TP1 : Interrogation et mise à jour de BDD

Dans ce TP nous allons découvrir le **langage SQL** (Structured Query Language) qui est le langage utilisé pour effectuer des requêtes sur une base de données relationnelles.

Pour manipuler des bases de données, nous utiliserons le logiciel **DB Browser**. Ce logiciel propose une interface graphique (GUI) permettant la gestion de base de données et l'écriture de requêtes SQL. Il s'appuie sur le SGDB **SQLite3**, qui est un gestionnaire de base de données relationnelles léger et simple.

Vous devrez d'abord installer la version portable (pour Windows) de DBBrowser sur une clef USB à apporter à chaque séance !
Téléchargement : `https://sqlitebrowser.org/dl/`

1 Analyse préliminaire de la base de données

1.1 Sur quoi allons-nous travailler ?

Nous allons manipuler une base de données permettant de traduire efficacement les informations contenues dan ce tableau résumé (*exemple du cours*) :

Titre	Nom Auteur	Prenom Auteur	Annee Naissance	Langue	Annee Publi	Themes
1984	Orwell	George	1903	Anglais	1949	Totalitarisme, science-fiction, anticipation, Dystopie
Dune	Herbert	Frank	1920	Anglais	1965	science-fiction, anticipation
Fondation	Asimov	Isaac	1920	Anglais	1951	science-fiction, Economie
Le meilleur des mondes	Huxley	Aldous	1894	Anglais	1931	Totalitarisme, science fiction, dystopie
Fahrenheit 451	Bradbury	Ray	1920	Anglais	1953	science-fiction, Dystopie
Ubik	K. Dick	Philip	1928	Anglais	1969	science-fiction, anticipation
Chroniques martiennes	Bradbury	Ray	1920	Anglais	1950	science-fiction, anticipation
La nuit des temps	Barjavel	René	1911	Français	1968	science-fiction, tragédie
Blade Runner	K. Dick	Philip	1928	Anglais	1968	Intelligence artificielle, science fiction
Les Robots	Asimov	Isaac	1920	Anglais	1950	science fiction, Intelligence artificielle
La Planète des singes	Boulle	Pierre	1912	Français	1963	science fiction, Dystopie
Ravage	Barjavel	René	1911	Français	1943	Science-Fiction, anticipation
Le Maître du Haut Château	K. Dick	Philip	1928	Anglais	1962	Dystopie, Uchronie
Le monde des A	Van Vogt	Alfred	1912	Anglais	1945	science fiction, IA
La Fin de l'éternité	Asimov	Isaac	1920	Anglais	1955	science-fiction, voyage dans le temps
De la Terre à la Lune	Verne	Jules	1828	Français	1865	Science-Fiction, aventure

J'ai déjà créé la base et l'ai partiellement remplie.
Elle est contenue dans le fichier `livres-preremplissage.db` disponible sur l'ENT.
Commencer par créer une copie de ce fichier dans un dossier de votre espace personnel en le renommant `livres.db`.
Ouvrir cette base dans DBBrowser.

L'objectif est d'interroger et de faire des mises à jour dans cette base. Cette base est constituée de 5 tables :

- Auteur
- Livre
- Langue
- Theme
- RelationLivreTheme

1.2 Schéma des tables

Le schéma des tables apparaît dans l'onglet « Structure de la base de données ».

Prendre le temps de consulter le schéma détaillé de chaque table : observer en particulier les types des attributs et relever les clefs primaires (PRIMARY KEY) ou clefs étrangères (FOREIGN KEY) dans chaque table.

Par exemple, l'attribut IdLangue est la clef primaire de la relation Langue. C'est aussi une clef étrangère pour la relation Auteur.

1.3 Lire le contenu d'une table : SELECT * FROM Table ;

L'onglet « Parcourir les données » permet de visualiser assez rapidement le contenu des tables. Mais nous nous interdirons cette façon de procéder car cette facilité proposée par DBBrowser cache en fait la mise en œuvre d'une requête SQL.

Toutes les requêtes SQL sont écrites et exécutées dans l'onglet « Exécuter le SQL ».

Nous allons écrire une première **requête d'interrogation** avec la commande **SELECT** afin de récupérer le contenu d'une table. Ces requêtes peuvent être beaucoup plus sophistiquées comme on le verra dans le 2ème TP. Pour le moment, nous nous contenterons de la forme la plus simple :

```
SELECT * FROM Langue;
```

On voit ainsi apparaître le contenu de la table.

Attention : Les requêtes SQL **se terminent toujours par un point-virgule**.
Elles peuvent s'écrire sur plusieurs lignes.
Elles sont non sensibles à la casse, mais c'est une bonne pratique que d'écrire toutes les commandes SQL en majuscules.

2 Insérer des enregistrements : INSERT INTO Table VALUES (données) ;

2.1 Exemple : table Langue

Pour remplir une table avec des données, il faut utiliser une requête **INSERT**.
Voici par exemple comment j'ai rempli la table **Langue** :

```
INSERT INTO Langue VALUES (1, "Anglais");
INSERT INTO Langue VALUES (2, "Français");
```

On peut refaire une autre requête **INSERT** à la suite si on souhaite ajouter encore des données au bout de la table.
 Ajouter ainsi une langue Espagnol et Allemand. Puis afficher (requête d'interrogation) le contenu de la table.

Il est possible d'omettre un attribut en lui affectant une valeur nulle (NULL) sauf si la création de la table spécifie NOT NULL (auquel cas l'attribut est obligatoire).

 Essayer d'exécuter la requête suivante :

```
INSERT INTO Langue VALUES (2, "Turc");
```

Expliquer l'origine du message d'erreur qui apparaît.

2.2 Table Auteur

 Compléter la table **Auteur** afin que celle-ci reflète les informations suivantes (penser à revoir le schéma de la table) :

Nom	Prenom	annee naissance	langue
Orwell	George	1903	Anglais
Herbert	Frank	1920	Anglais
Asimov	Isaac	1920	Anglais
Huxley	Aldous	1894	Anglais
Bradbury	Ray	1920	Anglais
K. Dick	Philip	1928	Anglais
Barjavel	René	1911	Français
Boulle	Pierre	1912	Français
Van Vogt	Alfred Elton	1912	Anglais
Verne	Jules	1828	Français

Une nouveauté apparaît ici : la table Auteur possède une **clef étrangère** définie par la ligne :

```
FOREIGN KEY(IdLangue) REFERENCES Langues(IdLangue)
```

Celle-ci permet de déclarer une **contrainte de référence** sur cette clef afin d'indiquer à SQLite que IdLangue est une clef étrangère. SQLite sera alors responsable de maintenir la cohérence entre les deux tables que l'on a ainsi reliées (autre rôle du SGBD).
Si on essaie d'entrer un enregistrement avec une valeur qui n'existe pas pour une clef étrangère, le SGBD envoie un message d'erreur.

2.3 La table Livre

1. Observer le schéma de la table **Livre** et afficher son contenu avec une requête SELECT.
2. Ajouter les 3 livres manquants : "La planète des singes", "Le monde des A" et "De la Terre à la Lune".
   (*les informations nécessaires sont résumées dans le tableau présenté en début de TP*)
3. Aurait-on pu créer ces livres avant d'avoir complété la table Auteur ? Justifier.

2.4 La table Theme

Traitons à présent la problématique des Thèmes. Cette table doit lister tous les thèmes qui peuvent correspondre aux livres de la base.

1. Observer le schéma de la table **Theme** et afficher son contenu avec une requête SELECT.
2. Trouver le thème qui a été oublié, et l'ajouter à la table.
   (*les informations nécessaires sont résumées dans le tableau présenté en début de TP*)

2.5 Une dernière table manquante

La saisie de notre base de donnée est incomplète ! Nous avons en effet saisi tous les auteurs, tous les livres, toutes les langues, tous les thèmes et pourtant il manque une information. Laquelle ?

Pour solutionner ce problème, la table **RelationLivreTheme** contient l'information manquante en mettant en relation les livres et les thèmes associés.

1. Observer le schéma de la table **RelationLivreTheme** et afficher son contenu avec une requête SELECT.
2. Quelle est sa clef primaire ? Quelle est particularité par rapport aux clefs primaires qu'on a rencontrées dans les autres tables ?
3. Possède-t-elle des clefs étrangères ? Si oui, lesquelles ?
4. Ajouter à cette table les enregistrements nécessaires pour que les Livres qui font références au thème "Intelligence artificielle" soient correctement pris en compte.
5. Enfin, ajouter à cette table les enregistrements décrivant les thèmes des 3 livres ajoutés plus tôt.

2.6 Contrôle du travail jusqu'à présent

Vérifier que les livres sont bien associés aux bons thèmes en recopiant la commande suivante (*explications de cette requête dans la suite du TP...*) :

```
SELECT Titre, Intitule FROM RelationLivreTheme
JOIN Livre ON Livre.IdLivre = RelationLivreTheme.IdLivre
JOIN Theme ON Theme.IdTheme = RelationLivreTheme.IdTheme;
```

Vérifier les autres informations sur les livres en recopiant la commande suivante (*explications de cette requête dans la suite du TP...*) :

```
SELECT DISTINCT
Titre, NomAuteur, PrenomAuteur, AnneeNaissance, Langue, AnneePubli
FROM RelationLivreTheme
JOIN Livre ON Livre.IdLivre = RelationLivreTheme.IdLivre
JOIN Auteur ON Auteur.IdAuteur = Livre.IdAuteur
JOIN Langue ON Langue.IdLangue = Auteur.IdLangue;
```

3 Requêtes d'interrogation : SELECT

Nous avons déjà rencontré la requête SELECT pour afficher tous les enregistrements d'une table. Nous allons voir plus précisément comment cette requête d'interrogation se manipule.

3.1 Sélection simple : clause WHERE

Pour être sûr de travailler sur une base correcte, téléchargez depuis l'ENT le fichier `livres-fin_partie_1.db`.
Supposons que l'on veuille lister seulement les noms et prénoms des auteurs nés avant 1900, on utilisera une clause **WHERE** `condition` :

```
SELECT NomAuteur, PrenomAuteur FROM Auteur WHERE AnneeNaissance < 1900;
```

Pour obtenir les auteurs prénommés Jules :

```
SELECT NomAuteur, PrenomAuteur FROM Auteur WHERE PrenomAuteur = "Jules";
```

Bien noter que les textes sont délimités par des guillemets !

Attention : essayez de remplacer "Jules" par "jules" et vous constaterez que la recherche est sensible aux majuscules.

3.2 Sélection multiple

Il est possible de croiser plusieurs critères à l'aide d'opérateurs booléens : **AND** et **OR**.
Voici la liste des auteurs français nés après 1900 :

```
SELECT NomAuteur, PrenomAuteur FROM Auteur
WHERE IdLangue = 2 AND AnneeNaissance > 1900;
```

3.3 COUNT : compter le nombre de réponses d'une requête

Combien y a t-il d'auteurs nés entre 1900 et 1915 ?

```
SELECT COUNT(*) FROM Auteur WHERE AnneeNaissance > 1900 AND AnneeNaissance < 1915;
```

3.4 Trier les réponses : ORDER BY

Nous allons lister tous les auteurs par ordre croissant d'année de naissance :

```
SELECT * FROM Auteur ORDER BY AnneeNaissance;
```

et par ordre décroissant, on ajoute **DESC** à la fin de la requête :

```
SELECT * FROM Auteur ORDER BY AnneeNaissance DESC;
```

3.5 Éviter les occurrences multiples : DISTINCT

Interrogeons la table Livres sur les années de publication, rangées par ordre croissant :

```
SELECT AnneePubli FROM Livre ORDER BY AnneePubli;
```

On constate la présence de quelques doublons :
Pour éviter les redondances dans les résultats, on peut rajouter le mot-clef **DISTINCT** juste après le SELECT :

```
SELECT DISTINCT AnneePubli FROM Livre ORDER BY AnneePubli;
```

3.6 Exo défi 1

1. Donner la liste de tous les titres des livres écrits entre 1920 et 1950.
2. Combien y en a t-il ?

4 Requêtes portant sur plusieurs tables

Jusqu'à présent, nos requêtes ne portaient que sur une seule table. Néanmoins notre liste de livres comporte des données en provenance de plusieurs tables simultanément. Nous allons voir comment effectuer des requêtes pour croiser des données en provenance de plusieurs tables.

```
SELECT * FROM Langue, Auteur;
```

Comme on peut le constater cette requête est peu pertinente car elle affiche toutes les données de chacune des tables sans effectuer de correspondances. La **clef de jointure** apparaît pourtant ici clairement : il s'agit de idLangue qui doit permettre de recouper les informations entre les deux tables : il est en effet inutile d'afficher les données pour lesquelles les langues ne correspondent pas entre les deux tables.

4.1 Jointure : JOIN ON

La **jointure** consiste à **croiser les données de plusieurs tables** pour les présenter sous forme d'un seul tableau. On va utiliser ce mécanisme pour afficher clairement la langue de l'auteur plutôt qu'un numéro qui n'est pas forcément très parlant. Nous utiliserons pour cela l'opérateur **JOIN** ... **ON** :

```
SELECT NomAuteur, PrenomAuteur, Langue
FROM Auteur JOIN Langue ON Auteur.IdLangue = Langue.IdLangue;
```

Remarque : on retrouve la syntaxe de la notation de programmation objet avec le point qui sépare le nom d'une relation et le nom d'un attribut de cette relation.

4.2 Exo défi 2

En croisant la table Livre avec la table Auteur :

1. Afficher une liste dont les attributs sont Titre, Prenom, Nom et Annee Publi, triée du plus récent au plus ancien.
2. Afficher une liste dont les attributs sont Titre, Prenom, Nom et Annee Publi écrits en Français (IdLangue = 2).

4.3 Le cas des relations de n à n

Parfois il arrive que les données à collecter se trouvent dans plus que deux relations : c'est le cas des Thèmes pour les livres qui nécessitent l'analyse de 3 tables : Livre et Thème bien sûr, mais aussi la table RelationLivreTheme.

Observer et étudier la requête ci-dessous :
Le principe est d'**enchaîner deux jointures** JOIN ... ON en utilisant la table de Relation au milieu. En effet, la requête se lit de la gauche vers la droite et on ne peut faire de jointure que si on a une clef étrangère en commun, ce qui n'est par exemple pas le cas entre Livre et Thème.

```
SELECT Titre, Intitule FROM Livre
JOIN RelationLivreTheme ON Livre.IdLivre = RelationLivreTheme.IdLivre
JOIN Theme ON Theme.IdTheme = RelationLivreTheme.IdTheme;
```

4.4 Exo défi 3

Écrire une requête permettant d'obtenir une liste dont les attributs sont Titre, Nom et Langue triée par ordre croissant de date de naissance de l'auteur.

5 Mise a jour, effacement : UPDATE et DELETE

5.1 Préliminaire : changeons d'outil...

Pour découvrir un peu d'autres outils de gestion SQL, nous proposons de finir ce TP online. Connectez-vous au site `https://sqliteonline.com/` et ouvrez dans cette interface Web la base `livres-fin_partie_1`. Ce site propose donc d'exécuter des requêtes SQL sans installation.

5.2 UPDATE

Les requêtes **UPDATE** et **DELETE** fonctionnent sur le même modèle que les requêtes d'interrogation SELECT.

Attention ! On a vite fait d'effacer toutes ses données si on ne configure pas bien sa requête. Une bonne habitude à prendre est de **tester d'abord ses critères à l'aide d'un SELECT**.

Observer les exemples ci-dessous ; on désire modifier des informations concernant Jules Verne :

```
SELECT NomAuteur FROM Auteur WHERE IdAuteur = 10;
```

Cette requête SELECT est juste une prudence pour contrôler que c'est bien lui le n°10 !

Modifions le nom de l'auteur grâce à une **requête de mise à jour UPDATE** sur le modèle suivant :

```
UPDATE Table
SET attribut1 = valeur1, attribut2 = valeur2, ...
WHERE critères;
```

```
UPDATE Auteur
SET NomAuteur = "Ze Djoule", PrenomAuteur = "Juju"
WHERE IdAuteur = 10;
```

Affichons les auteurs pour contrôler cette mise à jour : `SELECT * FROM Auteur ;`

5.3 DELETE

Et maintenant essayons vite de supprimer cette entrée !
Nous utiliserons une requête **DELETE FROM Table WHERE critères**.

Attention : soyez bien sûr de votre critère sous peine de perdre des données importantes !

```
DELETE FROM Auteur WHERE IdAuteur = 10;
```

Un bon SGBD ne nous aurait jamais laisser faire cela : pourquoi ?

Dans une base de données relationnelle il faut être vigilant lors de la suppression d'enregistrements : en effet la suppression d'un enregistrement entraîne la suppression de sa clef primaire qui peut être utilisée en tant que clef étrangère dans une autre table. Cela entraîne une corruption des données.

5.4 Exo défi 4

1. Réinsérer l'auteur Jules Verne à sa place.
2. Supprimer tous les livres écrits au 19ème siècle.
   Penser à faire le nécessaire pour conserver une base cohérente !

6 Apprendre et s'exercer en autonomie

Consulter et travailler le tutoriel SQL sur W3Schools `https://www.w3schools.com/sql/`.
Le programme de Terminale NSI s'arrête au tutoriel « SQL Joins ». Les autres tutoriels restent bien entendus intéressants pour ceux qui veulent « aller plus loin » mais sont hors programme.
Avec un bon navigateur supportant WebSQL (Chrome, Opera, Safari), vous pouvez faire tous les exercices.
Le quizz (jusqu'à la question 20) est à savoir faire !