Base de données : introduction

Exercice 1 : **Decouverte : création d'une base de données**

Nous allons travailler dans un premier temps avec le logiciel de gestion de base de données **SQLiteBrowser**. C'est un outil graphique pour gérer nos bases.
Les bases de données SQLite tiennent sur un fichier de petite taille que vous pouvez simplement copier d'un ordinateur à un autre.

- Télécharger la version adaptée du logiciel SQLiteBrowser : https://sqlitebrowser.org/dl/
  - Pour un usage sur l'ordinateur du lycée, utiliser la version : **Windows PortableApp**

Notation utilisée:

- CL : Click,
- CLDR : Click droit,
- DBCL : Double click

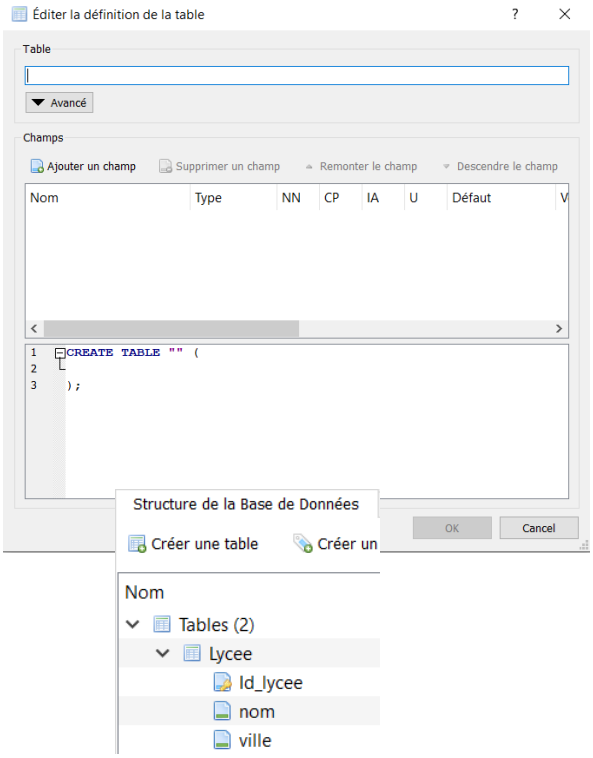
Partie 1 : Premier Pas : création d'une base de données

Nous allons créer la base de données de gestion des élèves dans un lycée (équivalent Pronote).

- CL sur *Nouvelle Base de donnée*.
- Donner le nom **Gestion_eleve** à votre base de type *SQLite3 DB*. Vous l'enregistrerez dans vos documents. La base est créée.

Attention :
Tout comme en python, on évitera d'utiliser des caractères accentués (à,ê,etc.), des caractères spéciaux (#,-, %,etc.) ou des espace dans le noms des bases, des table et des attributs.

Partie 2 : Manipulation : Création de la table LYCEE



- La fenêtre de création de table s'ouvre.
- Le *nom* sera LYCEE.
- CL sur *Ajouter Champs* :
  - *Nom Champ* : **id_lycee**
  - *Type Champ* : choisir dans la liste INTEGER
- Ne rien mettre dans Valeur défaut
- Cocher *Clé Primaire* et *Auto Increment*

Faites de même pour l'attribut **nom** :

- de *nom* nom, type *TEXT* et cocher *NOT NULL*

puis l'attribut **ville** :

- de *nom* ville, type *TEXT* et cocher *NOT NULL*
- puis valider en CL sur *OK*.

La table devrait apparaître dans l'arborescence à gauche.

Remarques

1. En cochant Clé Primaire et Auto Incrément pour le champs id_lycee, on a bien sûr précisé qu'il s'agissait de la clé primaire de la table mais aussi que ce champs serait automatiquement incrémenté à chaque nouveau tuple inséré dans la table.
2. Pour nom et ville nous avons coché NOT NULL, cela signifie que l'on ne peut insérer de tuple dans la table sans donner une valeur à ces champs.
3. On a pu voir la requête SQL qui aurait permis de créer la table LYCEE. On peut la retrouver en faisant un CLDR sur LYCEE puis « Copier l'instruction CREATE ».

Partie 3 : Finalisation de la base

Nous allons maintenant créer les autres tables.

Table CLASSE

On peut facilement créer la table avec la requête suivante :
CREATE TABLE CLASSE
(id_classe **INTEGER PRIMARY KEY NOT NULL** ,
nom **TEXT NOT NULL**,
niveau **INTEGER NOT NULL**)

Manipulation

Saisir la requête précédente dans la première zone de l'onglet « *Exécuter le SQL* » puis CL sur .
(Cette requête modifie la base de données. Elle ne revoit aucun résultat).

Table ELEVE

Créer la table *ELEVE* avec les champs suivants (par l'assistant ou en construisant la requête):
id_eleve : *TEXT PRIMARY KEY NOT NULL* ;
nom : *TEXT NOT NULL* ;
naissance : *TEXT* ;
email TEXT;
sexe : *INTEGER DEFAULT '0' NOT NULL* ;
adresse : *TEXT*;
telephone : *TEXT*;
id_lycee : *INTEGER DEFAULT '1'* ;
note_moyenne_bac : *NUMERIC*.

Remarque

- *sexe* a comme valeur par défaut 0 qui correspond au sexe masculin (toujours hélas plus de garçons que de filles dans les filières d'ingénieurs)
- *telephone* est de type *TEXT* car un numéro à toujours 10 chiffres. Ce serait une erreur de le stocker sous forme de nombre.
- *id_lycee* correspond au lycée de terminale avec comme valeur par défaut 1 qui signifiera "lycée inconnu".
- *note_moyenne_bac* est de type *NUMERIC*, ce qui signifie nombre à virgule.

Tables issues d'associations

Les tables qui représentent des associations sont plus complexes car elles font intervenir des clés étrangères.

Voici la requête SQL pour la table ELEVE_CLASSE :

```
CREATE TABLE ELEVE_CLASSE (
id_eleve INTEGER NOT  NULL,
id_classe INTEGER NOT  NULL,
annee INTEGER NOT  NULL,
boursier INTEGER DEFAULT 0,
PRIMARY KEY  (id_eleve,id_classe,annee),
FOREIGN KEY  (id_classe)
      REFERENCES CLASSE
      ON DELETE CASCADE
      ON UPDATE CASCADE,
FOREIGN KEY  (id_eleve)
      REFERENCES ELEVE
      ON DELETE CASCADE
      ON UPDATE CASCADE)
```

Partie 4 : Contenu de la base

L'objectif de cette partie est d'insérer des enregistrements dans les tables.

Manipulation : Par l'ihm

- Dans l'onglet « Parcourir les données », sélectionner la table CLASSE dans la liste déroulante
- CL sur le bouton « Nouvel enregistrement » *pour* insérer un tuple *nom* : PSI*, *niveau* : 2
- Modifier le contenu d'une donnée dans le cadre sur le côté droit de la fenêtre
- Appuyer sur « Valider » pour mémoriser la saisie (remplir d'id_classe si besoin)

Recommencer pour les tuples : PSI*,2 ; PCSI1,1 ; PT*,2

Remarque

On ne saisit pas la valeur de *id_classe* car il est définit **AUTO INCREMENT**.

Cette façon d'insérer des données semble simple et pratique mais elle devient très vite pénible lorsque l'on doit ajouter une grande quantité de données. Il est alors possible d'utiliser des requêtes (pré enregistré) ou une importation depuis un fichier de sauvegarde.

Par requête INSERT

Dans la zone Requête SQL, écrire la requête :

INSERT INTO CLASSE (nom,niveau) VALUES (' PTSI2 ', ' 1 ') ;

Faire de même pour ajouter la classe PT2, 2.

Par importation

Il est possible pour de grandes quantités de données par import des données se trouvant dans un fichier.

- CL sur le menu *Ficher* puis *Importer/Table depuis un fichier CSV*.
- choisir le fichier **BASE2223.csv** fourni.
- Choisir de l'importer dans la table de destination ELEVE

- Préciser comme caractères de séparation le **;** (cela peut être vérifié au préalable en ouvrant le fichier avec le bloc note)
- Cocher la case « Nom des colonnes en 1ère ligne »
- Importer !

Question : synthétiser la structure de la base de données à l'aide d'une représentation graphique.

Exercice 2 : Base de données météos

Le fichier « Noaa2012.db3 » contient des relevés météolo des stations météo situées aux USA en 2012.

- Copier ce fichier dans vos documents et l'ouvrir avec « SQLiteBrowser »

Q1: Décrire les tables et leur schéma relationnel utilisé.

A partir de la table « `station` », en écrivant des requêtes SQL,:

Q2: Donner la liste des altitudes du sol pour chaque stations

Q3: Donner toutes les propriétés des stations situées au texas (TX)

Q4: Donner la liste des altitudes au sol des stations situées au texas (TX)

- triées dans l'ordre croissant ;
- uniquement les 10 premières ;
- uniquement les 10 suivantes.

A partir de la table « `Moyennes` », en écrivant une requête SQL :

Q5: Donner pour tous les enregistrements l'écart de température (entre la température maximale et la température minimale) ;

Q6: Donner le numéro de la station et les écarts de température uniquement pour des écarts supérieurs à 10° ;

Q7: Donner les enregistrements dont la température minimale est supérieure à la moyenne des températures minimales (question pour un 5/2).

Exercice 3 : SQlite via python

Il est possible de se servir de SQLite dans Python.

- Lancer Spyder
- Récupérez le fichier cine1.s3db et le copier dans le répertoire de travail de Spyder.
- Essayez le script suivant :

```
7
8 import sqlite3 as sql
9 con = sql.connect("cine1.s3db")      #connecter le programme à la base de données
10 cur=con.cursor()                    #créer un "curseur" pour lire ou écrire dans la base
11 cur.execute('SELECT * FROM Film')   #executer une requête SQL via le curseur
12 rows = cur.fetchall()               #récupérer le resultat de la requête
13 for row in rows:
14     print(row)
15
16 con.close()                         #déconnecter la base de données
```

Sur cette base :

Q1: Afficher la liste des horaires des différents films

Q2: Afficher la liste des noms de salle

Q3: La liste des séances dans la salle « Le Trianon »

Q4: Afficher la liste des films de Mathieu Amalric

Q5: Créer un programme python qui fournira les lieux et heures des séances d'un film dont le titre a été indiqué par l'utilisateur.