TP #1

Mise en œuvre d'une base de données SQLite3 sous Python

L'objet de ce TP est permettre la prise en main de l'outil SQLite dans sa version 3 à travers l'interface de programmation (API) destinée à une exploitation avec le langage Python. Il s'agit de procéder à des opérations usuelles de définition et de manipulation de données.

Modèle logique/physique de données à considérer

Le modèle logique de données à prendre en compte est présenté sur la figure 1. Le type des valeurs des différents est précisé sur le modèle.

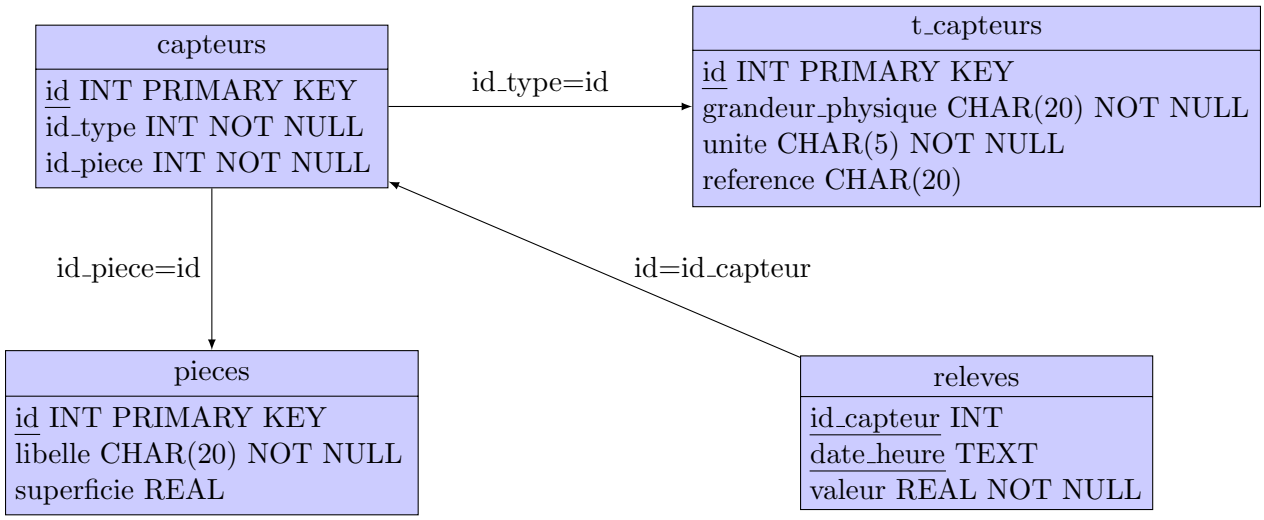

Fig. 1 – *Modèle logique des données*

Requêtes de définition de données

Quatre requêtes sont nécessaires pour définir la structure de la base de données. Chacune de ces requêtes est associée à une table du modèle logique/physique. Le code SQL de ces requêtes est donné ci-dessous.

L'ordre de création des tables répond à une logique d'implémentation préalable des champs hérités par clé étrangère. Il est en effet nécessaire qu'une table à l'origine d'une clé étrangère soit créée avant que cette clé soit appliquée à une autre table.

```
CREATE TABLE IF NOT EXISTS t_capteurs (
   id INT PRIMARY KEY,
   grandeur_physique CHAR(20) NOT NULL,
   unite CHAR(5) NOT NULL,
   reference CHAR(20));

CREATE TABLE IF NOT EXISTS pieces (
   id INT PRIMARY KEY,
   libelle CHAR(20) NOT NULL,
   superficie REAL);

CREATE TABLE IF NOT EXISTS capteurs (
   id INT PRIMARY KEY,
   id_type INT NOT NULL,
   id_piece INT NOT NULL,
   FOREIGN KEY (id_type) REFERENCES t_capteurs(id),
   FOREIGN KEY (id_piece) REFERENCES pieces(id));

CREATE TABLE IF NOT EXISTS releves (
   id_capteur INT NOT NULL,
   date_heure TEXT NOT NULL,
   valeur REAL NOT NULL,
   PRIMARY KEY (id_capteur, date_heure),
   FOREIGN KEY (id_capteur) REFERENCES capteurs(id));
```

Traîtement d'une requête

Le traîtement d'une requête SQL s'effectue en plusieurs étapes :

1. Connexion à la base de données.
2. Initialisation d'un curseur.
3. Exécution de la requête.
4. Récupération des lignes/enregistrements
5. Fermeture de la connexion à la base de données.

Le code ci-dessous présente un exemple simplifié de traîtement d'une requête `SELECT` destinée à la lecture de l'ensemble des enregistrements contenus dans la table `t_capteurs` :

```
import sqlite3

connexion = sqlite3.connect('metrologie.db')
curseur = connexion.cursor()
requete = '''SELECT * FROM t_capteurs;'''
curseur.execute(requete)
lignes = curseur.fetchall()
connexion.close()

for ligne in lignes:
    print(ligne)
```

L'exemple présenté intègre également une boucle d'affichage des enregistrements collectés.

Insertion d'un nouveau relevé

L'insertion d'un nouveau relevé n'est possible que lorsque les trois autres tables sont correctement définies et qu'elles intègrent des enregistrements. Il s'agit alors de consigner des enregistrements dans la table `releves` à l'aide de l'ordre SQL `INSERT INTO`.

Il est important de noter que le champ `date_heure` doit être affecté en considérant la date et de l'heure courantes. Pour ce faire, le langage Python met à disposition du programmeur des fonctions spécifiques qu'il convient d'utiliser.

Le bout de code ci-après montre un exemple de collecte de la date et de l'heure courantes dans une variable `dateHeure` :

```
import datetime

dateHeure = datetime.datetime.now()
print(dateHeure)
```

Une chaîne de caractères correspondant à la date et l'heure courantes est ainsi stockée dans la variable `dateHeure` pour être ensuite affichée.

Il est alors aisé de procéder à l'insertion d'un nouvel enregistrement dans la table `releves` comme le montre l'exemple suivant :

```
import sqlite3
import datetime

connexion = sqlite3.connect('metrologie.db')
curseur = connexion.cursor()

capteur = 1
valeur = 22.5
dateHeure = datetime.datetime.now()

curseur.execute("INSERT INTO releves VALUES (?, ?, ?);",
                (capteur, dateHeure, valeur))
connexion.commit()
connexion.close()
```

À noter que la fonction `commit()` appliquée à la variable `connexion` permet de valider les modifications apportées à la base de données avant que la connexion ne soit fermée.

Travail à réaliser

En vous basant sur le tutoriel *Python SQLite*, vous devez réaliser successivement les trois programmes Python suivants :

1. `def.py` : Création des quatre tables.
2. `init.py` : Insertion de données arbitraires dans les tables en respectant l'ordre `t_capteurs`, `pieces` et `capteurs`.
3. `exploit.py` : Affichage de la liste des enregistrements contenus dans les tables et insertion d'un enregistrement dans la table `releves`.

Le fichier de base de données doit être nommé `metrologie.db`.

Vous penserez à bien commenter le code et à y insérer des lignes de production de message permettant à l'utilisateur de suivre le déroulement du programme.

Il est recommandé d'exploiter l'environnement (*shell*) SQLite pour vérifier la bonne exécution des requêtes.