Bases de données et SQL : exercices 1 avec correction

1 Prise en main du logiciel

1.1 Création d'un fichier de base de données

Ouvrir SQLiteStudio ; cliquer sur Database > Add a database, ou taper Ctrl-O.
Appuyer sur le « + » vert ; ceci vous permet de créer un nouveau fichier de base de données (l'autre option étant d'ouvrir un fichier existant). Créer un fichier que vous appelez comme vous voulez (avec une extension `.db` pour qu'il apparaisse dans les menus, par exemple `Mabase.db`). Prendre le type de base « SQLite 3 » . Spécifier un répertoire où vous voulez enregistrer ce fichier : typiquement pas très loin de vos scripts Python. Attention utilisateurs de Mac, il propose par défaut d'enregistrer dans un répertoire sur lequel vous n'avez pas les droits d'écriture !!!

Votre base (vide pour l'instant) apparaît dans la colonne de gauche. Cliquer droit dessus, et sélectionner « Connect to the database » .
La petite icône s'éclaire : vous êtes connecté à la base de données. En double-cliquant dessus, vous déroulez un menu en-dessous comportant la rubrique « Tables » . En cliquant droit là-dessus, vous avez une option « Create table » , qui permet de créer une table. Pour cette fois nous allons plutôt le faire à la main, avec des instructions SQL.

Aller dans le menu Tools > SQL Editor (Alt-E). Vous ouvrez un champ de requêtes où vous pouvez taper des instructions qui seront effectuées par SQL.
On cherche à répertorier un ensemble de membres de groupes de musique, leur instrument, leur nationalité et leur pays d'origine.
On veut donc créer une table dont les champs sont :

- Nom (champ en format `TEXT`)
- Groupe (champ en format `TEXT`)
- Instrument (champ en format `TEXT`)
- Chant (champ en format `INTEGER`) : vaudra 1 si le musicien est chanteur dans son groupe, et 0 sinon.
- Pays (champ en format `TEXT`)
- Âge (en format `INTEGER`).

Exercice 1. Créer cette table, qu'on nommera Musiciens à partir de l'instruction `CREATE TABLE`.

```
CREATE TABLE Musiciens (
Nom TEXT,
Groupe TEXT,
Instrument TEXT,
Chant INTEGER,
Pays TEXT,
Age INTEGER)
```

Vous remarquez l'apparition d'un (1) à côté des Tables de la colonne de gauche : votre table est créée.
En double-cliquant sur Tables, vous déroulez la liste des tables (qui ne comporte donc que « Musiciens » pour l'instant).
En double-cliquant sur Musiciens, vous voyez apparaître des champs d'édition de la table.

- L'onglet « Structure » permet de renommer la table, liste les colonnes, leurs domaines et leurs caractéristiques (clé primaire, clé étrangère...). Vous pouvez ajouter ou enlever des colonnes, éditer leur caractéristiques en cliquant dessus, etc.
- L'onglet « Données » permet de peupler votre table avec des données. Le « + » vert permet d'ajouter une ou plusieurs lignes, qu'on remplit comme un tableau Excel.
  Attention il faut sauvegarder ses modifications avec le petit ✓ vert.

Tout en bas de la fenêtre vous voyez la liste des onglets ouverts (Éditeur SQL, tables consultées et la base de données à laquelle elles appartiennent...) Pensez à naviguer entre ces onglets au lieu d'en ouvrir de nouveaux.

Ensuite, l'éditeur SQL accessible par Tools > Open SQL Editor permet d'effectuer des requêtes sur cette table, avec la syntaxe développée dans le cours.
Attention, si ces requêtes modifient la table, les modifications ne sont pas immédiatement visibles : il faut cliquer sur le 🔄 bleu.

On veut maintenant remplir cette table, en s'intéressant à des groupes musicaux.

Exercice 2. Peupler la table avec le groupe Sonic Youth (USA) dont les membres, sont :

- Thurston Moore (Guitare, Chant), 50 ans
- Lee Ranaldo (Guitare, Chant), 52 ans
- Kim Gordon (Basse, Chant), 47 ans
- Steve Shelley (Batterie), 45 ans

On utilisera les commandes `INSERT INTO ... VALUES ...`

```
INSERT INTO Musiciens
VALUES
('Thurston Moore','Sonic Youth','Guitare',1,'USA',50)
('Lee Ranaldo','Sonic Youth','Guitare',1,'USA',52)
('Kim Gordon','Sonic Youth','Basse',1,'USA',47)
('Steve Shelley','Sonic Youth','Batterie',0,'USA',45)
```

2 Manipulation de données : modification de la table et recherche d'informations

Nous allons manipuler une version plus étoffée de cette base de données.

1. Récupérer le fichier `musique1.db` sur Cahier de Prépa.
2. Dans le menu Database > Add a database, de SQLiteStudio, ouvrir le fichier récupéré. Il apparaît dans la colonne de gauche.
3. Dans le menu Database, sélectionner « Connect to Database » . Le contenu de la base apparaît en-dessous.

Intéressons-nous pour l'instant à la Table `Musiciens`, qui est celle que vous venez de créer, augmentée de plusieurs enregistrements : vous trouverez la liste des membres de divers groupes musicaux. Retourner dans l'éditeur de requêtes SQL (Alt-E), et écrire des instructions permettant d'effectuer les opérations / recherches suivantes :

Exercice 3. 1. Afficher toutes les informations disponibles sur les groupes français.

```
SELECT *
FROM Musiciens
WHERE Pays='France'
```

2. Afficher les noms et âges de tous les chanteurs.

```
SELECT Nom,Age
FROM Musiciens
WHERE Chant=1
```

3. Afficher les noms de tous les chanteurs américains.

```
SELECT Nom
FROM Musiciens
WHERE Chant=1 AND Pays='USA'
```

4. Afficher les noms des groupes américains. Pour éviter la répétition des noms de groupe, on pourra utiliser la commande `SELECT DISTINCT...` qui renvoie la série de valeurs distinctes correspondant aux données sélectionnées.

```
SELECT DISTINCT Groupe
FROM Musiciens
WHERE Pays='USA'
```

5. Afficher tous les noms, âges et groupes des guitaristes, triés par âge croissant (commande `ORDER BY`).

```
SELECT Nom,Age,Groupe
FROM Musiciens
WHERE Instrument='Guitare'
ORDER BY Age
```

6. La bassiste de Sonic Youth quitte son groupe, et devient guitariste de Sepultura. Modifier la table en conséquence.
On donnera une commande SQL qui n'utilise pas le nom de cette bassiste.

```
UPDATE Musiciens
SET Instrument='Guitare',
Groupe='Sepultura'
WHERE
Instrument='Basse' AND Groupe='Sonic Youth'
```

7. Sonic Youth recrute donc un nouveau bassiste, appelé Krist Novoselic, de nationalité américaine, âgé de 22 ans. Rajouter cet élément dans la table.

```
INSERT INTO Musiciens
VALUES
('Krist Novoselic','Sonic Youth','Basse',0,'USA',22)
```

8. Le groupe Fugazi se sépare. Enlever de la table tous les enregistrements qui le concerne.

```
DELETE FROM Musiciens
WHERE Groupe='Fugazi'
```

9. Le groupe Tomb Mold décide de se rebaptiser Blood Incantation. Modifier la table en conséquence.

```
UPDATE Musiciens
SET Groupe='Blood Incantation'
WHERE Groupe='Tomb Mold'
```

10. Le bassiste de Sepultura désire adopter le nom « Paulo Destructor » . Modifier la table en conséquence (on n'utilisera pas le nom de ce bassiste dans la requête).

```
UPDATE Musiciens
SET Nom='Paulo Destructor'
WHERE Groupe='Sepultura' AND Instrument='Basse'
```

3 Jointures : utilisation de plusieurs tables

On souhaite maintenant stocker plus d'informations sur les groupes : date de création, nombre d'albums publiés, pays d'origine (ceci évitera les redondances vues dans la première table).
On crée alors une nouvelle base de données.
Celle-ci contient une table `Groupes`, contenant une clé primaire `id`, le nom du groupe, la date de création, le nombre d'albums publiés ; et une table `Musiciens2` où la référence au groupe se fasse via une clé étrangère liée à la clé primaire identifiant les groupes (dans la table `Groupes`, donc).

Exercice 4. Donner les syntaxes `CREATE TABLE` permettant de créer ces deux tables. On commencera par créer la table des groupes, munie d'une clé primaire ; puis on créera la liste des musiciens en spécifiant une clé étrangère rattachée à la première table.
La table `Groupes` comprend donc :

- une colonne `id` (format `INTEGER`, clé primaire)
- une colonne `Nom_groupe` (format `TEXT`)
- une colonne `Pays` (format `TEXT`)
- une colonne `Genre` (format `TEXT`)
- une colonne `Nombre_albums` (format `INTEGER`)
- une colonne `Date_formation` (format `INTEGER`)

```
CREATE TABLE Groupes (
id  INTEGER PRIMARY KEY,
Nom_groupe  TEXT,
Pays  TEXT,
Genre  TEXT,
Nombre_albums  INTEGER,
Date_formation  INTEGER
)
```

et la table `Musiciens2` comprend les colonnes :

- une colonne `Nom` (format `TEXT`)
- une colonne `Groupe` (format `INTEGER`, clé étrangère rattachée au champ `id` de la table Groupes)
- une colonne `Instrument` (format `TEXT`)
- une colonne `Chant` (format `INTEGER`)
- une colonne `Age` (format `INTEGER`)

```
1  CREATE TABLE Musiciens2 (
2  Nom  TEXT,
3  FOREIGN KEY Groupe REFERENCES Groupes(id),
4  Instrument  TEXT,
5  Chant INTEGER,
6  Age INTEGER)
```

NB : C'est la syntaxe au programme ; sur SQLiteStudio on écrira la ligne 3 comme :

```
1  Groupe REFERENCES Groupes(id),
```

On suppose ces tables remplies, comme elles le sont sur le fichier `musique2.db` disponible sur Cahier de Prépa dès à présent.
Dans l'exercice suivant, on lit diverses informations dans cette table à l'aide de requêtes SQL. On aura souvent à utiliser une jointure entre les deux tables... mais pas toujours !

Exercice 5.

1. Compléter la condition de jointure entre ces deux tables, et afficher le résultat :

```
1  SELECT *
2  FROM Musiciens2 INNER JOIN Groupes ON Groupes.id = Musiciens2.Groupe
```

2. Afficher toutes les infos disponibles sur les musiciens de Hip Hop.

3. Afficher la liste des noms et âges des guitaristes de Rock.

```
1  SELECT Nom,Age
2  FROM Musiciens2 INNER JOIN Groupes ON Groupes.id = Musiciens2.Groupe
3  WHERE Genre='Rock' AND Instrument='Guitare'
```

4. Afficher les noms et groupes des batteurs de Metal ayant publié strictement plus de 10 albums.

```
1  SELECT Nom,Nom_groupe
2  FROM Musiciens2 INNER JOIN Groupes ON Groupes.id = Musiciens2.Groupe
3  WHERE Genre='Metal' AND Instrument='Batterie' AND Nombre_albums>10
```

5. Combien d'albums d'Electro ont été publiées par les groupes de cette base de données ?

```
1  SELECT  SUM(Nombre_albums)
2  FROM Groupes
3  WHERE Genre='Electro'
```

6. Donner le nombre moyen d'albums publiés par les groupes de Rock de cette base de données.

```
1  SELECT  AVG(Nombre_albums)
2  FROM Groupes
3  WHERE Genre='Rock'
```

7. Retrouver à partir des tables `Musiciens2` et `Groupes` uniquement, la liste des chanteurs de Rock classés par âge décroissant, avec leurs âges et leurs groupes respectifs.

```
1  SELECT Nom,Age,Nom_groupe
2  FROM Musiciens2 INNER JOIN Groupes ON Groupes.id = Musiciens2.Groupe
3  WHERE Chant=1
4  ORDER BY Age
```

8. Donner la liste des chanteurs de cette table, classés par ordre décroissant du nombre d'albums publiés. On affichera aussi le nombre d'albums.

```
SELECT Nom ,Nombre_albums
FROM Musiciens2 INNER JOIN Groupes ON Groupes.id = Musiciens2.Groupe
WHERE Chant=1
ORDER BY Nombre_albums DESC
```

9. Compter le nombre de chanteurs-bassistes.

```
SELECT COUNT(*)
FROM Musiciens2
WHERE Chant=1 AND Instrument='Basse'
```

10. Afficher les pays d'origine de ces chanteurs-bassistes.

```
SELECT Pays
FROM Musiciens2 INNER JOIN Groupes ON Groupes.id = Musiciens2.Groupe
WHERE Chant=1 AND Instrument='Basse'
```

11. Afficher la liste des groupes et de leur nombre d'années d'existence (rappel : nous sommes en 2024)

```
SELECT Nom_groupe , 2024-Date_formation
FROM Groupes
```

12. Afficher une liste des noms des musiciens, et de leur âge au moment de la formation de leur groupe[1].

```
SELECT Nom,Nom_groupe,Age-(2024-Date_formation)
FROM Musiciens2 INNER JOIN Groupes ON Groupes.id = Musiciens2.Groupe
```

13. Afficher les pays d'origine des batteurs de strictement moins de 40 ans. On n'affichera qu'une seule fois chaque pays.

```
SELECT DISTINCT Pays
FROM Musiciens2 INNER JOIN Groupes ON Groupes.id = Musiciens2.Groupe
WHERE Instrument='Batterie' AND Age<40
```

14. .....

[1] qui n'est pas forcément l'âge auquel ils ont rejoint ce groupe...