Chapitre 8 - Le langage SQL

Objectifs :

- ▷ Identifier les composants d'une requête.
- ▷ Savoir utiliser des requêtes d'interrogation et de mise à jour d'une base de données.
- ▷ Construire des requêtes d'interrogation à l'aide des clauses du langage SQL : `SELECT`, `FROM`, `WHERE`, `JOIN`.
- ▷ Construire des requêtes d'insertion et de mise à jour à l'aide de : `UPDATE`, `INSERT`, `DELETE`.

1 Introduction

Dans le chapitre précédent sur le modèle relationnel, nous avons découvert la structure mathématique des bases de données. Maintenant que nous avons vu comment celles-ci sont organisées, nous allons interagir avec elles.

Les SGBD jouent le rôle d'interface entre l'être humain et la base de données. Par l'intermédiaire de requêtes, l'utilisateur va **consulter** ou **mettre à jour** la base de données.

Le langage utilisé pour communiquer avec le SGBD est le **langage SQL**, (pour *Structured Query Langage* ou langage de requêtes structurées).

Les SGBD les plus utilisés sont basés sur le **modèle relationnel**. Parmi eux, on peut trouver :

La quasi-totalité de ces SGBD fonctionnent avec un modèle **client-serveur**.

Cette année, nous allons travailler principalement avec le langage SQLite peut lui s'utiliser directement sans démarrer un serveur : la base de données est entièrement représentée dans le logiciel utilisant SQLite. Sa simplicité d'utilisation en fera notre choix pour illustrer cette présentation du langage SQL.

Il existe de nombreuses façons d'écrire des requêtes SQL avec une base de données sqlite :

- ▷ Avec un logiciel muni d'une interface graphique comme *DB Browser for SQLite*.
- ▷ Via un langage de programmation comme Python grâce au module `sqlite3`
- ▷ En console, en lançant : `sqlite3 nom_de_ma_bdd.db`.
- ▷ En ligne avec des sites comme SQLite online ou SQLite Viewer,

2 Le langage SQL

Les instructions SQL s'écrivent d'une manière qui ressemble à celle de phrases ordinaires en anglais. C'est un **langage déclaratif**, c'est-à-dire qu'il permet de décrire le résultat escompté, sans décrire la manière de l'obtenir.

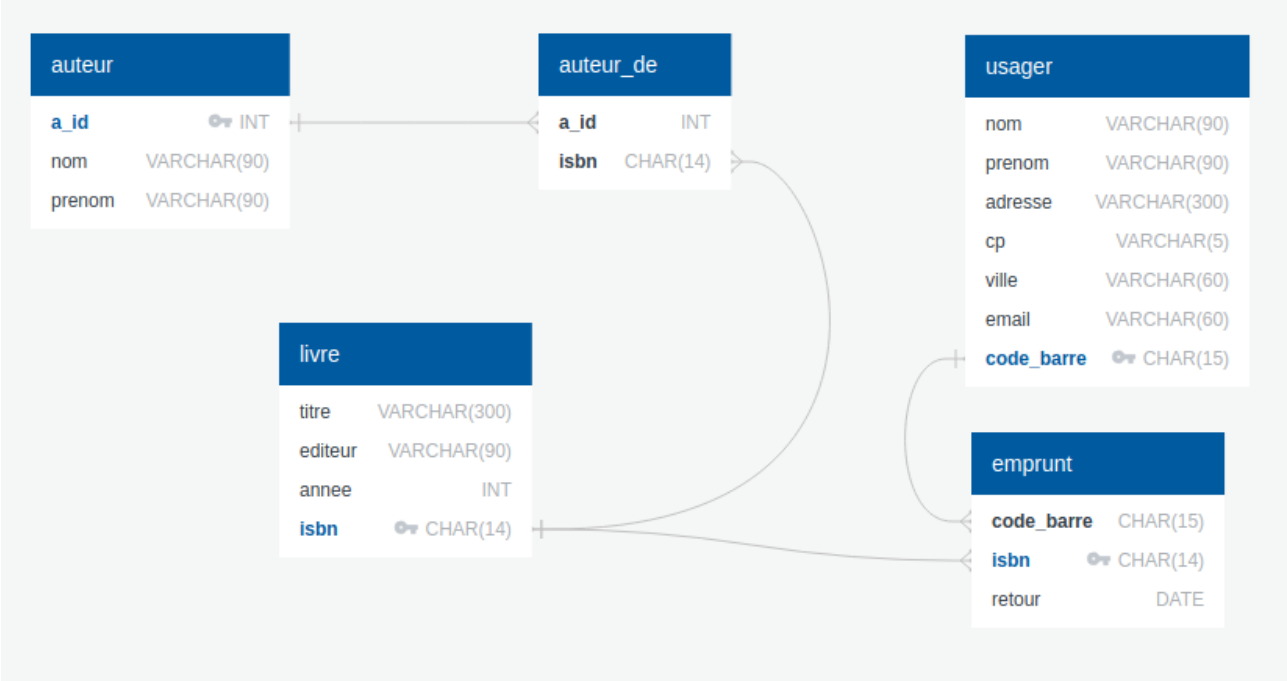
Le langage SQL n'est **pas sensible à la casse**, mais a l'habitude d'écrire les instructions en majuscules, on peut écrire les instructions sur plusieurs lignes avec ou sans indentation et chaque instruction doit être terminée par un **point-virgule**.

Nous verrons cette année les instructions de manipulation du contenu de la base de données qui commencent par les mots clés :

- `SELECT` : recherche de contenu ;
- `UPDATE` : modification,
- `INSERT` : ajout,
- `DELETE` : suppression.

Pour illustrer ce cours, nous étudierons les données situées dans la base de données `livres.db` jointe à votre espace *Classroom*.

Voici le diagramme relationnel de cette base :



— Les clés primaires sont en bleu (suivi d'une icone de clé)
— Les clés étrangères sont en noir et reliées à leur clé primaire.

3 Sélection de données

3.1 Comment interroger une base de données ?

3.1.1 Requête basique : SELECT ... FROM ...

Exemple 0

```
SELECT titre FROM livre ;
```

Traduction :
On veut tous les titres de la table `livre`.

Remarques :

- Les mots-clés SQL sont traditionnellement écrits en MAJUSCULES.
- Le ; signale la fin de l'instruction. Il peut donc être omis s'il n'y a pas d'instructions enchaînées (ce qui sera toujours notre cas).
- L'indentation n'est pas syntaxique (pas comme en Python). On peut donc faire des retours à la ligne et des indentations pour rendre le code plus lisible.

3.1.2 Requête filtrée : SELECT ... FROM ... WHERE ...

Exemple 1

```
SELECT titre FROM livre WHERE annee >= 1990;
```

Traduction :
On veut les titres de la table `livre` qui sont parus après (ou en) 1990 ;

Remarques :

- Le mot-clé WHERE doit être suivi d'un booléen. Les opérateurs classiques `=` , `!=`, `>`, `>=`, `<`, `<=` peuvent être utilisés, mais aussi le mot-clé `IN`.

3.1.3 Requête avec plusieurs possibilités : WHERE ... IN ...

Exemple 1bis

```
SELECT titre FROM livre WHERE annee IN (1990, 1991, 1992);
```

Traduction :
On veut les titres de la table «livre» qui sont parus en 1990, 1991 ou 1992.

3.1.4 Requête avec booléens : AND - OR

Exemple 2

```
SELECT titre FROM livre WHERE   annee >= 1970 AND
                                annee <= 1980 AND
                                editeur = 'Dargaud';
```

Traduction :
On veut les titres de la table `livre` qui sont parus entre 1970 et 1980 chez l'éditeur Dargaud ;

3.1.5 Requête approchée : LIKE

Exemple 3

```
SELECT titre FROM livre WHERE titre LIKE '%Asterix%';
```

Traduction :
On veut les titres de la table `livre` dont le titre contient la chaîne de caractères `Astérix`.

Remarque :

- ▷ Le symbole % est un joker qui peut symboliser n'importe quelle chaîne de caractères.

3.1.6 Plusieurs colonnes

Exemple 4

```
SELECT titre, isbn FROM livre WHERE annee >= 1990;
```

Traduction :
On veut les titres et les ISBN de la table `livre` qui sont parus après 1990.

Remarque :

- ▷ Le symbole % est un joker qui peut symboliser n'importe quelle chaîne de caractères.

3.1.7 Toutes les colonnes : *

Exemple 5

```
SELECT * FROM livre WHERE annee >= 1990;
```

Traduction :
On veut les titres et les ISBN de la table `livre` qui sont parus après 1990.

Remarque :

- ▷ L'astérisque * est un joker (*wildcard* en anglais).

3.1.8 Renommer les colonnes : AS

Exemple 6

```
SELECT titre AS titre_du_livre FROM livre WHERE annee >= 1990;
```

Traduction :
Lors de l'affichage du résulats et dans la suite de la requête (important), la colonne `titre` est renommée `titre_du_livre`.

Remarque :

- ▷ L'alias `AS` sera souvent utilisé pour raccourcir un nom, notamment lors des jointures de plusieurs tables.

4 Opérations sur les données

4.1 Sélection avec agrégation

Les requêtes effectuées jusqu'ici ont juste sélectionné des données grâce à différents filtres : aucune action à partir de ces données n'a été effectuée.

Nous allons maintenant effectuer des **opérations** à partir des données sélectionnées.

4.1.1 Compter : COUNT

Exemple 7

```
SELECT COUNT(*) AS total FROM livre
WHERE titre LIKE "%Asterix%";
```

Traduction :
On veut compter le nombre d'enregistrements de la tables `livre` comportant le mot "Astérix". Le résultat sera le seul élément d'une colonne nommée `total`.

4.1.2 Additionner : SUM

Exemple 8

```
SELECT SUM(annee) AS somme FROM livre
WHERE titre LIKE "%Asterix%";
```

Traduction :
On veut additionner les années des livres de la tables `livre` comportant le mot `Astérix`. Le résultat sera le seul élément d'une colonne nommée `somme`.

Remarque :

- ▷ Attention : dans notre cas précis, ce calcul n'a aucun sens car on additionne des années...

4.1.3 Faire une moyenne : AVG

Exemple 9

```
SELECT AVG(annee) AS moyenne FROM livre
WHERE titre LIKE "%Asterix%";
```

Traduction :
On veut calculer la moyenne des années de parution des livres de la table `livre` comportant le mot `Astérix`. Le résultat sera le seul élément d'une colonne nommée `moyenne`.

Remarque :

- ▷ Attention : là encore, ce calcul n'a aucun sens...

4.1.4 Trouver les extremums : MIN, MAX

Exemple 10

```
SELECT MIN(annee) AS minimum FROM livre
WHERE titre LIKE "%Asterix%";
```

Traduction :
On veut trouver la plus petite valeur de la colonne `annee` parmi les livres de la table `livre` comportant le mot `Astérix`. Le résultat sera le seul élément d'une colonne nommée `minimum`. Le fonctionnement est identique avec `MAX` pour la recherche du maximum.

4.1.5 Classer des valeurs : ORDER BY, ASC, DESC, LIMIT

Exemple 11

```
SELECT titre, annee FROM livre
WHERE titre LIKE "%Asterix%"
ORDER BY annee DESC;
```

Traduction :
On veut afficher tous les albums d'Astérix, et leur année de parution, classés par année décroissante.

Remarques :

- **Comportement par défaut** : Si le paramètre `ASC` ou `DESC` est omis, le classement se fait par ordre croissant (donc `ASC` est le paramètre par défaut).
- **Utilisation de `LIMIT`** : Le mot-clé `LIMIT` (suivi d'un nombre) permet de limiter le nombre de résultats affichés. Ainsi la requête

4.1.6 Suppression des doublons : DISTINCT

Exemple 12

```
SELECT DISTINCT editeur FROM livre;
```

Traduction :
On veut la liste de tous les éditeurs. Sans le mot-clé `DISTINCT`, beaucoup de doublons apparaîtraient.

5 Les jointures

On peut aussi effectuer des **recherches croisées** sur les tables. C'est ce que l'on appelle les **jointures**.

En SQL, une jointure est une opération qui permet de combiner des données de deux tables (ou plus) en fonction d'une ou plusieurs colonnes communes. Cela permet de relier des informations provenant de différentes tables pour effectuer des requêtes complexes et obtenir des résultats plus riches en informations. Les jointures sont couramment utilisées pour récupérer des données à partir de plusieurs sources de données liées entre elles, en utilisant des critères de correspondance spécifiques.

6 Exercices