Bases de données
Série n°2 : AR ~ SQL / Python ~ SQLITE

Exercice 2 : Concours blanc 2016-2017 ~ CORRECTION

Un institut veut gérer son parc informatique à l'aide d'une base de données qui décrit les installations de logiciels sur des postes de travail.
Un poste de travail est une machine sur laquelle sont installés certains logiciels, gratuits ou payants.
Pour chaque installation on enregistre la date d'installation et la durée de la licence (en mois). Lorsque le logiciel est gratuit, on saisit la valeur "infini" pour la durée de licence.

On a établi le modèle relationnel ci-dessous, dans lequel les données sont organisées sous forme de relations :

- Logiciel (**id**, nom, version, prix)
- Installer (***#id_poste, #id_logiciel***, date_installation, duree_licence)
- Poste(**id**, nom, *#id_salle*)
- Salle(**id**, nom, nb_prises_reseau)

Les attributs de la relation **Logiciel** sont :
• **id** : identifiant, entier naturel non nul, s'incrémente automatiquement.
• **nom** : attribut non nul de type chaine de caractères.
• **version** : attribut non nul de type chaine de caractères, dans la mesure où la version d'un logiciel s'écrit sous la forme de nombres séparés par un ou plusieurs points, comme 5.4.1 par exemple.
• **prix** : un nombre décimal avec deux chiffres après la virgule. Un logiciel gratuit a un prix égal à 0.

Les attributs de la relation **Installer** sont :
• **id_poste** : a le même domaine que l'attribut id de la relation poste . En effet, c'est une clé étrangère qui fait référence à la clé primaire id de la relation poste.
• **id_logiciel** : a le même domaine que l'attribut **id** de la relation logiciel . En effet, c'est une clé étrangère qui fait référence à la clé primaire id de la relation logiciel.
• **date_installation** : est un attribut non nul de type DATE.
• **duree_licence** : est un attribut non nul, de type chaine de caractères dans la mesure où on saisit la valeur "infini" lorsque le logiciel est gratuit et un nombre entier naturel de mois sinon.

Les attributs de la relation **Poste** sont :
• **id** : identifiant, entier naturel non nul, s'incrémente automatiquement.
• **nom** : attribut non nul de type chaine de caractères.
• **id_salle** : a le même domaine que l'attribut id de la relation salle . En effet, c'est une clé étrangère qui fait référence à la clé primaire id de la relation salle.

Les attributs de la relation **Salle** sont :
• **id** : identifiant, entier naturel non nul, s'incrémente automatiquement.
• **nom** : attribut non nul, unique, de type chaine de caractères.
• **nb_prises_reseau** : entier naturel.

Afin de mieux manipuler la base de données de l'institut, on proposera les classes Python suivantes :

➢ une **classe Logiciel** définie par :
(a) une méthode constructeur définissant les attributs
• **nom** pour le nom du logiciel
• **version** pour la version du logiciel
• **prix** pour son prix d'achat.
(b) une méthode **enregistrer(self)** permettant d'enregistrer un logiciel dans la table Logiciel.

anis_saied@hotmail.com

(c) une méthode **chercher(self)** permettant de chercher un logiciel dans la table Logiciel par son nom et sa version et retourne son identifiant.
(d) une méthode **__str__(self)** pour visualiser un logiciel (nom, version, prix).

➢ une **classe Poste** définie par :
(a) une méthode constructeur définissant les attributs
• **nom** pour le nom du poste.
• **liste_logiciels** pour la liste des logiciels installés sur un poste. Cette liste sera initialement vide.
(b) une méthode **chercher(self, idSalle)** permettant de chercher un poste dans la table Poste par son nom et la salle où il est installé et retourne son identifiant.
(c) une méthode **installer (self, logiciel, date , licence)** qui permet d'installer un logiciel sur un poste à une date avec une licence, données en paramètres.
(d) une méthode **charger_liste_logiciels (self)** permettant de mettre à jour la liste_logiciels par les logiciels installés sur un poste à partir de la base de données.
(e) une méthode **__str__(self)** pour visualiser la liste des logiciels installés sur un poste.

➢ une **classe Salle** définie par :
(a) une méthode constructeur définissant les attributs
• **nom** pour le nom de la salle.
• **liste_postes** pour la liste des postes localisés dans une salle. Cette liste sera initialement vide.
(b) une méthode **chercher(self)** permettant de chercher une salle dans la table Salle par son nom et retourne son identifiant.
(c) une méthode **installer (self, p)** qui permet d'ajouter un poste **p** à une salle.
(d) une méthode **charger_liste_postes (self)** permettant de mettre à jour la liste_postes par les postes installés dans une salle à partir de la base de données.
(e) une méthode **__str__(self)** pour visualiser les postes (nom, catégorie) d'une salle.

On supposera pour la suite des questions que l'institut a équipé 3 salles (101, 102 et 323) par 9 postes, 3 postes (P1, P2 et P3) pour chacune des salles.
La base de données de l'institut se nomme **'institut.db'**, et on utilisera le module SQLite pour sa manipulation.

Questions :

1. Quelle est la requête SQL permettant de Créer la table **Installer**.

```sql
CREATE TABLE `installer` (
    `id_poste`  INTEGER NOT NULL,
    `id_logiciel`   INTEGER NOT NULL,
    `date_installation` DATE NOT NULL,
    `duree_licence` VARCHAR NOT NULL,
    PRIMARY KEY(id_poste, id_logiciel),
    FOREIGN KEY(`id_poste`) REFERENCES poste,
    FOREIGN KEY(`id_logiciel`) REFERENCES logiciel
);
```

2. Définir une fonction Python **executer** permettant de retourner le résultat d'exécution d'une requête SQL de type Select.

```python
# Q2
def executer(requete):
    import sqlite3
    connexion = sqlite3.connect('gerer_logiciels.db')
    cur = connexion.cursor()
    lt=cur.execute(requete).fetchall() #retourne une liste de tuples
    connexion.close()
    return lt
# Exemple
```

```
requete = " SELECT * FROM logiciel "
L = executer(requete)
```

3. Définir la classe **Logiciel**.

```
#Q3
class Logiciel :
    def __init__ (self, nom, version, prix):
        self.nom = nom
        self.version = version
        self.prix = prix

    def enregistrer (self):
        """
        Permet d'enregistrer le logiciel p dans la table logiciel
        """
        requete = "INSERT INTO logiciel VALUES
                   (null,'"+self.nom+"','"+self.version+"','"+self.prix+"');"
        print (requete)
        executer(requete)

    def chercherId(self):
        """
        Permet de chercher un logiciel dans la table Logiciel
        par son nom et sa version et retourne son identifiant.
        """
        requete = "SELECT id FROM logiciel WHERE nom = '"+ self.nom +
                  "' AND  version = '"+ self.version +"';"
        print (requete)
        lt=executer(requete)
        id=lt[0][0]
        return (id)

    def __str__(self):
        """
        visualiser un logiciel (nom, version, prix).
        """
        return "Logiciel (" + self.nom +", "+ self.version +", "+ self.prix+")"

L = Logiciel("Chrome","5.0","infini")
print(L)
```

4. Créer une fonction **lister_salles** permettant de retourner à partir de la table Salle, une liste contenant toutes les salles de l'institut.

```
#Q4
def lister_salles():
    requete ="SELECT * FROM salle ;"
    liste_de_tuples=executer(requete)
    liste_de_salles=[]
    for tup in liste_de_tuples:
        nouv_salle = Salle(tup[1])
        liste_de_salles.append(nouv_salle)
    return liste_de_salles
```

5. Définir un objet **scilab** représentant le logiciel *Scilab 2.3* en sa version gratuite et l'enregistrer dans la base de données.

```
#Q5
scilab = Logiciel("Scilab","2.3","infini")
scilab.enregistrer()
```

6. Enregistrer dans la base de données l'opération d'installation du Scilab sur tous les postes de l'institut le 04/04/2017.

anis_saied@hotmail.com

```
toutes_les_salles = liste_salles()
for s in toutes_les_salles:
    les_postes = s.charger_liste_postes()
    for p in les_postes():
        p.installer(scilab,'04/04/2017', 'infini')
```

7. Changer la protée de l'attribut prix de la classe Logiciel du publique à privée. Réécrire seulement la/les instructions de la/les méthodes modifiées de la classe Logiciel.

```
#méthode __init__
self.__prix = prix

#méthode enregistrer
requete = "INSERT INTO logiciel \
VALUES (null,'"+self.nom+"','"+self.version+"','"+self.__prix+"');"

#méthode  __str__
return "Logiciel (" + self.nom +", "+ self.version +", "+ self.__prix+")"
```

8. Ajouter, à la classe Logiciel, une méthode publique nommée **getPrix(self)** permettant de retourner le prix d'achat d'un logiciel. Quelles sont les modifications à faire au niveau de la classe Logiciel pour garder la cohérence du code (Réécrire seulement la/les instructions modifiées de la classe Logiciel).

```
def getPrix(self):
    return self.__prix
```

9. Ajouter, à la classe Logiciel, une méthode publique **setPrix(self, nouveau_prix)** permettant de modifier le prix d'un logiciel, dans laquelle on fait tous les traitements/contrôles nécessaires pour préserver l'intégrité des données. Réécrire seulement la/les instructions modifiées de la classe Logiciel.

```
def setPrix(self, nouveauPrix):
    assert type(nouveauPrix)==float and nouveauPrix > 0, raise ValueError
    self.__prix = nouveauPrix
```

10. Ajouter à la classe Poste une méthode **nb_logiciels(self)** permettant de calculer et retourner le nombre de logiciels installés sur un poste .

```
def nb_logiciels(self):
    """
    calculer et retourner le nombre de logiciels installés sur un poste
    """
    requete =
    "SELECTf COUNT (*) AS NB FROM INSTALLER WHERE id_poste = "+ self.id +";"
    lt=executer(requete)
    nb=lt[0][0] #Premier élément du premier tuple du résultat
    return (nb)
```

11. Quelle est la requête SQL permettant d'afficher les postes sur lesquels le logiciel 'Python' n'est pas installé.

```
SELECT *
FROM   poste
WHERE  poste.id NOT IN (
                         SELECT  installer.id_poste
                         FROM    installer , logiciel
                         WHERE   installer.id_logiciel = logiciel.id
                         AND     logiciel.nom ='Python'
                        );
```


anis_saied@hotmail.com