, en Python. On utilise/crée :
  - Serveur Apache (avec WAMP par exemple) pour que l'accès à la base de données.
  - Bibliothèque flask pour le serveur ;
  - Bibliothèque mysql.connector pour l'accès aux bases de données ;
  - Page html avec formulaire pour la création de la requête ;
  - Page html + langage Jinja (extension de html, simple à utiliser) pour l'affichage des résultats.
  - *Cette version a l'avantage de faire travailler sur des outils et conceptions « modernes » : architecture propre et souple, éventuellement monopage (même si ici je ne vous le demanderai pas). Elle est bien plus riche que la version sans serveur. Le code est « relativement simple ».*
  - *Personnellement, je suis bluffé par la puissance, avec autant de simplicité, de flask + templates Jinja… Pour ceux qui souhaitent approfondir un peu, on peut même mettre des templates à l'intérieur de templates (par exemple un menu que l'on retrouve sur plusieurs pages)*

- Deuxième version avec serveur, en php + html. On utilise/crée :
  - Serveur Apache (avec WAMP par exemple) pour l'accès à la base de données, et pour que le php puisse être interprété.
  - Page html avec formulaire pour la création de la requête ;
  - Page php pour l'affichage des résultats.
  - *Cette version a l'avantage de faire travailler sur des outils très répandus, même si l'architecture induite par le php commence à avoir vieilli. C'est aussi une version riche, comme la précédente. Le code est « relativement simple », même pour ceux qui n'ont jamais fait de php.*

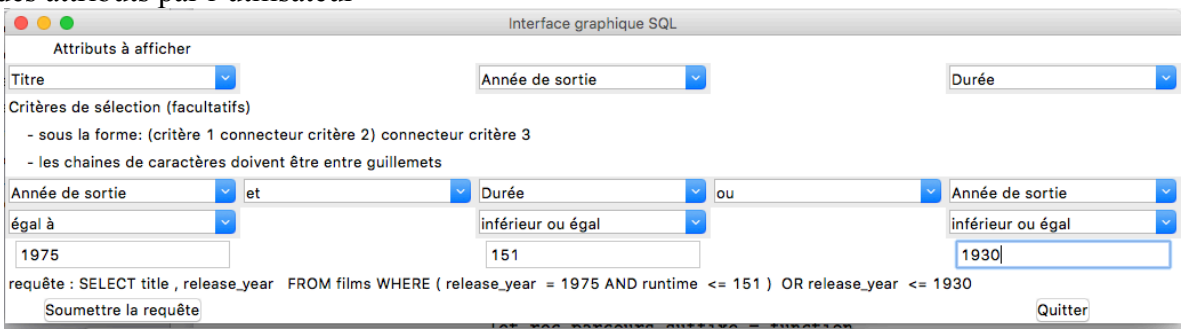
- Remarque : dans la version sans serveur, on utilise SQLlite. Dans la version avec serveur, on utilise MySQL. Vous pouvez exporter sous db Browser for SQLlite la version SQLlite, en version SQL, et

comparer avec le fichier de construction des requêtes MySQL qui se trouve ici : http://www.maths-info-lycee.fr/programmes/bdfilms.db.sql . Vous verrez des différences dans la syntaxe (les deux fichiers s'ouvrent sous NotePad++, Sublime Text, etc.)

1. Version tkinter

Exemple d'interface graphique

Choix des attributs par l'utilisateur



Affichage des résultats :

Titre	Année de sortie	Durée
Metropolis	1927	153
One Flew Over the Cuckoo's Nest	1975	133
Jaws	1975	124
Monty Python and the Holy Grail	1975	91
Pandora's Box	1929	109
Intolerance	1916	197
The Big Parade	1925	151
The Return of the Pink Panther	1975	113
Death Race 2000	1975	80
Hell's Angels	1930	127
The Broadway Melody	1929	100

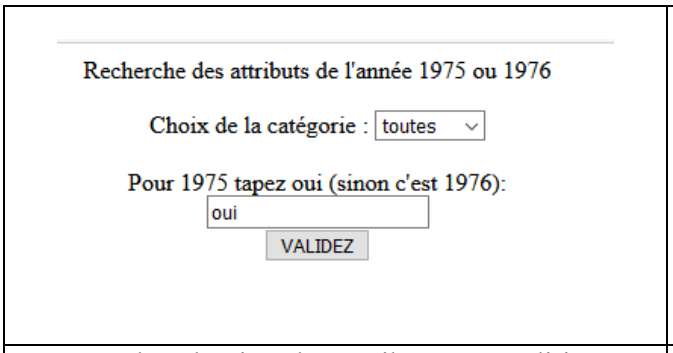
Remarques :

- Dans ces exemples, j'ai choisi d'afficher la requête. Ce n'est pas obligatoire.
- Tkinter plante s'il y a trop de résultats à afficher. On peut choisir de n'afficher que les 25 premiers au cas où (et pour fêter ça un `break` est autorisé)
- Utiliser la méthode grid de Tkinter pour un affichage propre.
- Utiliser le widget Combobox pour les listes déroulantes :

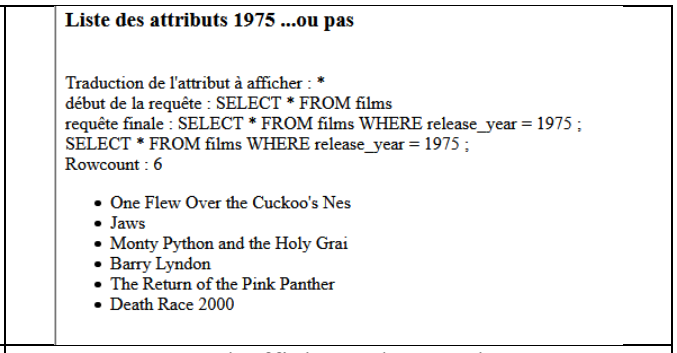
```
Import tkinter as tk
from tkinter import ttk

def action(event):
     # pour vérifier le choix fait, sinon inutile ici
     select = liste_combo0.get()
     print("Vous avez sélectionné en 0 : '", select,"'")
     return()

fenetre = tk.Tk()
fenetre.title("Liste déroulante")
liste_choix=["a", "b", "c", "d", "e"]
liste_combo0 = ttk.Combobox(fenetre, values=liste_choix)
liste_combo0.current(0)
liste_combo0.bind("<<ComboboxSelected>>", action)
liste_combo0.grid(row = 0 , column = 0)
fenetre.mainloop()
```

- Pour la partie SQL, utiliser le fichier `python_sqlite.py`
- Attention : le nombre de lignes sélectionnées est difficilement récupérable avant l'affichage (il existe une variable `rowcount`, utilisable avec le curseur `sqlLite`, mais elle vaut -1 sur un SELECT)

2. Pour les versions avec serveur
   a. Exemples d'interfaces
   Ce sont plutôt des contre-exemples vu comment c'est moche. L'idée est là, à vous de faire mieux. Le code est donné ci-après. Vous pourrez voir que le html est plus que minimaliste… vous avez les connaissances de 1ère pour en faire quelque chose de correct ! Ces captures d'écran montrent des versions de travail, avec les requêtes affichées pour vérification. Les résultats sont affichés de manière incomplète volontairement. Les codes html/php sont donnés dans le dossier compressé du devoir.



Page de sélection des attributs et conditions



Page d'affichage des résultats

   b. Installer WAMP (sous Windows)
   Installation :
   - suivre avec attention les instructions lors de l'installation ; pour une fois il faut les lire !
   - Notamment :
     - Présence des différentes versions de Visual C++. Lien pour l'outil disponible dans le premier écran d'installation (logiciel check_vcredist.exe). Le lancer. S'il manque des logiciels, les installer **avant** wampserver. Les liens sont disponibles là aussi dans le premier écran d'installation, la totalité étant là : http://wampserver.aviatechno.net/files/vcpackages/all_vc_redist_x86_x64.zip .
     - Le répertoire d'installation doit être à la racine du disque, et le nom ne doit contenir ni espaces ni caractères diacritiques.

   Usage de WAMP pour le serveur de bases de données :
   - Lancer wampserver. Une icône « wampmanager » rouge puis orange puis verte apparaît dans le menu accessible par la petite flèche. Si elle ne passe pas au vert, un clic gauche dessus permet de « démarrer les services ».

   Remarque : sous OsX, installer MAMP, sous Linux XAMP (les procédures sont plus simples)

   c. Créer la base de données
   Dans WAMP, sélectionner phpMyAdmin, qui ouvre une page dans le navigateur. Se connecter sous le nom `root`, sans mot de passe, serveur MySQL
   Importer la base de données `bdfilms.db.sql` , à télécharger sur http://www.maths-info-lycee.fr/programmes/bdfilms.db.sql (c'est « fichier de requêtes SQL de construction de la base de données bdfilms » sur l'accueil ou la page de terminale)
   Vous pouvez ensuite vérifier dans phpMyAdmin la présence de la base de données, ainsi que les noms des attributs etc.

3. Version flask
   Préliminaires :
   - Importer flask suivant votre environnement :

        - `pip install flask` , dans une invite de commande Windows. Il est possible que vous deviez auparavant rajouter une variable d'environnement Python, si `pip` n'est pas reconnu comme une commande. Les adresses suivantes donnent la méthode (visez les réponses utilisant les interfaces graphiques) :
          https://stackoverflow.com/questions/7054424/python-not-recognized-as-a-command (1ère réponse)
          https://stackoverflow.com/questions/47539201/python-is-not-recognized-windows-10 (3ème réponse)
          Pour trouver les variables d'environnement, tapez simplement « variables d'environnement » dans la barre de recherche, puis dans la fenêtre « propriétés systèmes », le bouton « variables d'environnement » et « nouvelle ». Suivez ensuite les instructions sur une des adresses précédentes. Il y a d'autres pages sur ce sujet, vous pouvez simplement googler « variable d'environnement Python » ou « Python not recognized », « Python non reconnu » etc. Ce n'est pas compliqué, contrairement à ce que vous pourriez croire.
        - `conda install flask` , dans une invite de commande Anaconda
        - Doc (si nécessaire uniquement, notamment en cas de problème d'installation. Il peut être nécessaire de créer un environnement de travail virtuel) : https://flask.palletsprojects.com/en/1.1.x/installation/
    - Créez un dossier de travail dans lequel vous mettez :
        - Le fichier `exemples_flask.py`
        - Un sous-dossier templates, avec `accueil.html`. et `resultats.html`.
        - Dans ce sous-dossier, le fichier `accueil.html`. Ouvrez ce fichier dans un éditeur, vous verrez qu'il y a du code de langage de programmation avec une boucle `for`, ce qui n'existe pas en html.
        - Ce code est du jinja : un vrai langage de programmation, intégré au html. Doc : https://jinja.palletsprojects.com/en/2.11.x/
        - Et surtout : https://jinja.palletsprojects.com/en/2.11.x/templates/ qui est la doc pour la construction d'un document html « template ».
    - Une fois le serveur flask lancé, en exécutant le programme `serveur_flask_jinja.py`, allez dans le navigateur et saisissez comme adresse `localhost :5000/accueil`. La page accueil.html du dossier template s'ouvre.
    - Le serveur flask s'arrête avec ctrl-C (dans le shell Python)
- Importer un connecteur SQL.
    - Comme ci-dessus, suivant votre environnement :
      ```
      pip  install mysql-connector-python
      conda install mysql-connector-python
      ```
    - Utiliser ensuite l'exemple donné dans le fichier `connecteur_sql.py`

4. Version php

Usage de WAMP pour le php

- Créer un projet, en ajoutant un dossier dans le dossier www de wampserver (sans espaces ni tiret bas). Y mettre les fichiers html, css, php etc.
- Un clic gauche sur l'icône wampmanager permet d'accéder au menu « virtualhost » puis « gestion virtualhost ». Ceci lance une page php, suivre les indications pour remplir le formulaire. Une autre possibilité est de taper localhost dans le navigateur.
- Comme indiqué après complétion du formulaire, redémarrer le serveur DNS ; accès par clic droit sur l'icône wampmanager.
- Pour que les pages PHP fonctionnent correctement, il est indispensable de passer par le menu « virtualhost » de wampmanager. Un clic sur l'hôte virtuel désiré ouvre une page html donnant accès aux différentes pages du projet, et permet de les ouvrir ainsi dans le navigateur.
- Ouvrir les pages directement depuis le système de fichiers ne lance pas l'interpréteur PHP ; ça n'est pas fonctionnel.

Fichiers d'exemples pour cette version: `rechercheFilms.html` et `listeAttributs1975.php`
Vous trouverez aussi quelques scripts élémentaires en PHP dans le dossier compressé pour le devoir.