Langage SQL
Création et modification d'une base de données

1 Cahier des charges

Avant de créer une base de données (de même pour tout projet informatique) il faut faire ce que l'on appelle le cahier des charges, c'est à dire faire la liste de tous les problèmes à résoudre

Dans le cadre de la création d'une base de données il faut partir des requêtes que l'on aimerait pouvoir faire sur cette base de données

Prenons un exemple : la création d'une base de données pour une bibliothèque municipale :

En tant qu'**emprunteur** j'aimerais pouvoir interroger la base pour savoir si :

1. le livre "Petits poèmes en prose" de Beaudelaire est dans la bibliothèque
2. En combien d'exemplaires ? En combien d'exemplaires disponibles ?Dans quel rayon ?
3. Quels livres de Dostoievski sont disponibles ?
4. Dans combien de temps dois je rendre les livres que j'ai empruntés ?

En tant qu'**administrateur** j'aimerais pouvoir interroger la base pour savoir si :

1. Quels sont les livres "en retard" ?
2. Quels sont les livres "en retard" de plus de trois mois ?
3. Quels sont les emprunteurs qui doivent renouveler leur inscription ?

En tant qu'**administrateur** j'aimerais pouvoir **mettre à jour** la base pour :

1. insérer de nouveaux livres et de nouveaux exemplaires
2. mettre à jour l'abonnement d'un emprunteur
3. changer l'adresse d'un emprunteur
4. Supprimer les prêts d'un emprunteur concernant un certain nombre d'exemplaires rendus

2 Modèle entité-association

Comment décrire ou modéliser une bibliothèque ?

Bien entendu il y a plusieurs façons de le faire et ce qui suit est une façon de le faire parmi d'autres possibles

D'ailleurs un choix de modélisation ici est de ne pas prendre en compte des éventuelles réservations d'exemplaires d'un livre

D'un côté il y a des abonnés ou emprunteurs caractérisés par un identifiant, un nom, un prénom, une adresse et email et une date d'abonnement

Ensuite dans les rayons de la bibliothèque il peut avoir plusieurs exemplaires d'un même livre, par exemple "Petits poèmes en prose" de Beaudelaire se décline en plusieurs livres possibles suivant l'éditeur et la date d'édition et chaque livre se décline en plusieurs exemplaires possibles

Une bibliothèque peut être vue alors comme des **Abonnés** empruntant des **d'Exemplaires** qui sont des instances de **Livres** écrits par des **Auteurs** et édités par des **Editeurs**

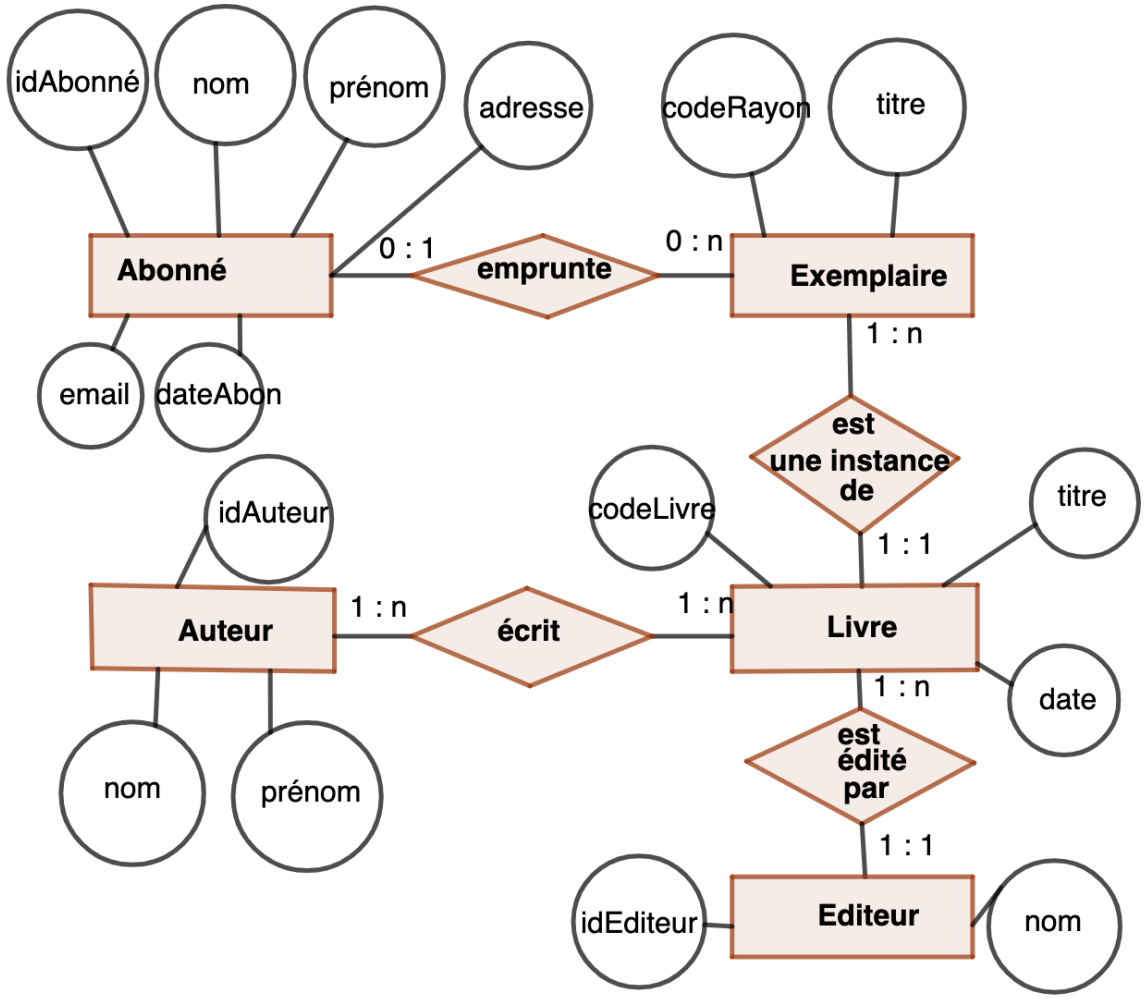
Les mots en gras sont appelés des **entités** et les liens entre les entités exprimés par des verbes sont appelés des **associations**

Les entités ont des **attributs**. Un attribut (ou plusieurs) joue un rôle important car il sert à **identifier chaque entité**

Par exemple l'entité Abonné a un identifiant idAbonné et comme autre attribut nom, prénom, adresse, email, dateAbon (pour la date d'abonnement à renouveler chaque année)

On réalise un schéma pour matérialiser le modèle entité-association

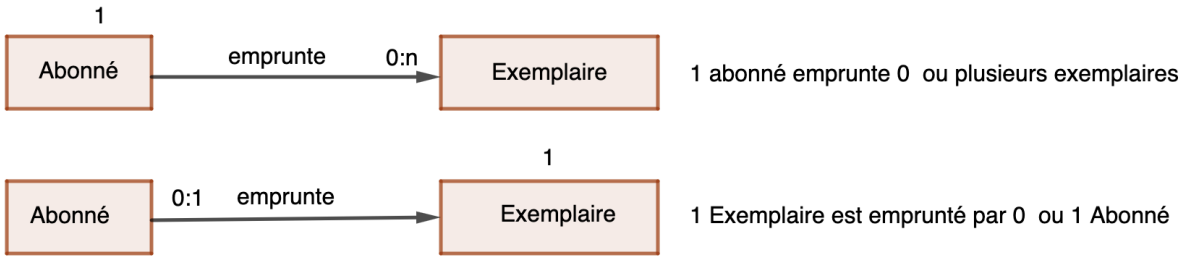
1. Chaque entité est matérialisée par un rectangle et les attributs sont matérialisés par des cercles
2. Chaque association est matérialisée par un losange
3. L'association est caractérisée aussi par des plages de valeurs (voir ci-dessous cardinalité d'une association) dénombrant les associations possibles entre les entités



2.1 Cardinalité d'une association

On suit ici la méthode de modélisation du langage UML *(Unified Modeling Langage)*.

Un abonné peut emprunter aucun exemplaire en ce moment ou plusieurs, ceci se traduit sur le schéma sur le lien entre Abonné et Exemplaire du côté Exemplaire par 0 : n (parfois on écrit aussi 0 : *)



Un exemplaire est disponible en ce moment ou peut être emprunté par un seul abonné, ceci se traduit sur le schéma sur le lien entre Abonné et Exemplaire du côté Abonné par 0 : 1

On dit que l'association **emprunte** est une **relation un à plusieurs**

Le nombre à gauche du symbole : est la cardinalité minimale et le nombre à droite du symbole : est la cardinalité maximale

Un auteur peut écrire un ou plusieurs livres, ceci se traduit sur le schéma sur le lien entre Auteur et Livre du côté Livre par 1 : n (parfois on écrit aussi 1 : *)

Un livre est écrit par un auteur ou plusieurs ceci se traduit sur le schéma sur le lien entre Auteur et Livre du côté Auteur par 1 : n

On dit que l'association **écrit** est une **relation plusieurs à plusieurs**

Nous allons voir que la cardinalité des associations va nous aider pour traduire le modèle entité-association en modèle relationnel

3 Modèle relationnel

On va traduire le modèle entité-association en un modèle relationnel en suivant des règles simples basées sur la cardinalité des associations

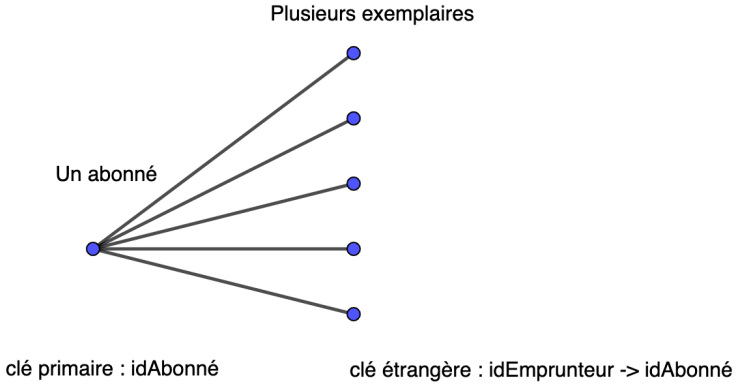
1. **Règle 1 :**

   Toute entité devient une relation avec **au moins** les attributs de l'entité.

   Au moins car suivant la nature des associations peuvent se rajouter des clés étrangères

2. **Règle 2 :**

   Pour chaque association un à plusieurs on crée une **dépendance fonctionnelle** en introduisant une clé étrangère dans la relation où un élément peut être associé à plusieurs éléments d'une autre relation



Par exemple :

(a) Dans la relation Livre on va introduire une clé étrangère idEditeur pointant vers la clé primaire idEditeur de la relation Editeur car un éditeur peut éditer plusieurs livres alors qu'un livre n'est édité que par un seul éditeur

(b) Dans la relation Exemplaire on va introduire une clé étrangère idEmprunteur pointant vers la clé primaire idAbonné de la relation Abonné car un abonné peut emprunter plusieurs exemplaires

(c) Dans la relation Exemplaire on va introduire une clé étrangère idLivre pointant vers la clé primaire codeLivre de la relation Livre car un livre peut avoir plusieurs instances d'exemplaires

3. **Règle 3 :**
   Une association **plusieurs à plusieurs** devient une **relation** de telle sorte que :
   Par exemple l'association écrit devient la relation **Création** (pour création littéraire) avec :

   (a) la clé primaire le couple d'attributs (idAuteur,idLivre)

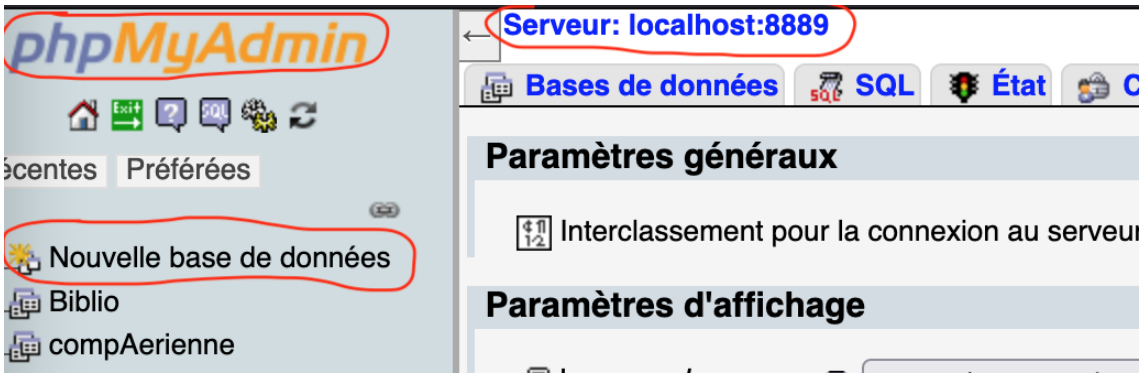
   (b) idAuteur est aussi une clé étrangère pointant vers la clé primaire idAuteur de la relation Auteur

   (c) idLivre est aussi une clé étrangère pointant vers la clé primaire codeLivre de la relation Livre

On obtient comme schéma relationnel :

1. Abonné(idAbonné,nom,prénom,adresse,email,dateAbonnement)
2. Auteur(idAuteur,nom,prénom)
3. Editeur(idEditeur,nom)
4. Livre(codeLivre,titre, date,#idEditeur)
5. Exemplaire(codeRayon,datePret,#idLivre,#idEmprunteur)
6. Création(#idAuteur,#idLivre)

4 *phpMyAdmin*

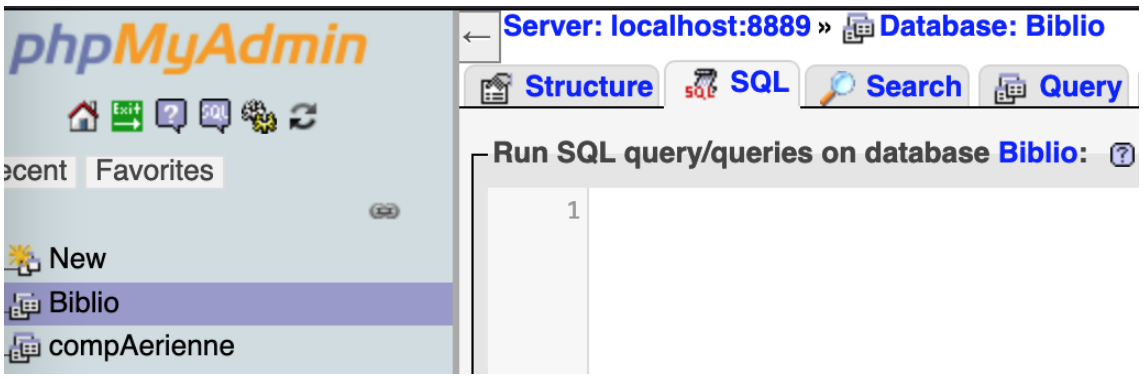


phpMyAdmin est une application graphique permettant de gérer une base de données située sur un serveur (distant ou en local) livrée avec MAMP, LAMP , WAMP (.....) ou XAMP

La partie pratique sera détaillée en TP

Dans un premier temps cliquer sur le logo phpMyAdmin pour mettre l'application en français

Ensuite cliquer sur Nouvelle base de données, puis entrer le nom de la base de données par exemple **Biblio** puis cliquer sur créer



Nous allons dans l'ordre :

1. Créer les tables en respectant les **contraintes référentielles** :
   Les tables contenant des clés étrangères faisant référence à des clés primaires d'autres tables, doivent être créées après celles contenant les clés primaires référencées.
   A la fin de cette étape on a uniquement une coquille vide, la structure logique de la base de données
2. Il faut ensuite "remplir" ou insérer des valeurs dans la base de données en respectant l'ordre imposé par les contraintes référentielles.

On commence par sélectionner l'onglet SQL et entrer les lignes de commandes SQL suivantes pour créer la première table Abonné

```
CREATE TABLE Abonné(
idAbonné int(3),
nom varchar(30),
prénom varchar(30),
adresse varchar(30),
email varchar(30),
dateAbonnement date,
PRIMARY KEY (idAbonné));
```

On ne va pas procéder ainsi table après table mais plutôt procéder de la manière suivante

1. Ouvrir un éditeur de texte comme notepad ++
2. On va créer un fichier biblio.sql dans lequel on va entrer toutes les commandes de création pour **mémoriser toutes les commandes**

```
-- Création des tables --

CREATE TABLE Abonné(
idAbonné int(3) not null AUTO_INCREMENT,
nom varchar(30) not null,
prénom varchar(30) not null,
adresse varchar(30) not null,
```

```
email varchar(30) not null,
dateAbo date not null,
PRIMARY KEY (idAbonné) );

CREATE TABLE Auteur(
idAuteur int(3) not null AUTO_INCREMENT,
nom varchar(30) not null,
prénom varchar(30) not null,
PRIMARY KEY (idAuteur));

CREATE TABLE Editeur(
idEditeur  int(3) not null AUTO_INCREMENT,
nom varchar(30) not null,
PRIMARY KEY (idEditeur));

CREATE TABLE Livre(
codeLivre int(3) not null AUTO_INCREMENT,
titre varchar(30) not null,
année int(11) not null,
idEditeur int(3) not null,
PRIMARY KEY (codeLivre),
FOREIGN KEY (idEditeur) REFERENCES Editeur(idEditeur));

CREATE TABLE Exemplaire(
codeRayon int(3) not null AUTO_INCREMENT,
datePrêt date,
idLivre int(3) not null,
idEmprunteur int(3),
PRIMARY KEY (codeRayon),
FOREIGN KEY (idLivre) REFERENCES Livre(codeLivre),
FOREIGN KEY (idEmprunteur) REFERENCES Abonné(idAbonné));


CREATE TABLE Création(
idAuteur int(3) not null,
idLivre int(3) not null,
PRIMARY KEY (idAuteur, idLivre),
FOREIGN KEY (idAuteur) REFERENCES Auteur(idAuteur),
FOREIGN KEY (idLivre) REFERENCES Livre(codeLivre));
```

Ensuite on copie et colle toutes les commandes dans la zone de commandes SQL, on corrige les éventuelles erreurs puis on exécute toutes les commandes en une seule fois

La base Biblio est crée mais vide !

Ensuite avec l'outil Concepteur on peut voir les **dépendances fonctionnelles** entre les tables

Il faut maintenant remplir la base de données avec quelques valeurs

5 Insertion

On doit faire attention à l'ordre d'insertion et respecter les dépendances fonctionnelles

Ainsi on insère d'abord les n-uplets dans la table Editeur avant ceux de la table Livre car il existe une clé étrangère dans la table Livre référençant la clé primaire de la table Editeur

A la fin du fichier biblio.sql on va insérer les valeurs suivantes dans l'ordre de création des tables

```
-- Insertion de valeurs

 INSERT INTO Abonné(idAbonné,nom,prénom,adresse,email,dateAbo)  VALUES
(1,'Hynome','Paul','2 rue des fraises Paris','p.hyn@bd.fr','2023-01-01')
(2,'Hochon','Paul','3 rue des prunes Paris','p.hoch@bd.fr','2023-01-01')
(3,'Mauve','Guy','4 rue des bananes Paris','g.mau@bd.fr','2023-01-01'),
(4,"Neymar","Jean","5 rue des poires Paris",'j.ney@bd.fr','2023-01-01');


 INSERT INTO Auteur(idAuteur,nom,prénom) VALUES
 (1,'Poe','Edgar'),
 (2,'Beaudelaire','Charles'),
 (3,'Char','René'),
 (4,'Gary','Romain'),
 (5,'Perec','Georges');

 INSERT INTO Editeur(idEditeur,nom) VALUES
(1,'Gallimard'),
(2,'Seuil'),
(3,'Hatier');

 INSERT INTO Livre(codeLivre,titre,année,idEditeur) VALUES
(1,'Fureur et mystère',1962,1),
(2,'Histoires extraordinaires',1973,1),
(3,"L'affaire homme",2005,1),
(4,'Cantatrix sopranica L.',1991,2),
(5,'Les Fleurs du mal',2003,3),
(6,'Les racines du ciel',1978,1);

 INSERT INTO Exemplaire(codeRayon,datePrêt,idLivre,idEmprunteur) VALUES
 (1,'2023-11-01',1,3),
 (2,null,1,null),
 (3,'2023-10-25',2,3),
 (4,'2023-11-12',3,1),
 (5,4),
 (6,5),
 (7,6),
 (8,6);
```

```
INSERT INTO Création(idAuteur,idLivre) VALUES
(3,1),
(1,2),
(4,3),
(5,4),
(2,5),
(4,6);
```

Ensuite on copie et colle la partie insertion de valeurs dans la partie SQL et on exécute
On a ainsi inséré **quelques valeurs** dans les tables

6 Interrogation de la base

On peut déjà avec les quelques valeurs que l'on a introduit interroger la base de données
Par exemple (à faire en exercice pratique ajouter les commandes dans le fichier biblio.sql)

```
-- Interrogation --
SELECT titre FROM Livre;
SELECT nom,prénom FROM Auteur;
SELECT titre FROM Livre WHERE année < 2000;
```

Exécuter les requêtes SQL permettant de :

1. Afficher tous les titres des livres
2. Afficher les noms et prénoms des auteurs
3. Afficher les titres de livres dont l'année d'édition est antérieure à 2000
4. Afficher combien de livres de Romain Gary
5. Afficher combien d'exemplaires de Romain Gary
6. Afficher le nombre total d'exemplaires.
   Afficher uniquement le nombre total d'exemplaires en rayon (non empruntés).
7. Afficher le titre des livres des exemplaires empruntés.
   Afficher le titre et le nom de l'auteur des livres des exemplaires empruntés

7 Mise à jour d'un n-uplet d'une table

1. Imaginons que l'abonné dont l'idAbonné est 1, a déménagé et a changé d'adresse, il habite maintenant 7 rue des orchidées dans ce cas on entre la commande SQL

   ```
   -- mise à jour --

   UPDATE Abonné SET adresse = '7 rue des orchidées'
   WHERE idAbonné = 1;
   ```

2. On veut prolonger le prêt de l'exemplaire de clé primaire 1 en mettant la date de prêt à la date du jour (imaginons que la date du jour est 17 novembre 2023)

```
-- mise à jour --

UPDATE Exemplaire SET datePrêt = '2023-11-17' WHERE codeRayon = 1;
```

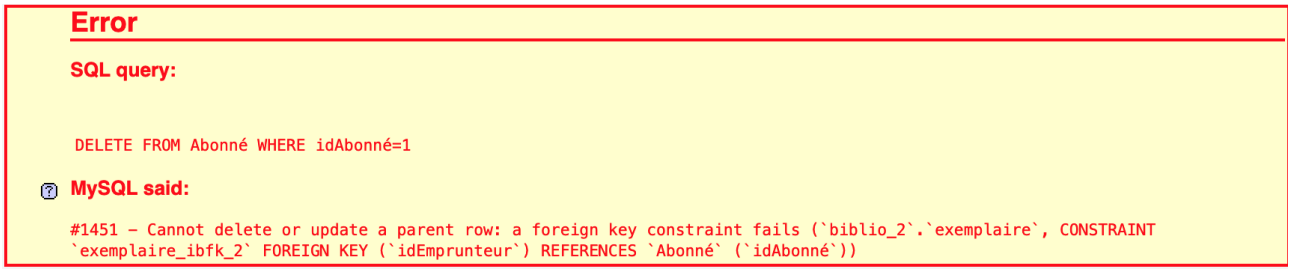
8 Suppression de n-uplets dans une table

Attention on doit aussi ici respecter les dépendances fonctionnelles

Ainsi on ne peut pas supprimer un n-uplet dans la table Abonné sans avoir au préalable supprimé tous les n-uplets en lien avec cet abonné

Supposons que l'abonné dont l'identifiant est 1 quitte précipitamment la région après avoir rendu tous les exemplaires empruntés

1. On ne peut donc pas supprimer directement dans la table `Abonné` l'abonné dont l'identifiant est 1, car sinon on aura un message d'erreur nous prévenant que cet abonné a emprunté des exemplaires.

   Donc on commence par mettre à jour toutes les lignes de la table Exemplaire dont l'idEmprunteur est 1 car il a rendu tous les livres qu'il a emprunté, on utilise alors la commande SQL suivante

   ```
   UPDATE Exemplaire SET idEmprunteur = null, datePret = null
   WHERE idEmprunteur = 1;
   ```

2. Puis on supprime dans la table Abonné cet abonné

   ```
   DELETE FROM Abonné WHERE idAbonné = 1;
   ```

On n'a jamais vu un utilisateur ou un administrateur interroger ou mettre à jour une base de données en faisant directement des requêtes en SQL

C'est pour cela qu'il faut mettre dans le coup un langage de programmation qui guide l'utilisateur par un menu et traduit ses choix en requêtes SQL (Python ou PHP)

9 Python et le module sqlite

Voici les premières lignes de la documentation Python `https://docs.python.org/fr/3/library/sqlite3.html`

SQLite est une bibliothèque C qui fournit une base de données légère sur disque ne nécessitant pas de processus serveur et qui utilise une variante (non standard) du langage de requête SQL pour accéder aux données. Certaines applications peuvent utiliser SQLite pour le stockage de données internes. Il est également possible de créer une application prototype utilisant SQLite, puis de modifier le code pour utiliser une base de données plus robuste telle que PostgreSQL ou Oracle.

9.1 Création de la base

Voici le code Python pour la création d'une base de données, ici le fichier `biblio.db`
On met les commandes SQL de création dans une constante CREATION de type str puis

1. On crée un objet de type connexion
2. On crée un objet de type Curseur
3. cet objet a une méthode `executescript()`
4. On ferme la connexion

```
import sqlite3

CREATION = """
CREATE TABLE Abonné(
idAbonné INTEGER,
nom TEXT,
prénom TEXT,
adresse TEXT,
email TEXT,
dateAbo INTEGER,
PRIMARY KEY (idAbonné));

CREATE TABLE Auteur(
idAuteur INTEGER,
nom TEXT,
prénom TEXT,
PRIMARY KEY (idAuteur));

CREATE TABLE Editeur(
idEditeur INTEGER,
nom TEXT,
PRIMARY KEY (idEditeur));

CREATE TABLE Livre(
codeLivre INTEGER,
titreTEXT,
année TEXT,
idEditeur INTEGER,
PRIMARY KEY (codeLivre),
FOREIGN KEY (idEditeur) REFERENCES Editeur(idEditeur));
```

```
CREATE TABLE Exemplaire(
codeRayon INTEGER,
idLivre INTEGER,
PRIMARY KEY (codeRayon),
FOREIGN KEY (idLivre) REFERENCES Livre(codeLivre));

CREATE TABLE Prêt(
idAbonné INTEGER,
idExemplaire INTEGER,
datePrêt  TEXT,
PRIMARY KEY (idAbonné, idExemplaire),
FOREIGN KEY (idAbonné) REFERENCES Abonné(idAbonné),
FOREIGN KEY (idExemplaire) REFERENCES Exemplaire(codeRayon));

CREATE TABLE Création(
idAuteur INTEGER,
idLivre INTEGER,
PRIMARY KEY (idAuteur, idLivre),
FOREIGN KEY (idAuteur) REFERENCES Auteur(idAuteur),
FOREIGN KEY (idLivre) REFERENCES Livre(codeLivre));
"""
#on crée une connexion avec la base de données
connexion = sqlite3.connect('biblio.db')

#on crée une variable de type Cursor
biblio = connexion.cursor()

#on créé les tables
biblio.executescript(CREATION)

#on ferme la connexion
connexion.close()
```

9.2 Insertion des valeurs

Ici il faut distinguer **une première insertion de nombreuses données** (et ici on peut entrevoir la solution d'un transfert de données, sauvegarde d'une base de données dans une autre ou utilisation d'un fichier .csv qui par un programme Python est transformé en une liste de tuples qui ensuite sont insérés dans la base de données (TP)), d'une insertion de quelques éléments peu nombreux, par exemple entrer des nouveaux abonnés à la bibliothèque municipale.

Dans ce dernier cas, par un menu Python ou un formulaire Html un administrateur peut insérer des valeurs dans la table Emprunteur les informations nom, prénom, adresse, email récupérées

```
import sqlite3

#on crée une connexion avec la base de données
```

```
connexion = sqlite3.connect('biblio.db')

#on crée une variable de type Cursor
biblio = connexion.cursor()

#on insère les nouveaux abonnés
while True:
    id = int(input("id ? "))
    nom = input("nom ? ")
    prenom = input("prenom ? ")
    adresse = input("adresse ? ")
    email = input("email ? ")
    dateAbo = input("date abonnement ? ")
    biblio.execute('INSERT INTO Abonné VALUES (?,?,?,?,?)',
     (id,nom,prenom,adresse,email,dateAbo))
    choix = input('On continue ? (O/n)')
    if choix == "n":
        break


#on valide les modifications
connexion.commit()
#on ferme la connexion
connexion.close()
```

9.3 Modification de certains attributs

L'administrateur veut modifier les adresses de certains abonnés qui ont déménagé

```
import sqlite3

#on crée une connexion avec la base de données
connexion = sqlite3.connect('biblio.db')

#on crée une variable de type Cursor
biblio = connexion.cursor()

#on modifie les adresses d'abonnés
while True:
    id = int(input("id ? "))
    adresse = input("nouvelle adresse ? ")

    biblio.execute('UPDATE Abonné SET adresse = ? WHERE idAbonne = ?',
     (adresse,id))
    choix = input('On continue ? (O/n)')
    if choix == "n":
        break
```

```
#on valide les modifications
connexion.commit()
#on ferme la connexion
connexion.close()
```

9.4 Suppression de certaines lignes

L'administrateur veut supprimer les prêts de tel utilisateur qui a rendu un certain nombre d'exemplaires

```
import sqlite3

#on crée une connexion avec la base de données
connexion = sqlite3.connect('biblio.db')

#on crée une variable de type Cursor
biblio = connexion.cursor()

#on supprime des emprunts
while True:
    idAbonne = int(input("idAbonné ? "))
    idExemplaire = int(input("idExemplaire ? "))

    biblio.execute('DELETE FROM Prêt WHERE idAbonné = ? and idExemplaire
     (idAbonne,idExemplaire))
    choix = input('On continue ? (O/n)')
    if choix == "n":
        break


#on valide les modifications
connexion.commit()
#on ferme la connexion
connexion.close()
```

9.5 Requêtes d'interrogation

Un utilisateur veut interroger la base de données pour avoir tous les livres (pas les exemplaires) de la bibliothèque d'un auteur donné par exemple René CHAR

La requête SQL (jointure) pour cela est :

```
SELECT nom, titre

FROM Auteur as a JOIN Livre as l ON a.idAuteur = l.idAuteur
```

```
WHERE  a.nom = 'CHAR'
```

SQLITE n'exprime pas les jointures comme SQL

```
import sqlite3

#on crée une connexion avec la base de données
connexion = sqlite3.connect('biblio.db')
connexion.row_factory = sqlite3.Row

#on crée une variable de type Cursor
biblio = connexion.cursor()


#on visualise tous les livres d'un Auteur
biblio.execute("SELECT nom, titre FROM Auteur as a  JOIN Livre as l ON
a.idAuteur =  l.idAuteur WHERE a.nom = 'CHAR'")
for row in biblio:
    print('{} : {}'.format(row['nom'],row['titre']))

#on ferme la connexion
connexion.close()
```

10 PHP

On verra en TP l'utilisation de PHP pour interagir avec une base de données située sur un serveur

11 Exercices

Ex 1

Dans le but de créer une base de données Messagerie on cherche à modéliser l'activité d'envoi et de réception de mails pour un particulier

Dans un premier temps on isole deux entités : Contact et Message

1. Contact concerne la personne à qui on envoie un mail ou qui nous a envoyé un mail.
   Quels attributs proposez vous pour l'entité Contact ?
2. Un message est caractérisé par un identifiant idMessage, un contenu, une date dateEnvoi
   Une première association entre les entités Contact et Message est "émet"
   Définir la cardinalité de cette association
3. Parfois **un contact envoie un message à plusieurs destinataires** on va donc introduire une relation supplémentaire entre les entités Contact et Message "envoie à destination de "
   Définir la cardinalité de cette association
4. Enfin on aimerait suivre le fil de discussion entre deux contacts en introduisant une association entre l'entité Message et elle-même "succède à "
   Définir la cardinalité de cette association

Ex 2

Dans le but de créer une base de données Clients pour un atelier de réparation de voitures on cherche à modéliser l'activité de cet atelier

Dans un premier temps on isole les entités suivantes : Voiture - Client - Marque - Réparation
Par exemple une voiture est caractérisée par les attributs suivants :

1. idVoiture : un entier identifiant la voiture
2. catégorie : par exemple citadine, break, etc...
3. nom : une chaîne de caractères par exemple : Peugeot, Renault, Fiat, ...
4. modèle : par exemple pour une Peugeot : 108, 308, etc...
5. année : l'année de mise en circulation

Questions :

1. Quels attributs proposez vous pour l'entité Client ?
2. On suppose que l'atelier n'utilise uniquement des pièces détachées d'origine fournies par le constructeur de la marque de la voiture par conséquent quels attributs proposez vous pour l'entité Marque pour que l'atelier puisse passer une commande au fournisseur ?
3. Une réparation fait suite à une demande d'un client. Quels attributs proposez vous pour l'entité Réparation ?
4. Proposer un modèle entité-association pour l'activité de cet atelier

Ex 3

On souhaite modéliser l'activité d'un office de tourisme uniquement sur la partie location de logements à des particuliers et proposition d'activités en rapport

Par exemple on peut louer un appartement F3 dans la ville de Gavarnie (Hautes-Pyrénées) et réserver une randonnée dans le cirque de Gavarnie pour une journée

On isole les entités suivantes : Logement - Voyageur - Activité

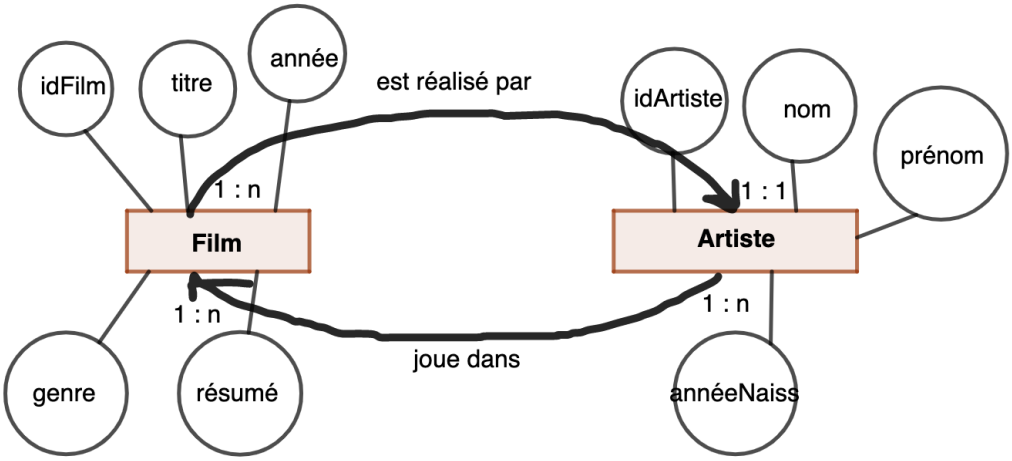
1. Quels attributs proposez vous pour l'entité Voyageur ?
2. Quels attributs proposez vous pour l'entité Logement ?
3. Quels attributs proposez vous pour l'entité Activité ?
4. Proposer un modèle entité-association de la location de logements à des particuliers et proposition d'activités en rapport

Ex 4

On modélise une cinémathèque destinée à un usage personnel avec le modèle entité-association suivant

Pour simplifier la modélisation on dit qu'un film n'est réalisé que par un seul réalisateur

Traduire le modèle entité-association en un modèle relationnel



Ex 5

Traduire les modèles entité-association des exercices 1, 2 et 3 en schémas relationnels

Ex 6

Ecrire en langage SQL les commandes pour créer les relations de la base de données Messagerie de l'exercice 1

Ex 7

Ecrire en langage SQL les commandes pour créer les relations de la base de données Garage de l'exercice 2

Ex 8

Ecrire en langage SQL les commandes pour créer les relations de la base de données Tourisme de l'exercice 3

Ex 9

Ecrire en langage SQL les commandes pour créer les relations de la base de données Films de l'exercice 4

Ex 10

Ecrire en langage SQL les commandes pour insérer quelques valeurs de votre choix dans la base de données Tourisme

Ex 11

Ecrire en langage SQL les commandes pour insérer quelques valeurs de votre choix dans la base de données Films

Ex 12

Ecrire en langage SQL les commandes pour faire quelques mises à jour dans la base de données Tourisme

Ex 13

Ecrire en langage SQL les commandes pour supprimer quelques valeurs de votre choix dans la base de données Tourisme

Ex 14

Ecrire en langage SQL les commandes pour interroger la base de données Biblio du cours pour

1. Afficher tous les titres des livres
2. Afficher les noms et prénoms des auteurs
3. Afficher les titres de livres dont l'année d'édition est antérieure à 2000
4. Afficher combien de livres de Romain Gary
5. Afficher les livres de Romain Gary dans l'ordre croissant de l'année d'édition
6. Afficher combien d'exemplaires de Romain Gary
7. Il se trouve que tous les exemplaires des 'Racines du ciel' de Romain Gary ont été empruntés. Afficher les noms et prénoms des emprunteurs
8. Afficher les noms et prénoms des abonnés dont la durée de prêt est supérieure à 15 jours