Chapitre 1

BDD partie 3

1 Niveau physique Langage SQL

1.1 Le SGBD

Il garantit entre autres-

- *l'indépendance physique* de la BDD : l'utilisateur n'a pas à se soucier des aspects matériels ;
- *l'indépendance logique* : les programmes qui utilisent la BDD sont indépendants de sa structure logique ;
- *l'accès aux données* : il se fait grâce à un *langage de manipulation des données* (LMD) optimisé pour la rapidité et l'accès simultané multiple en lecture/écriture ;
- *la centralisation des données pour administration* ;
- *la non-redondance des données* ;
- *la sécurité des données* vis-à-vis du piratage mais aussi des pannes physiques.

1.2 Principaux SGBD en 2020

1.3 Le SQL

– *Structured Query Langage* (langage de requêtes structuré).

– Créé en 1974, normalisé en 1986, dernière version parue en 2011.

– Utilisé par la plupart des SGBD avec de petites différences.

1.4 Exemple de requête

SQL

```sql
SELECT DISTINCT nom, prenom
FROM Auteur
        JOIN Ecrire ON Ecrire.id_auteur = Auteur.id_auteur
        JOIN Livre ON Livre.num_isbn = Ecrire.num_isbn
WHERE Livre.titre LIKE '%s%';
```

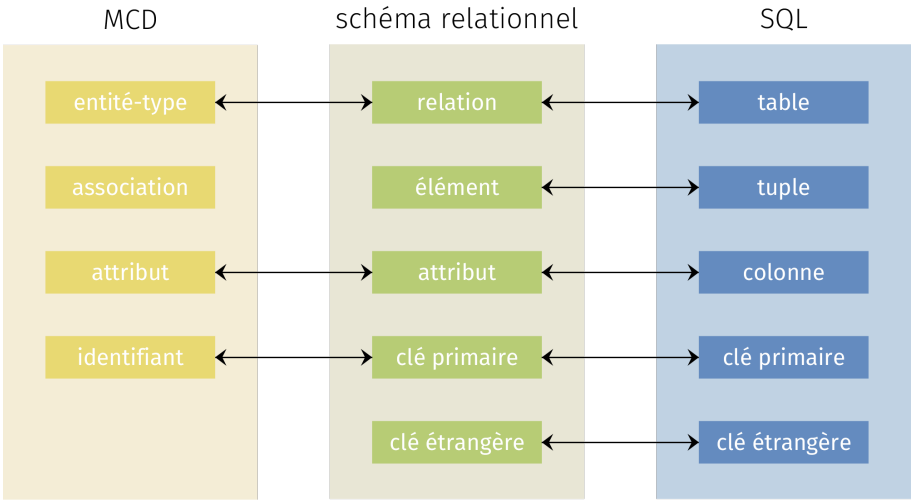
Voici comment obtenir la liste des noms et prénoms des auteurs ayant écrit un livre dont le titre comporte la lettre « s ». Nous expliquerons comment produire de telles requêtes plus tard.

1.5 Vocabulaire

En SQL, les relations s'appellent des *tables*.

Les éléments des tables s'appellent des *tuples*.

1.6 Bilan des termes utilisés



1.7 Conventions

On écrira les mots-clés SQL en majuscules.

On ne met pas d'accents ou d'espaces dans les noms des tables ou des attributs.

Les espaces et tabulations n'ont qu'un rôle esthétique.

Les requêtes peuvent prendre plusieurs lignes mais doivent se terminer par un point-virgule.

On utilisera SQLite car on peut s'en servir avec DB Browser sans installation compliquée.

2 Création de la BDD

2.1 Créer une BDD

SQL

```
CREATE DATABASE Bibliotheque;
USE Bibliotheque;
```

On n'utilisera pas (ou peu) cette commande dans les exercices.

2.2 Supprimer une BDD

SQL

```sql
DROP DATABASE Bibliotheque;
```

On n'utilisera pas cette commande non plus.

2.3 Créer une table

Voici comment créer la table **Pays** :

SQL

```sql
DROP TABLE IF EXISTS Pays; -- recréer la table de zéro
CREATE TABLE Pays
(
    nom_pays   TEXT,
    population INTEGER,
    superficie INTEGER,
    PRIMARY KEY (nom_pays), -- clé primaire
    CHECK (population > 0), -- contraintes utilisateur
    CHECK (superficie > 0)
);
```

2.4 Table Livre

SQL

```sql
DROP TABLE IF EXISTS Livre;
CREATE TABLE Livre
(
    num_isbn INTEGER,
    titre    TEXT,
    annee    TEXT,
    PRIMARY KEY (num_isbn),
    CHECK (date(annee) BETWEEN '1900' AND '2100')
);
```

`date(annee) BETWEEN '1900' AND '2100'` est l'équivalent SQL de `'1900' <= date(annee) <= '2100'` en Python.

Attention

> SQLite ne connaît pas le type `DATE`, il faut créer des attributs de type `TEXT` et utiliser la fonction `date`.

2.5 Table Livre

SQL

```sql
DROP TABLE IF EXISTS Auteur;
CREATE TABLE Auteur
(
    id_auteur      INTEGER,
    nom_pays       TEXT,
    nom            TEXT,
    prenom         TEXT,
    date_naissance TEXT,
    PRIMARY KEY (id_auteur),
    UNIQUE (nom, prenom), -- contrainte d'unicité
    FOREIGN KEY (nom_pays) REFERENCES Pays (nom_pays)
    /*nom_pays est une clé étrangère*/
        ON DELETE CASCADE
        /*si on supprime des tuples dans Pays, automatiquement
        (en cascade) on supprimera les tuples qui y font
        réference dans Auteur*/
        ON UPDATE CASCADE
        /*si on met à jour les attributs nom_pays dans Pays,
        alors le SGBD les mettra à jour aussi dans Auteur*/
```

2.6 Ordre des créations des tables

On ne peut pas créer **Auteur** avant d'avoir crée **Pays** car **Auteur** possède une clé étrangère liée à **Pays**.

2.7 Table Ecrire

SQL

```sql
DROP TABLE IF EXISTS Ecrire;
CREATE TABLE Ecrire
(
    id_auteur    INTEGER,
    num_isbn     INTEGER,
    nb_chapitres INTEGER,
    PRIMARY KEY (id_auteur, num_isbn),
    FOREIGN KEY (id_auteur) REFERENCES Auteur (id_auteur)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (num_isbn) REFERENCES Livre (num_isbn)
            ON DELETE CASCADE
            ON UPDATE CASCADE
);
```

3 Variantes syntaxiques et autres

3.1 Écriture plus compacte

On peut signifier qu'un attribut est une clé étrangère dans sa définition même :

SQL

```sql
DROP TABLE IF EXISTS Ecrire;
CREATE TABLE Ecrire
(
    id_auteur    INTEGER REFERENCES Auteur (id_auteur)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    num_isbn     INTEGER REFERENCES Livre (num_isbn)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    nb_chapitres INTEGER,
```

```
    PRIMARY KEY (id_auteur, num_isbn)
);
```

3.2 Écriture plus compacte (bis)

On peut signifier qu'un attribut est une clé primaire dans sa définition même :

SQL

```sql
DROP TABLE IF EXISTS Livre;
CREATE TABLE Livre
(
    num_isbn INTEGER PRIMARY KEY,
    titre    TEXT,
    annee    TEXT,
    CHECK (date(annee) BETWEEN '1900' AND '2100')
);
```

4 Insertion des données dans la BDD

4.1 Données de Pays

SQL

```sql
INSERT INTO Pays
VALUES ('France', 672051, 67064000),
       ('Italie', 301336, 66436000),
       ('Royaume-Uni', 242900, 60317000);
```

Les attributs des tuples sont dans le même ordre que lors de la création.

4.2 Et cætera

De même que lors de la création, on ne peut pas insérer de tuples dans **Auteur** avant d'avoir peuplé **Pays** : en effet dans un tuple de **Auteur** tel que

```
(1, 'France', 'Hugo', 'Victor', '1802-02-26')
```

Les contraintes de référence font qu'un tuple « France » doit d'abord exister dans **Pays**.

5 Exercices

Exercice 1

Ouvrir le fichier `create_Auteurs.sql` qui a servi à créer la BDD du cours (avec le bloc-notes ou autre).

1. Trouver une écriture compacte telle que vue dans le cours.
2. Trouver une écriture compacte qui n'avait pas été vue.
3. En fouillant « à vue d'œil » dans les données
   a. Combien y a-t-il de livres ?
   b. Combien y a t-il d'auteurs ?
   c. Où voit-on clairement qu'un auteur a écrit 2 livres ?

Exercice 2

1. Voici le modèle relationnel de l'exercice **Hotel Reservation Client Chambre** du chapitre précédent :

   Hotel(nom_hotel TEXT, adresse TEXT)

   Chambre(num_ch INTEGER, prix INTEGER, nom_hotel TEXT)

   Client(num_client INTEGER, nom_client TEXT)

   Reservation(num_resa INTEGER, date_resa DATE, num_client INTEGER, num_chambre INTEGER)

   Écrire le script SQL qui crée cette BDD.
2. Insérer 3 valeurs de votre choix dans chaque table.

Exercice 3

On considère une base de données qui ne contient aucune table.
Dire quelles erreurs produisent les séquences SQL suivantes (tout au moins ce qui n'est pas acceptable du point de vue du cours).

1.
```sql
DROP TABLE Client;
CREATE TABLE Client
(
    cid             INTEGER,
    nom             TEXT,
    prenom          TEXT,
    points_fidelite INTEGER NOT NULL,
        PRIMARY KEY (cid),
    CHECK (points_fidelite >= 0)
);
```

2.
```sql
CREATE TABLE Client
(
    cid             INTEGER PRIMARY KEY,
    nom             TEXT,
    prenom          TEXT,
    points_fidelite INTEGER NOT NULL,
    CHECK (points_fidelite >= 0)
);
CREATE TABLE Commande
(
    cid  INTEGER PRIMARY KEY, -- variante plus rapide et valide
    pid  INTEGER,
    date TEXT NOT NULL,
    FOREIGN KEY (cid) REFERENCES Client (cid),
    FOREIGN KEY (pid) REFERENCES Client (pid)
);
CREATE TABLE Produit
(
    pid  INTEGER PRIMARY KEY,
    nom  TEXT,
    prix REAL(10, 2) -- 10 chiffres max avant la virgule, 2
    ↪ après
```

```
);
```

3.
```
CREATE TABLE Client
(
    cid             INTEGER PRIMARY KEY,
    nom             TEXT,
    prenom          TEXT,
    points_fidelite INTEGER NOT NULL,
    CHECK (points_fidelite >= 0)
);
CREATE TABLE Produit
(
    pid  INTEGER PRIMARY KEY,
    nom  TEXT,
    prix REAL(10, 2)
);
CREATE TABLE Commande
(
    cid  TEXT REFERENCES Client (cid),
    nomp  INTEGER REFERENCES Produit (nom),
    date TEXT NOT NULL,
    PRIMARY KEY (cid, pid)
);
```

4.
```
CREATE TABLE Client
(
    cid             INTEGER PRIMARY KEY,
    nom             TEXT,
    prenom          TEXT,
    points_fidelite INTEGER NOT NULL,
    CHECK (points_fidelite >= 0)
);
CREATE TABLE Produit
(
    pid  INTEGER PRIMARY KEY,
    nom  TEXT,
    prix REAL(10, 2)
);
```

```sql
CREATE TABLE Commande
(
    cid  INTEGER,
    pid  INTEGER,
    date TEXT NOT NULL,
    PRIMARY KEY (cid, nomp),
    FOREIGN KEY (cid) REFERENCES Client (cid),
    FOREIGN KEY (pid) REFERENCES Produit (pid)
);

INSERT INTO Commande VALUES(0, 0, '2020-03-02')
```

Exercice 4

Liste les différents attributs de cette relation. Donne le domaine de chaque attribut. Pour chaque attribut dire s'il peut jouer ou non le rôle de clef primaire et pourquoi.

Film

id	titre	ealisateur	ann_sortie	note_sur_10
1	Alien, le huitième passager	Scott	1979	10
2	Dune	Lynch	1985	5
3	2001 : l'Odysée de l'Espace	Kubrick	1968	9
4	Blade Runner	Scott	1982	10

Exercice 5

Indique les attributs qui peuvent servir de lien entre ces deux relations.

Auteur

id	nom	prenom	ann_naiss	langue_ecriture
1	Orwell	George	1903	anglais
2	Herbert	Frank	1920	anglais
3	Asimov	Isaac	1920	anglais
4	Barjavel	René	1911	français
5	Verne	Jules	1828	français
...	...	...	...	...

Livre

id	titre	id_auteur	ann_publi	note
...	...	...	...	...
34	La nuit des temps	4	1968	7
35	De la Terre à la Lune	5	1865	10
36	Les Robots	6	1950	9
...	...	...	...	...

Exercice 6

1. En partant de la relation **Film** ci-dessus, crée une relation **Realisateur** (attributs de la relation : `id`, `nom`, `prenom` et `ann_naissance`, tu trouveras toutes les informations nécessaires sur le Web). Modifie ensuite la relation **Film** afin d'établir un lien entre les relations **Film** et **Realisateur**. Tu préciseras l'attribut qui jouera le rôle de clef étrangère.
2. Écris le code SQL permettant de générer les 2 tables.
3. Écris le code SQL pour insérer les films et les réalisateurs correspondants.