Chapitre 20 : Base de données relationnelle

I)Introduction

Les bases de données sont très utilisées dans les secteurs de la banque, des assurances, de la finance et de l'administration.
Une *base de données*[1] est un conteneur constitué de *tables* dans lesquelles sont stockées des données. Lorsque ces tables ont des liens entre elles, on parle de *base de données relationnelle*.

Quelques mots à connaître :

– *table* = tableau constitué de lignes et de colonnes,

– *cellule* = intersection d'une ligne et d'une colonne[2] d'une table,

– *valeur* = contenu d'une cellule,

– *attribut* = nom d'une colonne de la table,

– *descripteur* = ensemble des attributs,

– *champ* = attribut ou ensemble d'attributs,

– *enregistrement*[3] = ligne de la table, c'est un objet de type *tuple*,

– *clé primaire*[4] d'une table = attribut (ou groupe d'attributs) permettant d'identifier de manière unique tout enregistrement de la table,

– *clé étrangère*[5] d'une table = à la fois attribut de cette table et clé primaire d'une autre table.

Pour manipuler des tables d'une base de données, on utilise un langage de requêtes, appelé SQL[6]

II)Création d'une table

Pour créer une table dans une base de données existante, on commence par donner un nom à la table et aux colonnes de cette table.
Puis, on exécute la requête CREATE TABLE[7] par la commande :

CREATE TABLE nom_table(attributs et type)

1. database
2. ligne=row et colonne=column
3. record
4. primary key
5. foreign key
6. structured query language
7. On utilise souvent la requête CREATE TABLE IF NOT EXISTS afin d'éviter le message d'erreur : ‘table nom_table already exists’.

Exemple :
CREATE TABLE clients (nom TEXT, prenom TEXT, date INTEGER)

Pour chaque attribut, on peut également définir des options comme NOT NULL ou PRIMARY KEY.

Exemple :
CREATE TABLE clients (numclient INTEGER PRIMARY KEY, nom TEXT, prenom TEXT, date INTEGER)

On déclare ici l'attribut numclient comme clé primaire.

Exemple :
CREATE TABLE clients (nom TEXT NON NULL, prenom TEXT, date INTEGER)

En mettant not null, on oblige tous les clients à avoir un nom.

III)Modification des données d'une table

III.1)Insertion d'un enregistrement

Pour ajouter un enregistrement dans une table existante, on exécute la requête INSERT INTO ... VALUES... par la commande :

INSERT INTO nom_table(attributs)**VALUES**(valeurs)

Exemple :
INSERT INTO clients(nom,prenom,date) VALUES('dupont','daniel',1963)

A l'issue de cette commande, la table clients comporte une ligne de plus, constituée de l'enregistrement ('dupont','daniel',1963).

On peut aussi ne pas préciser les attributs de la table, ce qui simplifie un peu la requête :

INSERT INTO clients VALUES('dupont','daniel',1963)

III.2)Mise à jour des données

Pour remplacer dans la colonne d'une table une valeur donnée par une autre, on exécute la requête UPDATE...SET...WHERE... par la commande :

UPDATE nom_table **SET** attribut=nouvelle_valeur
WHERE condition

Cette mise à jour n'affecte que les cellules d'une colonne donnée dont la valeur vaut « ancienne_valeur ».

III.3)Suppression de données

Dans une table, pour supprimer tous les enregistrements dont un attribut donné prend une valeur donnée, on exécute la requête DELETE FROM...WHERE... par la commande :

DELETE FROM nom_table **WHERE** attribut=valeur

III.4)Suppression d'une table

On utilise la requête DROP TABLE par la commande :

DROP TABLE nom_table

IV)Selection et extraction de données

Pour selectionner et extraire les enregistrements d'une table existante, on exécute la requête SELECT...FROM ou la requête SELECT...FROM...WHERE... si l'on souhaite une recherche plus poussée.

IV.1)Extraction de tous les enregistrements de la table

Pour extraire tous les enregistrements d'une table existante, on exécute la requête SELECT * FROM par la commande[8] :

SELECT * FROM nom_table

Après l'exécution de la requête, tous les enregistrements de la table sont copiés dans une mémoire temporaire, appelé *curseur*.

IV.2)Sélection et extraction de tous les enregistrements d'un champ

On sélectionne ici tous les enregistrements de la table, mais pour chacun d'eux, on n'extrait que les valeurs appartenant à un certain champ de la table, c'est-à-dire appartenant à une colonne particulière ou à un ensemble de colonnes particulières de la table.

On exécute la requête SELECT ... FROM ... par la commande :

SELECT champ **FROM** nom_table

8. Le symbole * souvent utilisé en informatique signifie « tout »

IV.3)Selection et extraction de certains enregistrements de la table

Pour sélectionner et extraire certains enregistrements d'une table existante, on exécute la requête SELECT* FROM ... WHERE ... par la commande :

SELECT * FROM nom_table **WHERE** condition

IV.4)Selection et extraction de certains enregistrements d'un champ

Pour sélectionner et extraire certains enregistrements d'un champ d'une table existante, on exécute la requête SELECT... FROM ... WHERE ... par la commande :

SELECT champ **FROM** nom_table **WHERE** condition

V)Jointure

En faisant le produit cartésien d'une table 1 par une table 2, on obtient une table 3, appelée *jointure*, dont les enregistrements sont formés d'un enregistrement de la table 1, suivi d'un enregistrement de la table 2.

Pour extraire tous les enregistrements de la table 3, on exécute la requête :

SELECT * FROM table 1 **INNER JOIN** table 2

Pour extraire tous les enregistrements de la table 3, en se limitant à un champ[9] donné, on exécute la requête :

SELECT champ **FROM** table 1 **INNER JOIN** table 2

On peut aussi extraire les enregistrements de la table 3 satisfaisant à une condition donnée, en effectuant la requête :

SELECT * FROM table 1 **INNER JOIN** table 2 **ON** condition

On peut enfin extraire les enregistrements de la table 3, en se limitant à un champ donné et satisfaisant à une condition donnée, en effectuant la requête :

SELECT champ **FROM** table 1 **INNER JOIN** table 2 **ON** condition

9. les attributs de ce champ sont des attributs provenant des tables 1 et 2

VI)Commandes non exigibles

VI.1)Fonction COUNT

La fonction COUNT() permet de compter le nombre d'enregistrements dans une table. On utilise la requête :

SELECT COUNT(*) FROM nom_table

VI.2)Fonction MAX

La fonction MAX() permet de retourner la valeur maximale d'un attribut numérique donné dans une table existante. On utilise la requête :

SELECT MAX(attribut) **FROM** nom_table

VI.3)Fonction MIN

La fonction MIN() permet de retourner la valeur minimale d'un attribut numérique donné dans une table existante. On utilise la requête :

SELECT MIN(attribut) **FROM** nom_table

VI.4)Fonction SUM

La fonction d'agrégation SUM() permet de retourner la somme des valeurs d'un attribut numérique donné dans une table existante.
On utilise la requête :

SELECT SUM(attribut) **FROM** nom_table

VI.5)Fonction AVG

La fonction d'agrégation AVG() permet de retourner la moyenne [10] d'un attribut numérique donné dans une table existante.
On utilise la requête :

SELECT AVG(attribut) **FROM** nom_table

. VI.6)Commande ORDER BY

La commande ORDER BY permet de trier les enregistrements d'un champ ou de la table entière par ordre croissant selon une colonne de référence donnée. On utilise la requête :

SELECT * FROM nom_table **ORDER BY** colonne

10. average

. VII)Exercices

exercice 1

Une société de matériel informatique possède un fichier (de type excel) qui répertorie les produits et les clients :

produit	prix	prénom	nom	adresse
pc	600	Jean	Bonnot	Lyon
souris	30	Amélie	Poulain	Valence
lampe	30	Robert	Poulain	Valence
clavier	20	Jean	Bonnot	Lyon
lampe	30	Jean	Bonnot	Lyon
souris	30	Robert	Lingot	Montélimar
clavier	20	Amélie	Poulain	Valence

Reporter ces données dans deux tables reliées entre elles :

− une table *clients* d'attributs *numclient* désignant le numéro du client, *prénom*, *nom* et *adresse*,

− une table *commandes* d'attributs *numcommande* désignant le numéro de la commande, *produit*, *prix* et *numclient*.

Préciser les clés primaires et étrangères de chaque table

exercice 2 On considère la table *clients* ci-dessous :

nom	prénom	date
Martin	Rémy	1970
Roux	Alice	1989
Roux	Philippe	1967
Durand	Arthur	1970
Dupont	Josette	1950
Martin	Blandine	1975

Quel est l'effet de la requête suivante sur la table *clients* ?

UPDATE clients SET date=1971 WHERE date=1970

exercice 3

On considère la table *clients* ci-dessous :

nom	prénom	date
Martin	Rémy	1970
Roux	Alice	1989
Roux	Philippe	1967
Durand	Arthur	1970
Dupont	Josette	1950
Martin	Blandine	1975

Quel est l'effet de la requête suivante sur la table *clients* ?

DELETE FROM clients WHERE nom="Roux"

exercice 4

On considère la table *commandes* ci-dessous :

numcommande	marque	prix
1	Peugeot	20500
2	Audi	25300
3	Citroën	15700
4	Renault	12300
5	Peugeot	18000

Quel est le but de la requête suivante ?

SELECT numcommande,marque FROM commandes

exercice 5

On considère la table *clients* ci-dessous :

numclient	ville	paiement
1	Nice	chèque
2	Montpellier	espèces
3	Marseille	cb
4	Lyon	cb
5	Nice	chèque

Quel est le but de la requête suivante ?

SELECT * FROM clients WHERE paiement="cb"

exercice 6

On considère la table *clients* ci-dessous :

numclient	ville	paiement
1	Nice	chèque
2	Montpellier	espèces
3	Marseille	cb
4	Lyon	cb
5	Nice	cb

Quel est le but de la requête suivante ?

SELECT paiement,ville FROM clients WHERE ville="Nice"

exercice 7

On considère les tables *etudiants* et *controles* ci-dessous :

Table etudiants

id_etudiant	prénom	classe
1	Kilian	1A
2	Léo	1B
3	Zinédine	1A

Tables controles

id_controle	matière	note	id_etudiant
M1	maths	15	1
M1	maths	7	2
M1	maths	12	3
P1	philo	10	1
P1	philo	8	2
P1	philo	16	3

1)Quels sont les enregistrements renvoyés par la requête ci-dessous ?

SELECT * FROM etudiants INNER JOIN controles

2)Quels sont les enregistrements renvoyés par la requête ci-dessous ?

SELECT * FROM etudiants INNER JOIN controles ON etudiants.id_eleve=controles.id_eleve

3)Laquelle des deux requêtes précédentes est la plus pertinente ?

4)Ecrire une requête qui affiche tous les couples (note,prénom) possibles en ordonnant les notes par ordre croissant.

5)Que renvoie la requête ci-dessous ?

SELECT AVG(note) FROM controles

VIII)Correction des exercices

correction exercice 1

clé primaire

Table clients

numclient	prénom	nom	adresse
1	Jean	Bonnot	Lyon
2	Amélie	Poulain	Valence
3	Robert	Poulain	Valence
4	Robert	Lingot	Montélimar

clé primaire

Table commandes

clé étrangère

numcommande	produit	prix	**numclient**
1	pc	600	1
2	souris	30	2
3	lampe	30	3
4	clavier	20	1
5	lampe	30	1
6	souris	30	4
7	clavier	20	2

correction exercice 2

A l'issue de cette commande, dans la colonne *date*, la valeur 1970 a été remplacée par 1971, ce qui donne :

nom	prénom	date
Martin	Rémy	1971
Roux	Alice	1989
Roux	Philippe	1967
Durand	Arthur	1971
Dupont	Josette	1950
Martin	Blandine	1975

correction exercice 3

A l'issue de cette commande, dans la colonne *nom*, la valeur "Roux" a été supprimée, ce qui donne :

nom	prénom	date
Martin	Rémy	1971
	Alice	1989
	Philippe	1967
Durand	Arthur	1971
Dupont	Josette	1950
Martin	Blandine	1975

correction **exercice 4**

La requête met en mémoire les enregistrements (1,"Peugeot"), (2,"Audi), (3,"Citroën), (4,Renault) et (5,Peugeot).

correction **exercice 5**

Le programme recherche la valeur "cb" dans la table et met en mémoire les deux enregistrements qui contiennent cette valeur, c'est-à-dire : (3,"Marseille","cb") et (4,"Lyon","cb").

correction **exercice 6**

Le programme recherche la valeur "Nice" dans les colonnes du champ ville, paiement et met en mémoire les deux enregistrements qui contiennent cette valeur, c'est-à-dire : ("chèque","Nice") et ("cb","Nice").

correction **exercice 7**

1)La requête met en mémoire $3 \times 6 = 18$ enregistrements.
Ce sont les éléments du produit cartésien de la table *étudiants* par la table *controles* :

(1, 'Kilian', '1A', 'M1', 'maths', 15, 1), (1, 'Kilian', '1A', 'M1', 'maths', 7, 2), (1, 'Kilian', '1A', 'M1', 'maths', 12, 3), (1, 'Kilian', '1A', 'P1', 'philo', 10, 1), (1, 'Kilian', '1A', 'P1', 'philo', 8, 2), (1, 'Kilian', '1A', 'P1', 'philo', 16, 3), (2, 'Léo', '1B', 'M1', 'maths', 15, 1), (2, 'Léo', '1B', 'M1', 'maths', 7, 2), (2, 'Léo', '1B', 'M1', 'maths', 12, 3), (2, 'Léo', '1B', 'P1', 'philo', 10, 1), (2, 'Léo', '1B', 'P1', 'philo', 8, 2), (2, 'Léo', '1B', 'P1', 'philo', 16, 3), (3, 'Zinédine', '1A', 'M1', 'maths', 15, 1), (3, 'Zinédine', '1A', 'M1', 'maths', 7, 2), (3, 'Zinédine', '1A', 'M1', 'maths', 12, 3), (3, 'Zinédine', '1A', 'P1', 'philo', 10, 1), (3, 'Zinédine', '1A', 'P1', 'philo', 8, 2), (3, 'Zinédine', '1A', 'P1', 'philo', 16, 3)

2)La requête selectionne dans les 18 enregistrements précédents ceux pour lesquels id_eleve est le même dans la table *etudiants* et dans la table *controles*

(1, 'Kilian', '1A', 'M1', 'maths', 15, 1), (1, 'Kilian', '1A', 'P1', 'philo', 10, 1), (2, 'Léo', '1B', 'M1', 'maths', 7, 2), (2, 'Léo', '1B', 'P1', 'philo', 8, 2), (3, 'Zinédine', '1A', 'M1', 'maths', 12, 3), (3, 'Zinédine', '1A', 'P1', 'philo', 16, 3)

3)La deuxième requête est la plus pertinente.

4)SELECT note,prenom FROM etudiants INNER JOIN controles

$$\text{ON} \quad \underbrace{\text{etudiants.id\_eleve}}_{\text{id\_eleve de la table etudiants}} = \underbrace{\text{controles.id\_eleve}}_{\text{id\_eleve de la table controles}}$$

ORDER BY note

5)La requête renvoie la moyenne des notes obtenues par les trois étudiants.

IX)Programmer en SQL

La création d'une base de données et la gestion des tables se fait à l'aide du logiciel sqlite3.
Pour y avoir accès, on l'importe en tant que module au sein d'un environnement python par la commande **import sqlite3**.
sqlite3 utilise le langage sql différent du langage python.

IX.1)Création d'une base, connexion ou déconnexion à une base de données déjà existante

On utilise la commande :

conn**=sqlite3.connect**("ma_base.**db**")

Si la base de données n'existe pas encore, on lui choisit un nom, par exemple : ma_base.[11] C'est un fichier de type *database*.

On remarquera l'utilisation de la fonction *connect* qui renvoie un objet de type *Connection* auquel on attribue un nom, par exemple conn.
Cet objet permet de se connecter à la base de données.

En fin de programme, on ferme cette connexion avec la commande :

conn.**close()**

IX.2)Création d'un curseur

Une fois connecté à la base de données, on fait appel un autre objet, nommé curseur, de type *Cursor*.
Le curseur est une mémoire tampon, c'est-à-dire une zone de la mémoire vive où l'on peut entreposer temporairement des données, avant de les transférer définitivement dans la base de données.

Pour créer un curseur, on lui donne un nom, par exemple cur, puis on utilise la commande :

11. On peut aussi créer la base de données dans la mémoire vive (RAM) de l'ordinateur en utilisant la commande : conn=sqlite3.connect(" :memory :").
Mais, les données seront effacées après la déconnexion

cur=conn.**cursor()**

Ensuite, on peut demander au curseur d'effectuer une tâche, appellée requête [12] (créer une table, insérer une ligne dans une table, importer une table, etc). Cette requête s'effectue à l'aide de la commande *execute* :

cur.**execute**(""" ...... requête ......""")

La requête est placée de part et d'autre entre deux ou trois guillemets doubles.

IX.3)Structure d'un programme de requêtes sql

import sqlite3
conn=**sqlite3.connect**("ma_base.**db**")
cur=conn.**cursor()**
.
.
.
REQUETES ET INSTRUCTIONS
.
.
.
conn**.commit()**
conn**.close()**

La fonction commit() permet de valider les requêtes précédentes.

IX.4)Affichage des données d'une table ou d'un champ

Après l'exécution de la requête SELECT, le curseur met en mémoire une selection d'enregistrements (table entière, colonne ou champ).
Pour les visualiser, on utilise la commande *fetchall()* [13] :

cur.**execute**(""""**SELECT ....**"""')
print(cur.**fetchall()**)

On peut rencontrer aussi les commandes :
fetchone() qui n'affiche que le premier enregistrement,
fectchmany(p) qui affiche les p premiers enregistrements.

12. query
13. fetch=extraire