Thème BDD – Partie 2/3

Requêtes SQL (Structured Query Language)

Nous avons comment créer des tables d'une BDD et comment les remplir avec des enregistrements.

L'objectif de ce chapitre est d'expliquer comment mettre à jour des données et comment sélectionner des données à partir de critères divers. Ce sont des requêtes SQL qui permettent de répondre à ces questions.

I) Sélection de données

On considère la table suivante :

CREATE TABLE LIVRE(titre VARCHAR(300) NOT NULL, editeur VARCHAR(90) NOT NULL, annee INT NOT NULL,   
 isbn CHAR(14) PRIMARY KEY);

Voici comment trouver tous les **titres** publiés **après 2000** :

SELECT **titre** FROM LIVRE **WHERE** **annee >= 2000**

En Python

curseur.execute("SELECT titre from livre WHERE annee >= 2000")

for element in curseur :

print(element)

La clause WHERE

Ce qui suit WHERE doit être une expression booléenne (une expression vraie ou fausse). Les opérateurs logiques sont acceptés : OR , AND et NOT.

Une image contenant texte

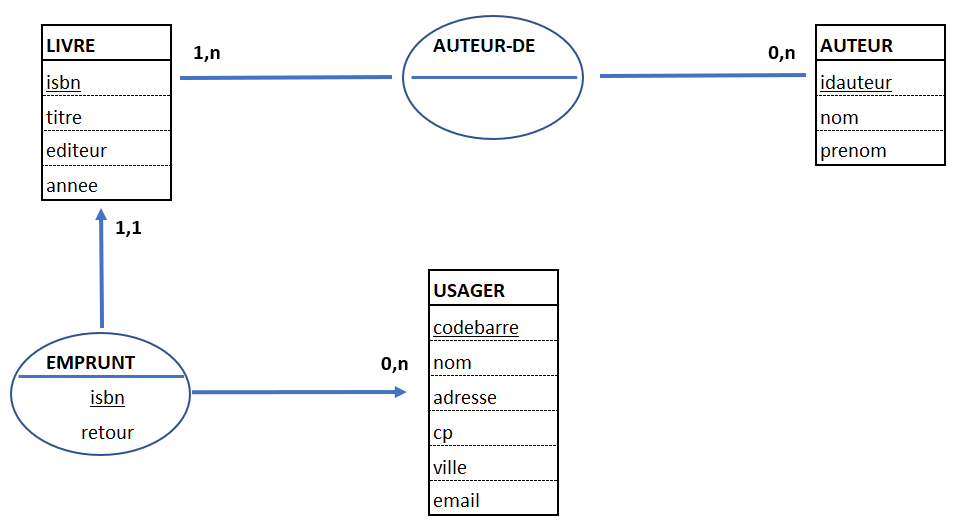
Description générée automatiquementSELECT titre from livre WHERE **annee >= 2000 AND annee <= 2010**

SELECT titre from livre WHERE **annee >= 2000 AND editeur = 'Editions Gallimard'**

On peut chercher les livres dont le titre contient par exemple le mot 'Astérix' :

SELECT titre from livre WHERE **titre LIKE '%astérix%'**

Nous partons du MCD suivant :



Il s'agit donc d'un Modèle Conceptuel de Données d'une Base De Données pour une bibliothèque.  
On a d'abord 3 tables : LIVRE, AUTEUR et USAGER

On a ensuite une table AUTEUR-DE liant les tables LIVRE et AUTEUR.  
Cette table ne comporte aucun attribut à part les clefs étrangères isbn et idauteur.  
La clef primaire de cette table est le couple (isbn;idauteur).

On a enfin une table pour les emprunts liant les tables LIVRE et USAGER.

La clef primaire de cette table est le couple (isbn;retour).   
isbn est le code du libre (une clef étrangère) et retour est la date de retour.  
Remarquons que ce couple est forcément unique.

Voici la **commande** SQL (ou **requête** SQL) pour créer la table USAGER :

**CREATE TABLE** usager(nom VARCHAR(90), adresse VARCHAR(300), cp VARCHAR(5) , ville   
 VARCHAR(40), email VARCHAR(50) , **codebarre** CHAR(15) **PRIMARY\_KEY);**

Cette commande SQL crée la table (vide au départ) usager.  
L'ordre se termine avec le **;** final.  
La clef primaire code\_barre est indiquée à l'aide du mot clef **PRIMARY\_KEY.**

L'ordre des atttributs n'a pas d'importance.

Pour chaque attribut, on précise son type. Il existe de nombreux types.   
VARCHAR(n) est un STRING d'au plus n caractères.

Remarque : cela ressemble au MLD !

Voici la commande SQL pour créer la table LIVRE :

**CREATE TABLE** livre(titre VARCHAR(300), editeur VARCHAR(90), annee INT,  
 isbn CHAR(14) **PRIMARY\_KEY);**

Remarquer le type INT.

Voici la commande SQL pour créer la table AUTEUR :

**CREATE TABLE** auteur(nom VARCHAR(90), prenom VARCHAR(90), idauteur INT **PRIMARY\_KEY);**

Créons la table d'associations AUTEUR-DE

**CREATE TABLE** auteur-de(idauteur INT **REFERENCES** **auteur**(idauteur),  
 **isbn** CHAR(14) **REFERENCES** **livre**(isbn),

**PRIMARY-KEY** (idauteur,isbn));

La clé primaire est bien un couple de deux clés étrangères.   
On indique qu'un attribut est une clé étrangère à l'aide du mot clef **REFERENCES** suivi du nom de la table où se trouve la clé primaire suivi du nom de la clef étrangère entre parenthèses.

Créons la table EMPRUNT.  
Elle contient deux clés étrangères, et la clé étrangère **isbn** est utilisée pour la clé primaire.

**CREATE TABLE** emprunt(**codebarre** CHAR(15) **REFERENCES** **usager**(codebarre),  
 **isbn** CHAR(14) **REFERENCES** **livre**(isbn) ,  
 retour DATE,  
 **PRIMARY-KEY** (**isbn**,retour));   
 );

Remarquer le type DATE.

Le logiciel SGBD envoie cette suite d'ordres SQL, ce qui permet la création des tables. Ces tables sont soit créées physiquement dans le disque dur (fichiers d'extension **.db**) soit créées dans la mémoire vive.

Evidemment ces tables sont créées mais elles sont vides… Disons que la structure de la BDD est créée et va pouvoir accueillir des enregistrements.

**L'indentation, les espaces, la casse, sont ignorées. Elles ne sont là que pour la lisibilité des ordres SQL. L'usage est d'écrire le nom des tables et les attributs en minuscules. L'usage veut également que les noms des tables soient au singulier.**

Ces ordres SQL spécifient le type des différents attributs.

Voici les différents types possibles.

SMALLINT entier 16 bits

INTEGER entier 32 bits  
INT entier 32 bits  
BIGINT entier 64 bits signé

DECIMAL(t,f) décimal signé de t chiffres dont f après la virgule

REAL flottant 32 bits

DOUBLE PRECISION flotttant 65 bits

CHAR(n) Chaîne de n caractères (espaces si chaîne trop petite)

VARCHAR(n) Chaîne d'au plus n caractères

TEXT Chaîne de taille quelconque

BOOLEAN Dépend du SGBD utilisé

DATE Chaîne au format 'AAAA-MM-JJ'

TIME Chaîne au format 'hh:mm:ss'

NULL

Le type DECIMAL(5,2) permet de stocker des décimaux compris entre -999,99 et 999,99.

La valeur NULL est interdite pour les clés primaires. On teste la valeur NULL pour un attribut avec IS NULL ou IS NOT NULL.

**Dans les requêtes SQL, les guillemets sont des guillemets simples. Les retours chariot dans une chaîne sont ignorés.**

On peut interdire à un attribut la valeur NULL et spécifier qu'un attribut doit être unique dans la table (comme la clé primaire). Ces contraintes sont vérifiées par le SGBD.

**CREATE TABLE** usager(nom VARCHAR(90) NOT NULL, adresse VARCHAR(300), cp VARCHAR(5),  
 ville VARCHAR(40), email VARCHAR(50) NOT NULL UNIQUE, **codebarre** CHAR(15) **PRIMARY\_KEY);**

Quand des enregistrements seront créés, le SGBD vérifiera que les attributs nom et email ont bien une valeur.  
Il vérifiera également que les emails sont uniques (bien que l'attribut mail ne soit pas la clef primaire).

Si dans une table on a deux atributs a et b et si on interdit la présence de deux couples (a;b) identiques, alors on ajoute à la fin :

**CREATE TABLE** unetable(…………………………………………… , UNIQUE(a,b) );

Il est possible d'ajouter des contraintes pour les valeurs d'un attribut à l'aide du mot **CHECK** suivi d'une expression booléenne. Ces contraintes sont à placer obligatoirement à la fin de la requête.

**CREATE TABLE** produit(nom VARCHAR(90) NOT NULL, quantite INT NOT NULL,  
 prix DECIMAL(10,2) NOT NULL, **id** INT **PRIMARY\_KEY,  
 CHECK (quantite >=0 AND prix >=0));**

**II) Destruction de tables**

La requête SQL est :

**DROP TABLE** nom\_table

Il n'est pas possible de supprimer une table si elle sert de référence à une autre table pour une clé étrangère. Si on essaie malgré tout, le SGBD renverra un message d'erreur.

Dans ce cas il faut supprimer les tables dans le bon ordre.

Par exemple, on ne peut pas supprimer la table USAGER car elle apparaît sous forme de clef étrangère dans la table emprunt. Il faut donc commencer par supprimer la table EMPRUNT.

**DROP TABLE** emprunt

**DROP TABLE** usager

**III) Insertion dans une table**

La requête SQL est :

**INSERT INTO** nom\_table VALUES (attrib1,attrib2,attrib3) , (attrib1,attrib2,attrib3) ,

(attrib1,attrib2,attrib3) , (attrib1,attrib2,attrib3) ;

Ici, on ajouté 4 enregistrements à nom\_table.  
Clairement nom\_table possède 3 attributs (dont la ou les clefs !)

Pour savoir qui est qui, tout dépend de l'ordre de déclaration des attributs dans la création de la table.

Si on a

**CREATE TABLE** nom\_table (nom VARCHAR(90), prenom VARCHAR(90), idauteur INT **PRIMARY\_KEY);**

alors attrib1 est le nom, attrib2 est le prenom et attrib est un entier (clef primaire)

**Exemple**

**CREATE TABLE** **auteur**(nom VARCHAR(90), prenom VARCHAR(90), annee INT,a\_id INT **PRIMARY\_KEY);**

**INSERT INTO auteur VALUES ('Bernard','Jean',1990,1) ('Dupont','Eric',1989,2) ;**

Le mot clé VALUE est suivi ici de 4-uplets.

Si on souhaite passer les attributs dans un ordre différent, il faut spécifier l'ordre **avant le mot clé VALUES**

INSERT TABLE auteur (id,prenom,nom,annee) VALUES (1,'Jean','Bernard',1990) (2,'Eric','Dupont',1989) ;

Toute tentative d'insertion de données ne respectant les contraintes définies dans la déclaration de la table (CHECK) conduit à un message d'erreur.

**IV) Python et SQL (création d'une bdd et lecture des enregistrements)**

**import sqlite3**

bdd = sqlite3.connect("bdd.db") # *les tables seront stockées dans le fichier en paramètre*

curseur = bdd.cursor() # *obligatoire*

**def creation():**

requete ="""

CREATE TABLE exemple

On reconnaît une requête SQL

Elle est écrite entre deux """  
Les guillements sont triples car il y a plusieurs lignes.

(

id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,

nom TEXT,

nombre INTEGER

);

"""

curseur.execute(requete) # *obligatoire pour exécuter la requête*

bdd.commit() # *obligatoire , le fichier est créé*

curseur.execute("""INSERT INTO exemple VALUES (148,"toto",25),(149,"tata",38);""")

# *On ajoute deux enregistrements dans la table exemple*

bdd.commit() # *obligatoire , le fichier est modifié*

On reconnaît une requête SQL

Voir chapitre suivant

**def lecture():**

requete = " SELECT \* FROM exemple"

curseur.execute(requete) # *obligatoire pour exécuter la requête*

for element in curseur :

print(element) # permet d'afficher tous les enregistrements

Une image contenant texte

Description générée automatiquement

**creation()**

**lecture()**

**bdd.close()** #Déconnexion # *obligatoire*