Séquence : **Base de données relationnelle**

# Les rêquetes SQL

## Introduction

Nous savons créer et remplir des tables avec des ordres SQL ; cet ensemble de tables forme une base de données, contenant des données cohérentes vis à vis de nos contraintes d'intégrité.

Nous allons maintenant aborder deux autres aspects importants de l'interaction avec les SGBD :

- la sélection de données,
- la mise à jour des données.

## Base de données de travail `mediatheque.sql`

Une base de données a été associée à ce notebook ; elle est générée à partir d'un fichier texte `mediatheque.sql` contenant tous les ordres `SQL` pour

- créer toutes les tables `livre`, `usager`, `emprunt`, `auteur` et `auteur_de` se conformant aux schémas définis dans la modélisation précédemment fournie :

  $Livre(titre\ \texttt{String}, editeur\ \texttt{String}, annee\ \texttt{Int}, \underline{isbn\ \texttt{String}})$

  $Usager(nom\ \texttt{String}, prénom\ \texttt{String}, adresse\ \texttt{String}, cp\ \texttt{String}, ville$

  $Emprunt(\#code\_barre\ \texttt{String}, \underline{\#isbn\ \texttt{String}}, retour\ \texttt{Date})$

  $Auteur\ (\underline{a\_id\ \texttt{'Int'}}, nom\ \texttt{String}, prenom\ \texttt{String})$

  $Auteur\_de\ (\underline{\#a\_id\ \texttt{'Int'}, \#isbn\ \texttt{String}})$

- ajouter des enregistrements à chacune de ces tables.

# 1. Sélection de données

## 1.1 Requêtes sur une table

**La clause `SELECT`**

On considère la table $livre$ crée avec l'ordre suivant :

```
CREATE TABLE livre (titre VARCHAR(300),
                    editeur VARCHAR(90),
                    annee INT,
                    isbn CHAR(14) PRIMARY KEY);
```

Pour extraire des informations de cette table, on effectue une **requête d'interrogation** à l'aide du mot clé `SELECT` selon la syntaxe suivante :

```
SELECT attrib0, attrib1, ... FROM nom_table;
```

On obtient alors une nouvelle table formée avec les attributs `attrib0`, `attrib1`, ... .

Pour obtenir tous les attributs d'une table, on peut utiliser le caractère spécial `*` (appelé wildcard ou joker) après `SELECT` au lieu de les nommer.

**Travail à faire**

1. Formule une requête pour obtenir uniquement les titres et les années de la table.
2. Formule une requête pour obtenir toutes les données de la table.

**La clause `WHERE`**

Pour selectionner uniquement certain enregistrements de la table, on peut utiliser la clause `WHERE` selon la syntaxe :

`SELECT` selon la syntaxe suivante :

```
SELECT attrib0, attrib1, ... FROM nom_table WHERE ex
p_bool ;
```

L'expression suivant `WHERE` doit être une expression booléenne que l'on peut former à partir :

- des opérateurs de comparaison `<`, `>`, `>=`, `<=`, `=` et `<>` ;
- des opérateurs logiques `NOT`, `AND` et `OR`,
- des opérateurs spéciaux comme `LIKE`, opérateur de comparaison de texte

**Travail à faire**

Formule une requête pour obtenir tous les titres de l'éditeur Dargaud publiés entre 1970 et 1980.

Entrée[ ]:
```
1 SELECT titre FROM livre WHERE editeur = 'Dargaud' AND annee >=
```

**Opérateur spécial `LIKE`**

Pour obtenir tous les livres dont le titre contient 'Astérix' on pourrait utiliser l'opérateur `LIKE` comme il suit :

```
SELECT titre FROM livre WHERE titre LIKE motif;
```

Cet opérateur recherche des chaînes ou des sous-chaînes de caractères spécifiques et renvoie tous les enregistrements qui correspondent au motif donné.

Un motif se forme à partir d'une chaîne de caractère quelconque qui peut contenir des caractères spéciaux comme :

- le caractère `%` permet de représenter zéro, un ou plusieurs caractères quelconques ;
- Le caractère `_` permet de représenter exactement un caractère.

**Travail à faire**

1. Formule une requête por obtenir tous les titres commençant par le mot « Astérix».
2. Formule une requête pour obtenir tous les titres contenant le mot « Astérix ».
3. Formule une requête pour obtenir tous les titres dont le quatorzième caractère est un « les ».

Entrée[ ]:
```
1 SELECT titre FROM livre WHERE titre LIKE 'Astérix%';
```

Entrée[ ]:
```
1 SELECT titre FROM livre WHERE titre LIKE '%Astérix%';
```

Entrée[ ]:
```
1 SELECT titre FROM livre WHERE titre LIKE '_____________l%';
```

**Les fonctions d'agrégation**

La clause `SELECT` permet aussi d'appeler des *fonctions d'agrégation*. Ces fonctions permettent d'appliquer une fonction à l'ensemble des valeurs d'une colonne et de renvoyer le résultat comme une table ayant une seule case.

- `COUNT` : donne le nombre de résultats ;
- `AVG` : calcule la moyenne d'une colonne (ne s'applique que sur des types « numériques »);
- `SUM` : effectue la somme d'une colonne (ne s'applique que sur des types « numériques »);
- `MIN` et `MAX` : trouve respectivement le minimum et le maximum d'une colonne.

Pour savoir combien de livres contiennent la chaîne « Astérix », on peut formuler la requête suivante :

```
SELECT COUNT(titre) AS total FROM livre WHERE titre LIKE
'%Astérix%'
```

On utilise ici `AS` pour renommer la colonne « titre » en « total » dans la table produite.

Entrée[ ]:
```
1 SELECT COUNT(titre) AS total FROM livre WHERE titre LIKE '%Asté
```

**Travail à faire**

1. Formule une requête pour obtenir la somme et la moyenne des années.
2. Formule une requête l'année la plus récente.

Entrée[ ]:
```
1 SELECT SUM(annee) AS somme FROM livre ;
```

Entrée[ ]:
```
1 SELECT AVG(annee) AS somme FROM livre ;
```

## 1.2 Tri et suppression des doublons

**La clause `ORDER BY`**

Les résultats de requêtes sont affichés par le SGBD dans un ordre a priori quelconque. Pour ordonner les résultats, on peut utiliser la clause `ORDER BY` en fin de requête suivi par :

- `ASC` (ascending) pour un tri croissant ;
- `DESC` (descending) pour un tri décroissant.

**Travail à faire**

Formule une requête pour afficher tous les livres publiés après 1970 par ordre croissant.

Entrée[ ]:
```
SELECT * FROM livre WHERE annee >= 1970 ORDER BY annee ASC;
```

**Mot clé `DISTINCT`**

Supposons maintenant que l'on souhaite connaître toutes les années de publication. La requête

```
SELECT annee FROM livre;
```

nous fournit bien le résultat attendu, néanmoins la même année peut apparaître plusieurs fois.

Pour retirer les doublons d'un résultat, le mot clé `DISTINCT` peut être rajouter à la clause `SELECT`

**Travail à faire**

Teste puis commente l'effet de `DISTINCT` sur la requête précédente.

Entrée[ ]:
```
SELECT DISTINCT annee FROM livre WHERE annee >= 1970 ORDER BY a
```

### 1.3 Jointure

**La clause `JOIN`**

Étant données deux tables $A$ et $B$, la jointure consiste à créer toutes combinaisons de lignes de $A$ et de $B$ ayant un attribut de même valeur.

On obtient une jointure en utilisant la clause `JOIN`, illustré dans la syntaxe suiante :

```
SELECT * FROM table_0 JOIN table_1 ON table_0.attrib = t
able_1.attrib;
```

L'expression booléenne `table_0.attrib = table_1.attrib` représente la **condition de jointure** permettant de selectionner les lignes des deux tables qui doivent être fusionnées.

**Travail à faire**

1. Formule une requête pour obtenir le titre de tous les livres ayant été empruntés.
2. Modifie la requête précedente en y ajoutant une clause `WHERE` pour obtenir le titre des livres empruntés ayant publiés après l'an 2000.
3. En formulant plusieurs jointures dans la même requête, affiche la table contenant le nom et le prénom des usagers ayant un emprunt en cours, ainsi que le titre des livres correspondants.

Entrée[ ]:
```
SELECT livre.titre FROM emprunt JOIN livre ON emprunt.isbn = li
```

Entrée[ ]:
```
SELECT livre.titre, livre.annee FROM emprunt JOIN livre ON empr
```

## 2. Modification des données

Les données stockées dans un SGBD en sont a priori pas figées et peuvent être modifiées au cours du temps ; la médiathèque doit pouvoir enregistrer de nouveaux livres, supprimer ceux qui auraient été perdus et mettre à jour les emprunts de ses usagers.

### 2.1 Suppression de lignes

**Ordre de suppression**

L'ordre

```
DELETE FROM t WHERE c
```

permet de supprimer de la table `t` toutes les lignes vérifiant la condition `c`.

**Travail à faire**

L'usager Sébastien Petit, dont le `code_barre` est `934701281931582` , a rendu tous ses livres.

1. Écris l'ordre permettant de supprimer de la table `emprunt` les enregistrements correspondants.
2. Vérifie que la table `emprunt` ne possède plus d'enregistrements pour cet usager avec une requête de selection qui utilise la fonction `COUNT` .

Entrée[ ]:
```
1 DELETE FROM emprunt WHERE code_barre = '934701281931582'
```

Entrée[ ]:
```
1 SELECT COUNT() AS total FROM emprunt WHERE code_barre = '934701
```

**Point de vigilence**

Un ordre `DELETE` sans clause `WHERE` efface toutes les lignes de la table! Il ne faut pas confondre les opérations :

- `DROP` qui détruit la table (et ses données) de la base de données, elle ne peut donc plus y être référencée ;
- `DELETE` qui peut vider une table de son contenu mais sans la supprimer de la base de données. Il est donc possible d'y insérer à nouveau des enregistrements.

**Travail à faire**

Essaie de supprimer un enregistrement à l'aide de l'ordre suivant puis observe le résultat.

```
DELETE FROM livre WHERE isbn = '978-0201914658'
```

Comment peux-tu expliquer le résultat obtenu?

Entrée[ ]:
```
1 DELETE FROM livre WHERE isbn = '978-0201914658'
```

Lorsque l'on met à jour une table, les contraintes sont à nouveau vérifiées. Pour supprimer un enregistrement, il faut s'assurer que les valeurs des attributs déclarés comme clé primaire ne sont plus référencés comme clé étrangère dans une autre table et au besoin les supprimer d'abord.

**Travail à faire**

Supprime les enregistrements référençant l'isbn `978-0201914658` comme clé étrangère, puis essaie à nouveau de supprimer le livre correspondant.

Entrée[ ]:
```
1 DELETE FROM auteur_de WHERE isbn = '978-0201914658'
```

## 2.2 Mise à jour

Imaginons maintenant qu'un usager ait perdu sa carte de médiathèque. Une nouvelle carte lui étant délivrée, il faut alors mettre à jour la base de données en remplaçant la valeur de l'ancien code barre par la nouvelle.

**Ordre `UPDATE`**

On peut remplacer certaines valeurs d'attribut d'un ensemble de lignes avec l'ordre suivant :

```
UPDATE t SET a1 = e1, ..., SET an = en WHERE c
```

Toutes les lignes de la table `t` vérifiant la conditions `c` sont sélectionnées et pour chacune, les valeurs des attributs `a1, ..., an` sont mises à jour avec les valeurs `e1, ..., en` .

**Travail à faire**

1. Essaie de mettre à jour la valeur de `code_barre` de la table `usager` pour Sébastien Petit (qui a perdu sa carte) avec la nouvelle valeur `'934701281931581'` . Qu'observes-tu?
2. Supprime les lignes correspondant à cette référence de la table `emprunt` puis essaie à nouveau.

Entrée[ ]:
```
1 UPDATE usager SET code_barre = '934701281931581' WHERE code_bar
```

Entrée[ ]:
```
1 SELECT * FROM emprunt WHERE code_barre = '934701281931582'
```

Entrée[ ]:
```
1 DELETE FROM emprunt WHERE code_barre = '934701281931582'
```

Entrée[ ]:
```
1 SELECT * FROM usager WHERE code_barre = '934701281931582'
```