Chapitre 7 - Bases de données

Séance 2 - Le langage SQL

Du modèle relationnel au SGBD

Les considérations sur le modèle relationnel du cours précédent traitaient plutôt de la structure mathématique des données.

Il s'agissait de déterminer la meilleure structure pour représenter les données et les relations qui les lient.

Il convient maintenant d'aborder la partie logicielle : les **SGBD** (Systèmes de Gestion de Bases de Données).

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 modifier la base de données. Le SGBD est garant de l'intégrité de cette base, et prévient notamment que les modifications (souvent appelées **transactions**) ne soient pas préjudiciables à la base de données.

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

Les SGBD les plus utilisés sont basés sur le modèle relationnel. Parmi eux, citons Oracle, MySQL, Microsoft SQL Server, PostgreSQL, Microsoft Access, SQLite, MariaDB...

Mais de plus en plus de SGBD **non-relationnels** sont utilisés, spécialement adaptés à des données plus diverses et moins structurées. On les retrouve sous l'appelation **NoSQL** (pour *Not only SQL*). Citons parmi eux MongoDB, Cassandra (Facebook), BigTable (Google)…

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

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

Introduction au langage SQL

Dans toute la suite, nous allons travailler avec la base de données **livres.db** qui provient de l'ouvrage paru chez Ellipses, cité en bibliographie

Différents moyens d'interroger la base de données

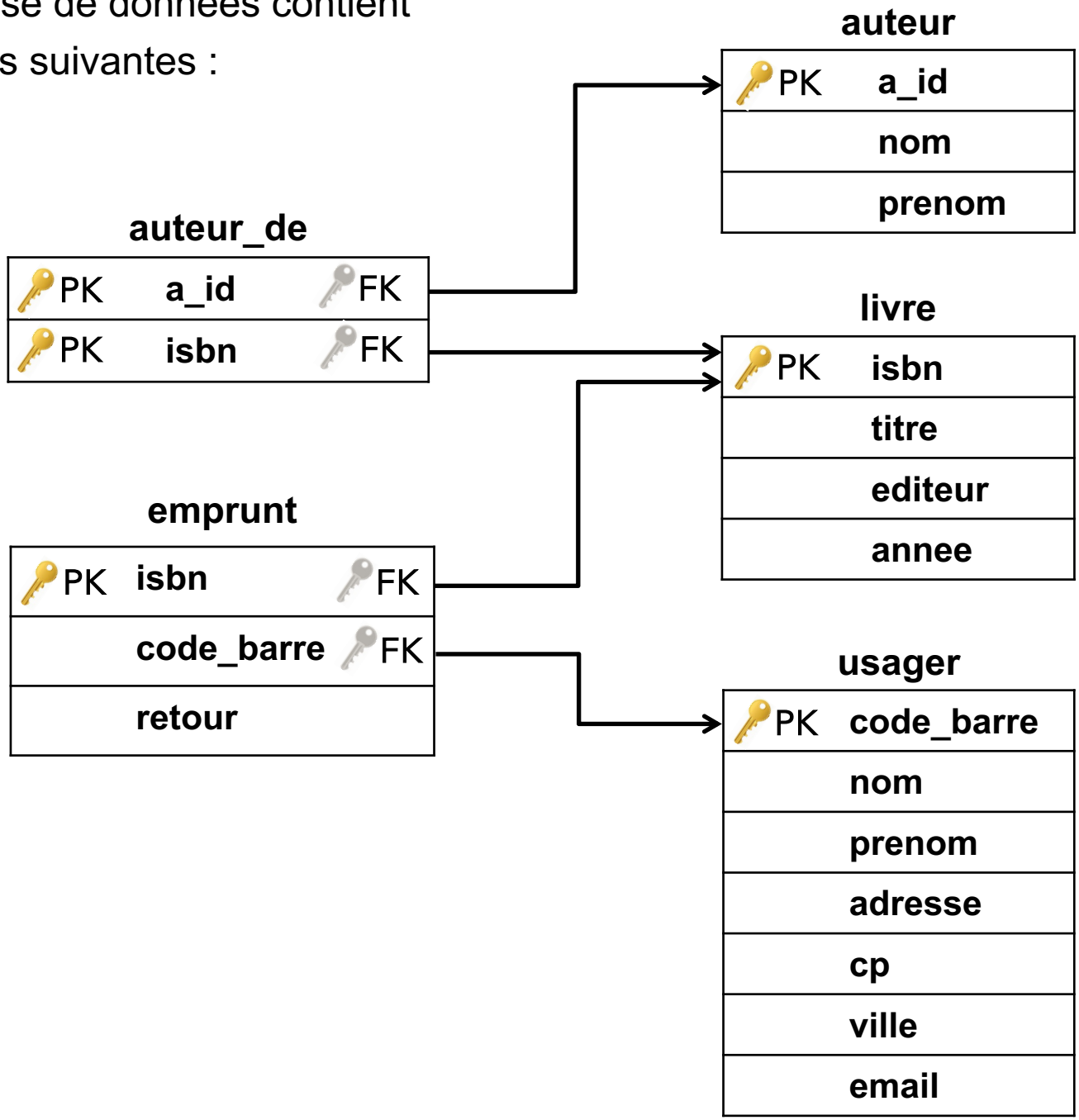
- **En ligne avec sqliteonline.com**
  - Rendez vous sur **https://sqliteonline.com/**
  - Par File / OpenDB, ouvrez le fichier livres.db précédemment téléchargé
  - Écrivez votre requête plus cliquez sur Run.

- **Avec un logiciel externe : DB Browser for SQLite**
  - Téléchargez la base de données **livres.db** sur Pronote
  - Lancez **DB Browser for SQLite**
  - Cliquez sur « Ouvrir une Base de Données » et ouvrez le fichier **livres.db**


DB Browser for SQLite

Dans toute la suite, les manipulations sont à faire en interrogeant la base de données **livres.db** avec les méthodes indiquées.

Cette base de données contient les tables suivantes :



1. Sélection de données *(exemples 1 à 9)*

Exemple 1

Requête basique : SELECT, FROM

La projection **SELECT** consiste à créer une nouvelle relation en conservant certains attributs (en partie ou en totalité). Autrement dit, la projection opère une sélection sur les colonnes d'une table. Elle se réalise avec la commande **SELECT**, qui est de loin la commande la plus utilisée en SQL.

- **Commande :**

```
SELECT * FROM livre ;
```

- **Traduction :**

  On veut obtenir le contenu entier de la table livre

  * est un joker (*wildcard* en anglais), un raccourci qui signifie « tous les attributs »

- **Résultat :**

	titre	editeur	annee	isbn
1	Les Aventures de Huckleberry Finn	Flammarion	2020	978-2081509511
2	Fondation et Empire	Editions Denoël	1999	978-2207249123
3	Akira	Glénat	2000	978-2723428262
4	Les Robots	Editions Milan	2017	978-2745989857
5	Astérix chez les Pictes	Editions Albert René	2013	978-2864972662

L'exécution s'est terminée sans erreur.
Résultat : 128 enregistrements ramenés en 80ms
À la ligne 1 :
SELECT * FROM livre ;

Remarques

- Les mots-clés SQL sont traditionnellement écrits en MAJUSCULES.
- Le ; signale la fin de l'instruction.
- L'indentation n'est pas syntaxique (pas comme en Python). On peut faire des retours à la ligne et des indentations pour rendre le code plus lisible.

Exemple 2

Requête basique : SELECT, FROM

- **Commande :**

```
SELECT titre, annee FROM livre ;
```

- **Traduction :**

  On veut les titres et les années de la table livre

- **Résultat :**

	titre	annee
1	Les Aventures de Huckleberry Finn	2020
2	Fondation et Empire	1999
3	Akira	2000
4	Les Robots	2017
5	Astérix chez les Pictes	2013

L'exécution s'est terminée sans erreur.
Résultat : 128 enregistrements ramenés en 17ms
À la ligne 1 :
SELECT titre, annee FROM livre ;

Exemple 3

Requête basique : SELECT, FROM, WHERE

La sélection **WHERE**, aussi appelée restriction, crée une nouvelle relation en sélectionnant les tuples sur la base de conditions à définir et qui portent sur les attributs. Autrement dit, la sélection opère une sélection sur les lignes selon un prédicat qui doit être précisé.

- **Commande :**

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

- **Traduction :**

  On veut les titres de la table «livre» qui sont parus après (ou en) 1990 ;

- **Résultat :**

	titre
1	Les Aventures de Huckleberry Finn
2	Fondation et Empire
3	Akira
4	Les Robots
5	Astérix chez les Pictes
6	Les Monades urbaines

```
L'exécution s'est terminée sans erreur.
Résultat : 112 enregistrements ramenés en 20ms
À la ligne 1 :
SELECT titre FROM livre WHERE annee >= 1990;
```

Remarque :

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

Exemple 4

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

- **Commande :**

```
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.

- **Résultat :**

	titre
1	Croisière sans escale
2	Le Domaine des dieux
3	L'Éducation sentimentale

```
L'exécution s'est terminée sans erreur.
Résultat : 3 enregistrements ramenés en 4ms
À la ligne 1 :
SELECT titre FROM livre WHERE annee IN (1990, 1991, 1992);
```

Exemple 5

Requête avec booléens : AND - OR

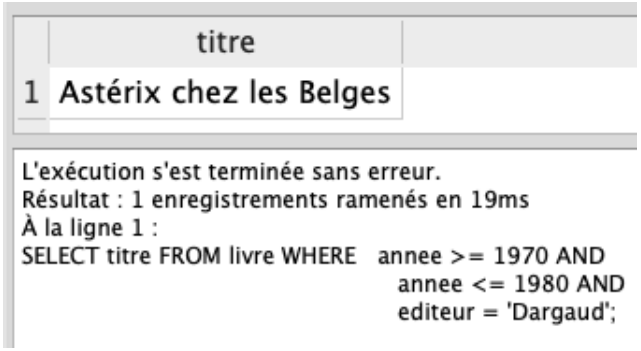
- **Commande :**

```
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;

- **Résultat :**



	titre
1	Astérix chez les Belges

```
L'exécution s'est terminée sans erreur.
Résultat : 1 enregistrements ramenés en 19ms
À la ligne 1 :
SELECT titre FROM livre WHERE   annee >= 1970 AND
                                annee <= 1980 AND
                                editeur = 'Dargaud';
```

Exemple 6

Requête approchée : LIKE

- **Commande :**

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

- **Traduction :**

On veut les titres de la table «livre» dont le titre contient la chaîne de caractères "Astérix".

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

- **Résultat :**

	titre
1	Astérix chez les Pictes
2	Astérix et Cléopâtre
3	Le Tour de Gaule d'Astérix
4	Astérix et les Normands

L'exécution s'est terminée sans erreur.
Résultat : 10 enregistrements ramenés en 19ms
À la ligne 1 :
SELECT titre FROM livre WHERE titre LIKE '%Astérix%';

Exemple 7

Plusieurs colonnes

- **Commande :**

```
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.

- **Résultat :**

	titre	isbn
1	Les Aventures de Huckleberry Finn	978-2081509511
2	Fondation et Empire	978-2207249123
3	Akira	978-2723428262
4	Les Robots	978-2745989857

L'exécution s'est terminée sans erreur.
Résultat : 112 enregistrements ramenés en 20ms
À la ligne 1 :
SELECT titre, isbn FROM livre WHERE annee >= 1990;

Exemple 8

Toutes les colonnes

- **Commande :**

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

- **Traduction :**

On veut toutes les colonnes disponibles de la table «livre» pour les livres qui sont parus après 1990.

- **Résultat :**

	titre	editeur	annee	isbn
1	Les Aventures de Huckleberry Finn	Flammarion	2020	978-2081509511
2	Fondation et Empire	Editions Denoël	1999	978-2207249123
3	Akira	Glénat	2000	978-2723428262
4	Les Robots	Editions Milan	2017	978-2745989857
5	Astérix chez les Pictes	Editions Albert René	2013	978-2864972662

L'exécution s'est terminée sans erreur.
Résultat : 112 enregistrements ramenés en 20ms
À la ligne 1 :
SELECT * FROM livre WHERE annee >= 1990;

Exemple 9

Renommer les colonnes : AS

- **Commande :**

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

- **Traduction :**

Lors de l'affichage du résulat et dans la suite de la requête (important), la colonne "titre" est renommée "titre_du_livre ".

- **Résultat :**

	titre_du_livre
1	Les Aventures de Huckleberry Finn
2	Fondation et Empire
3	Akira
4	Les Robots
5	Astérix chez les Pictes

L'exécution s'est terminée sans erreur.
Résultat : 112 enregistrements ramenés en 19ms
À la ligne 1 :
SELECT titre AS titre_du_livre FROM livre WHERE annee >= 1990;

2. Opérations sur les données : sélection avec agrégation *(exemples 10 à 15)*

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. On appelle ces opérations des **opérations d'agrégation**.

Exemple 10

Compter : COUNT

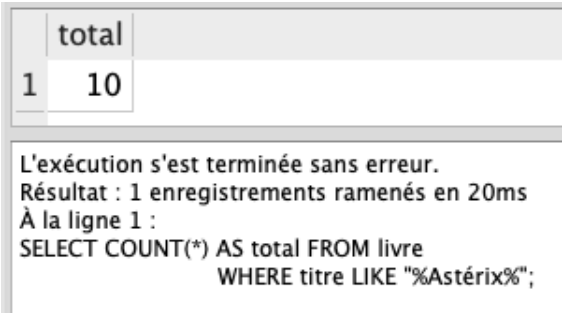
- **Commande :**

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

- **Traduction :**

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

- **Résultat :**



Exemple 11

Additionner : SUM

- **Commande :**

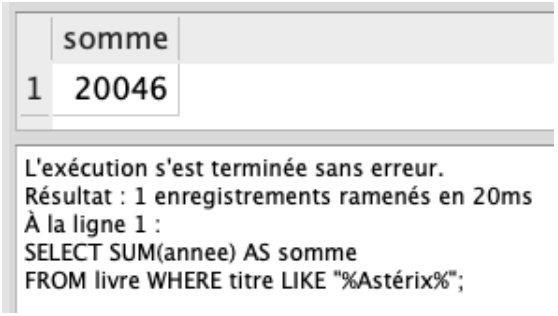
```
SELECT SUM(annee) AS somme
FROM livre WHERE titre LIKE "%Astérix%";
```

- **Traduction :**

On veut additionner les années des livres de la tables livres comportant le mot "Astérix". Le résultat sera le seul élément d'une colonne nommée «somme».

Attention : dans notre cas précis, ce calcul n'a aucun sens....

- **Résultat :**



	somme
1	20046

L'exécution s'est terminée sans erreur.
Résultat : 1 enregistrements ramenés en 20ms
À la ligne 1 :
SELECT SUM(annee) AS somme
FROM livre WHERE titre LIKE "%Astérix%";

Exemple 12

Faire une moyenne : AVG

- **Commande :**

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

- **Traduction :**

On veut calculer la moyenne des années de parution des livres de la table livres comportant le mot "Astérix". Le résultat sera le seul élément d'une colonne nommée «moyenne».

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

- **Résultat :**

	moyenne
1	2004.6

L'exécution s'est terminée sans erreur.
Résultat : 1 enregistrements ramenés en 19ms
À la ligne 1 :
SELECT AVG(annee) AS moyenne
FROM livre
WHERE titre LIKE "%Astérix%";

Exemple 13

Trouver les extremums : MIN, MAX

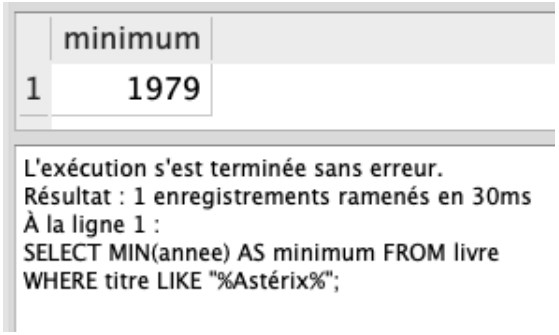
- **Commande :**

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

- **Traduction :**

On veut trouver la plus petite valeur de la colonne annee parmi les livres de la tables 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.

- **Résultat :**



	minimum
1	1979

L'exécution s'est terminée sans erreur.
Résultat : 1 enregistrements ramenés en 30ms
À la ligne 1 :
SELECT MIN(annee) AS minimum FROM livre
WHERE titre LIKE "%Astérix%";

Exemple 14

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

- **Commande :**

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

- **Traduction :**

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

- **Résultat :**

	titre	annee
1	Astérix et la Transitalique	2017
2	Astérix chez les Pictes	2013
3	Astérix légionnaire	2011
4	L'Odyssée d'Astérix	2008

```
L'exécution s'est terminée sans erreur.
Résultat : 10 enregistrements ramenés en 18ms
À la ligne 1 :
SELECT titre, annee FROM livre
                    WHERE titre LIKE "%Astérix%"
                    ORDER BY annee DESC;
```

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 suivante permet d'obtenir les renseignements sur l'Astérix le plus récent.

```
SELECT titre, annee FROM livre
                WHERE titre LIKE "%Astérix%"
                ORDER BY annee DESC
                LIMIT 1;
```

Exemple 15

Suppression des doublons : DISTINCT

- **Commande :** `SELECT DISTINCT editeur FROM livre;`
- **Traduction :**

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

- **Résultat :**

	editeur
1	Flammarion
2	Editions Denoël
3	Glénat
4	Editions Milan

L'exécution s'est terminée sans erreur.
Résultat : 70 enregistrements ramenés en 20ms
À la ligne 1 :
SELECT DISTINCT editeur FROM livre;

3. Des recherches croisées sur les tables : les jointures *(exemples 16 à 18)*

Observons le contenu de la table «emprunt» :

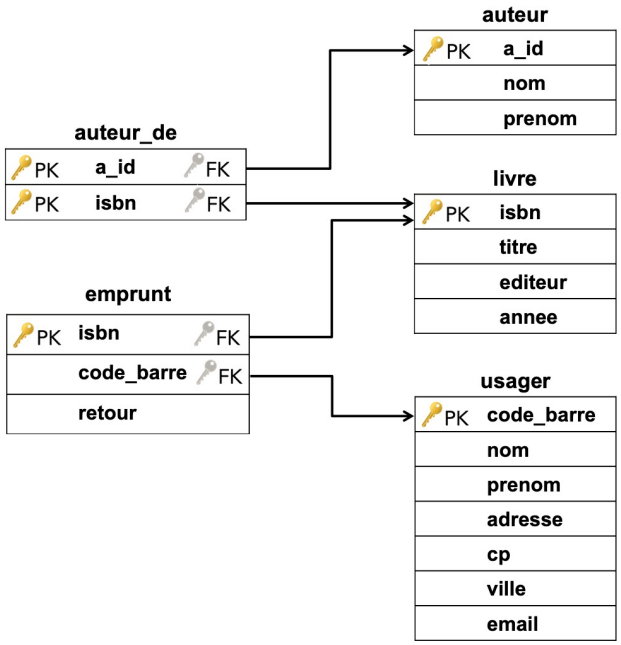
```
SELECT * FROM emprunt;
```

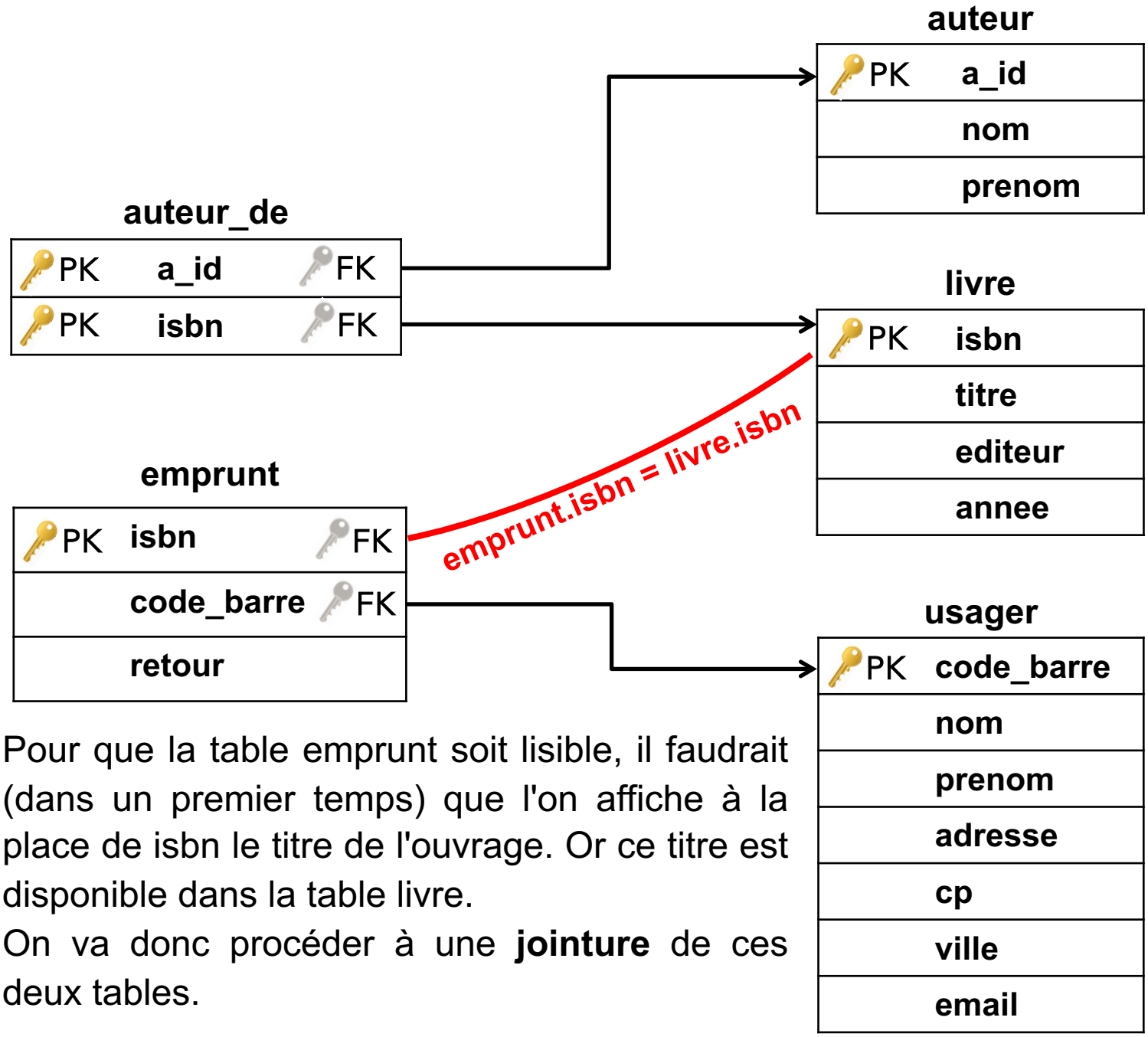
	code_barre	isbn	retour
1	421921003090881	978-2081358881	2020-04-28
2	421921003090881	978-2207249123	2020-04-28
3	421921003090881	978-2824709420	2020-04-28
4	137332830764072	978-2352879183	2020-02-20
5	137332830764072	978-2335008586	2020-02-20

L'exécution s'est terminée sans erreur.
Résultat : 17 enregistrements ramenés en 44ms
À la ligne 1 :
SELECT * FROM emprunt;

Le contenu est peu lisible : qui a emprunté quel livre ?

Souvenons-nous du diagramme de la base de données.





Pour que la table emprunt soit lisible, il faudrait (dans un premier temps) que l'on affiche à la place de isbn le titre de l'ouvrage. Or ce titre est disponible dans la table livre.

On va donc procéder à une **jointure** de ces deux tables.

Exemple 16

Jointure de 2 tables : JOIN

- **Commande :**

```
SELECT * FROM emprunt
JOIN livre ON emprunt.isbn = livre.isbn;
```

- **Traduction :**

Comme plusieurs tables sont appelées, nous préfixons chaque colonne avec le nom de la table. Nous demandons ici l'affichage d'une nouvelle table en liant les lignes de deux tables qui respectent la condition emprunt.isbn = livre.isbn.

- **Résultat :**

emprunt.isbn livre.isbn

	code_barre	isbn	retour	titre	editeur	annee	isbn
1	421921003090881	978-2081358881	2020-04-28	Mrs Dalloway	Flammarion	2015	978-2081358881
2	421921003090881	978-2207249123	2020-04-28	Fondation et Empire	Editions Denoël	1999	978-2207249123
3	421921003090881	978-2824709420	2020-04-28	Le Journal d'un fou	Bibebook	2013	978-2824709420
4	137332830764072	978-2352879183	2020-02-20	Guerre et Paix	Archipoche	2016	978-2352879183
5	137332830764072	978-2335008586	2020-02-20	Les Voyages de Gulliver	Primento	2015	978-2335008586

L'exécution s'est terminée sans erreur.
Résultat : 17 enregistrements ramenés en 23ms
À la ligne 1 :
SELECT * FROM emprunt
JOIN livre ON emprunt.isbn = livre.isbn;

L'expression JOIN livre ON emprunt.isbn = livre.isbn doit se comprendre comme ceci : on «invite» la table livre, et la correspondance entre la table livre et la table emprunt doit se faire sur l'attribut isbn, qui est la clé primaire de livre et une clé étrangère d'emprunt. Il est donc très important de spécifier ce sur quoi les deux tables vont se retrouver (ici, isbn).

Exemple 17

Jointure de 2 tables : JOIN

- **Commande :**

```
SELECT livre.titre, emprunt.code_barre, emprunt.retour
FROM emprunt
JOIN livre ON emprunt.isbn = livre.isbn;
```

- **Traduction :**

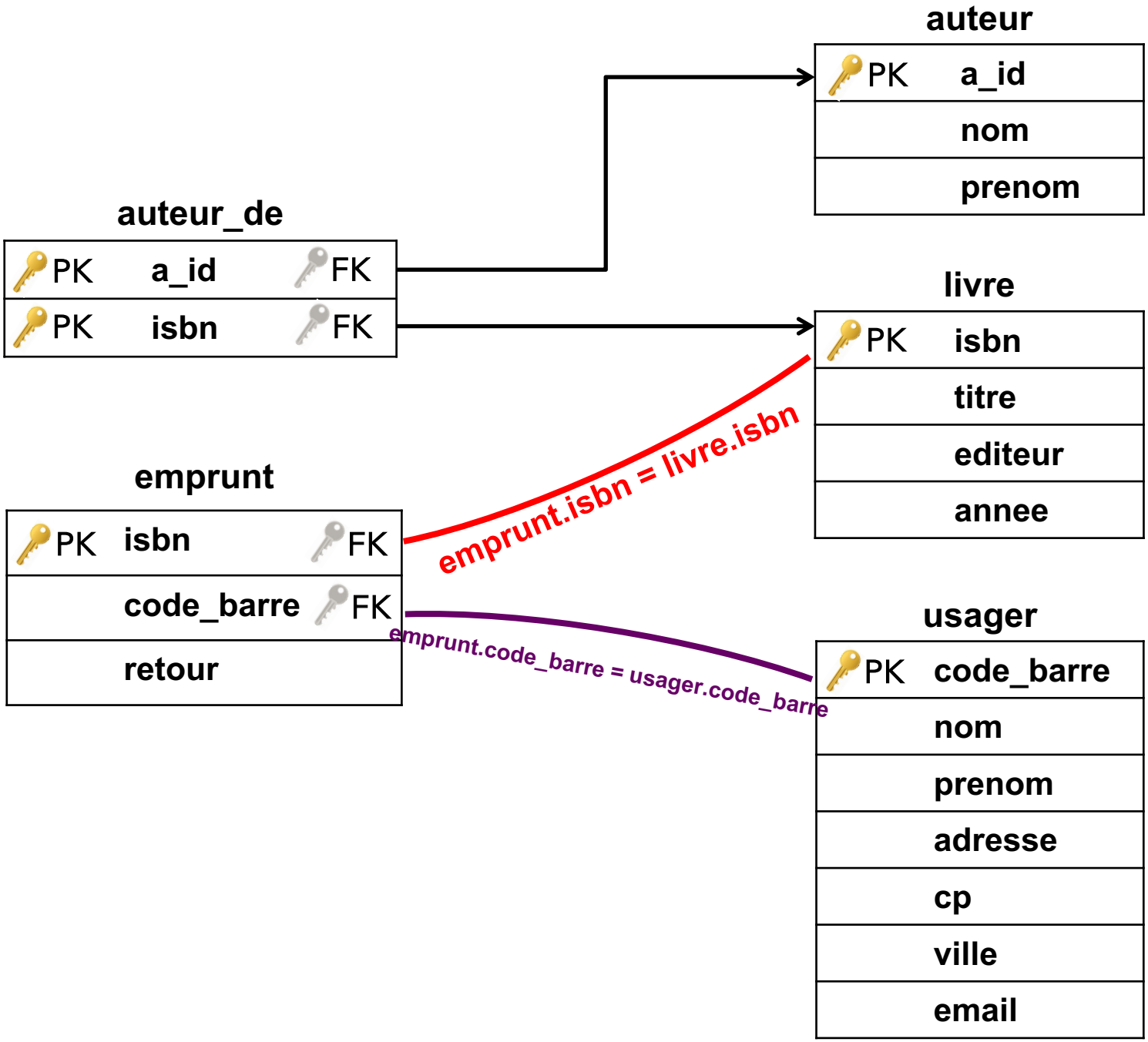
Nous demandons ici l'affichage de la table en conservant certains attributs.

- **Résultat :**

	titre	code_barre	retour
1	Mrs Dalloway	421921003090881	2020-04-28
2	Fondation et Empire	421921003090881	2020-04-28
3	Le Journal d'un fou	421921003090881	2020-04-28
4	Guerre et Paix	137332830764072	2020-02-20
5	Les Voyages de Gulliver	137332830764072	2020-02-20

L'exécution s'est terminée sans erreur.
Résultat : 17 enregistrements ramenés en 22ms
À la ligne 1 :
SELECT livre.titre, emprunt.code_barre, emprunt.retour FROM emprunt
JOIN livre ON emprunt.isbn = livre.isbn;

Ce résultat a permis d'améliorer la visibilité de la table emprunt, mais il reste la colonne code_barre qui est peu lisible. Nous pouvons la remplacer par les nom et prénom de l'emprunteur, en faisant une nouvelle jointure, en invitant maintenant les deux tables livre et usager.


auteur
PK a_id
nom
prenom
auteur_de
PK a_id FK
PK isbn FK
livre
PK isbn
titre
editeur
annee
emprunt
PK isbn FK
code_barre FK
retour
emprunt.isbn = livre.isbn
emprunt.code_barre = usager.code_barre
usager
PK code_barre
nom
prenom
adresse
cp
ville
email

Exemple 18

Jointure de 3 tables : JOIN

- **Commande :**

```
SELECT usager.nom, usager.prenom, livre.titre, emprunt.retour
FROM emprunt
JOIN livre ON emprunt.isbn = livre.isbn
JOIN usager ON emprunt.code_barre = usager.code_barre ;
```

- **Traduction :**

Il faut bien comprendre que la table principale qui nous intéresse ici est emprunt, mais qu'on modifie les valeurs affichées en allant chercher des correspondances dans deux autres tables.

- **Résultat :**

	nom	prenom	titre	retour
1	MOREAU	ALAIN	Mrs Dalloway	2020-04-28
2	MOREAU	ALAIN	Fondation et Empire	2020-04-28
3	MOREAU	ALAIN	Le Journal d'un fou	2020-04-28
4	DUBOIS	PHILIPPE	Guerre et Paix	2020-02-20
5	DUBOIS	PHILIPPE	Les Voyages de Gulliver	2020-02-20

```
L'exécution s'est terminée sans erreur.
Résultat : 17 enregistrements ramenés en 25ms
À la ligne 1 :
SELECT usager.nom, usager.prenom, livre.titre, emprunt.retour
FROM emprunt
JOIN livre ON emprunt.isbn = livre.isbn
JOIN usager ON emprunt.code_barre = usager.code_barre ;
```

Remarque :

Des alias peuvent être donnés aux tables (par AS) afin de faciliter l'écriture.

```
SELECT u.nom, u.prenom, l.titre, e.retour
FROM emprunt AS e
JOIN livre AS l ON e.isbn = l.isbn
JOIN usager AS u ON e.code_barre = u.code_barre;
```

4. Création et modification d'une base de données *(19 à 24)*

L'objectif est de créer la table suivante :

id	Nom	Maths	Anglais	NSI
1	Alice	16	11	17
2	Bob	12	15	10
3	Charles	9	11	18

Exemple 19

Création d'une table : CREATE TABLE

- **Commande :**

```
CREATE TABLE Table_notes (
        Id INTEGER PRIMARY KEY,
        Nom TEXT,
        Maths INTEGER,
        Anglais INTEGER,
        NSI INTEGER
        );
```

- **Remarques :**

  C'est l'utilisateur qui spécifie, éventuellement, quel attribut sera une clé primaire.

- **Résultat :**

  Dans DB Browser, il faut avoir au préalable créé une nouvelle base de données.

SQL 1 ✖

```sql
CREATE TABLE Table_notes (
        Id INTEGER PRIMARY KEY,
        Nom TEXT,
        Maths INTEGER,
        Anglais INTEGER,
        NSI INTEGER
        );
```

```
Requête exécutée avec succès : CREATE TABLE Table_notes (
        Id INTEGER PRIMARY KEY,
        Nom TEXT,
        Maths INTEGER,
        Anglais INTEGER,
        NSI INTEGER
        ); (en 0 ms)
```

Exemple 20

Insertion de valeurs : INSERT INTO, VALUES

- **Commande :**

```
INSERT INTO Table_notes VALUES  (1, 'Alice', 16, 11, 17),
                                (2, 'Bob', 12, 15, 10),
                                (3, 'Charles', 9, 11, 18);
```

- **Résultat :**

```
10    INSERT INTO Table_notes VALUES  (1, 'Alice', 16, 11, 17),
11                                    (2, 'Bob', 12, 15, 10),
12                                    (3, 'Charles', 9, 11, 18);
```

```
Requête exécutée avec succès : INSERT INTO Table_notes VALUES     (1, 'Alice', 16, 11,
17),

(2, 'Bob', 12, 15, 10),

(3, 'Charles', 9, 11, 18); (en 0 ms, 3 enregistrements affectés)
```

Exemple 21

Intérêt de la clé primaire

Essayons d'insérer un 4ème enregistrement ayant le même `id` qu'un autre élève.

- **Commande :**

```
INSERT INTO Table_notes VALUES  (3, 'Denis', 18, 10, 12);
```

- **Résultat :**

La contrainte de relation est violée : le SGBD «protège» la base de données en n'acceptant pas la proposition d'insertion. La base de données n'est pas modifiée.

```
UNIQUE constraint failed: Table_notes.Id: INSERT INTO Table_notes VALUES  (3, 'Denis',
18, 10, 12);
```

- **Remarque :**

Il est possible de «déléguer» la gestion des valeurs de la clé primaire avec l'instruction `AUTOINCREMENT`. La déclaration de la table et l'insertion des valeurs serait :

```
CREATE TABLE Table_notes (
        Id INTEGER PRIMARY KEY AUTOINCREMENT,
        Nom TEXT,
        Maths INTEGER,
        Anglais INTEGER,
        NSI INTEGER
        );

INSERT INTO Table_notes (Nom, Maths, Anglais, NSI) VALUES
    ('Alice', 16, 11, 17),
    ('Bob', 12, 15, 10),
    ('Charles', 9, 11, 18);
```

et le résultat serait :

Id	Nom	Maths	Anglais	NSI
Filtre	Filtre	Filtre	Filtre	Filtre
1	Alice	16	11	17
2	Bob	12	15	10
3	Charles	9	11	18

L'attribut `id` est donc géré automatiquement par le SGBD.

Exemple 22

Modification d'une valeur : UPDATE, SET

Pour modifier la note de Maths d'Alice :

- **Commande :**

```
UPDATE Table_notes SET Maths = 18 WHERE Nom = 'Alice';
```

Exemple 23

Suppression d'un enregistrement : DELETE

Pour supprimer totalement la ligne concernant Charles :

- **Commande :**

```
DELETE FROM Table_notes WHERE Nom = 'Charles';
```

Si une autre table contient par exemple l'attribut `id` comme clé étrangère, et si l'`id` de Charles fait partie de cette table, le SGBD refusera de supprimer cette ligne, afin de ne pas violer la contrainte de référence.

Exemple 24

Suppression d'une table totale : DROP TABLE

Pour supprimer totalement et définitivement la table :

- **Commande :**

```
DROP TABLE Table_notes;
```

Là encore, si une autre table est reliée à `Table_notes` par une clé étrangère, la suppression sera bloquée par le SGBD.

Vidéo Lumni

Interrogation d'une base de données relationnelle

https://www.lumni.fr/video/interrogation-d-une-base-de-donnees-relationnelle