Exercice 2 (6 points)

Cet exercice porte sur les bases de données et la programmation orientée objet.

Dans cet exercice, on pourra utiliser les clauses du langage SQL pour :

- construire des requêtes d'interrogation à l'aide de `SELECT`, `FROM`, `WHERE` (avec les opérateurs logiques `AND`, `OR`), `JOIN ... ON` ;
- construire des requêtes d'insertion et de mise à jour à l'aide de `UPDATE`, `INSERT`, `DELETE` ;
- affiner les recherches à l'aide de `DISTINCT` et `ORDER BY`.

Susie décide de créer une base de données qui recense des randonnées allant d'un parking à un lac.

Elle crée trois relations représentées sur le schéma ci-dessous (figure 1).

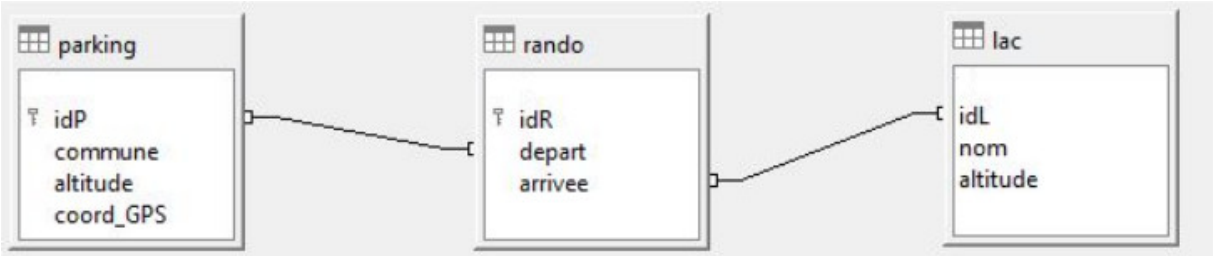

Figure 1. Schéma des trois relations

Les clés primaires sont signalées par une clé. Dans la relation `rando` :

- `depart` est une clé étrangère qui référence l'attribut `idP` de la relation `parking` ;
- `arrivee` est une clé étrangère qui référence l'attribut `idL` de la relation `lac`.

L'altitude, exprimée en mètre, est un entier.

Voici un extrait des enregistrements de ces trois relations.

parking			
idP	commune	altitude	coord_GPS
1	Chamonix	1 026	(45.98;6.89)
2	Argentiere	1 429	(45.99;6.92)
3	Passy	600	(45.92;6.72)
4	Passy	1 181	(45.95;6.71)
5	Nevache	2 022	(45.05;6.52)

rando		
idR	depart	arrivee
1	1	1
2	2	1
3	1	2
4	3	3

lac		
idL	nom	altitude
1	Lac Blanc	2 354
2	Lacs Noirs	2 564
3	Lac Vert	1 266
4	Lac Rouge	2 585

1. Indiquer ce que renvoie la requête suivante lorsqu'on l'applique aux extraits précédents.

```
SELECT nom
FROM lac
WHERE altitude <= 2000;
```

2. Indiquer les noms des lacs qu'on peut atteindre depuis le parking de Chamonix d'après la base de données de Susie.

À partir de maintenant, on travaille sur la totalité des enregistrements et non plus seulement sur les extraits précédents.

3. Donner une requête permettant d'obtenir les coordonnées GPS des parkings situés dans la commune de Passy à une altitude comprise strictement entre 800 et 1 000 mètres.

4. Donner une requête permettant d'obtenir les noms des lacs qu'il est possible d'atteindre depuis le parking situé à 1300 mètres d'altitude dans la commune de Cordon (on admet qu'un tel parking existe dans la base de données).

Dans les questions suivantes, l'ordre des requêtes SQL est important. On considère une nouvelle randonnée qui part du parking dont l'identifiant est 3 à Passy et qui conduit au lac d'Anterne situé à 2 059 mètres d'altitude.

Le parking est déjà dans la base de données mais par contre, ni la randonnée, ni le lac n'y figurent.

5. Donner les requêtes permettant à Susie d'ajouter à sa base de données cette randonnée et ce lac (on pourra utiliser l'identifiant 42 pour le lac et l'identifiant 100 pour la randonnée).

6. Susie a fait une erreur de saisie en insérant le nom du lac, elle a écrit 'Lc d Anterne'. Donner la requête permettant de corriger cette erreur.

7. Le parking dont l'identifiant est 28 a été transformé en un parc et n'existe plus.

   Donner les requêtes permettant de supprimer ce parking de la base de données.

Susie souhaite obtenir pour chacun des parkings le nombre de randonnées qui en partent.

Elle n'a pas encore appris à le faire en SQL et décide de le faire en Python. Pour cela elle définit la classe `Rando` ci-dessous permettant de représenter chacune des randonnées. La table `rando` est alors donnée par une liste d'objets de la classe `Rando`.

```
1 class Rando:
2     def __init__(self, idR, depart, arrivee):
3         self.idR = idR          # identifiant de la rando
4         self.depart = depart    # identifiant du parking
5         self.arrivee = arrivee  # identifiant du lac
```

8. Recopier et compléter les lignes 3 et 5 de la fonction `get_parking` qui prend en paramètre une liste de randonnées et qui renvoie la liste des identifiants des différents parkings, points de départ de ces randonnées (cette liste ne devra pas avoir de doublon).

   ```
   1 def get_parking(randos):
   2     parkings = []
   3     for ...:
   4         if rando.depart not in parkings:
   5             ...
   6     return parkings
   ```

   Par exemple, `get_parking([Rando(1, 1, 1), Rando(2, 2, 1), Rando(3, 1, 2)])` renvoie `[1, 2]`.

9. Recopier et compléter la ligne 4 de la fonction `get_nb_rando` qui prend en paramètres un identifiant de parking et une liste de randonnées, et qui renvoie le nombre de randonnées qui partent de ce parking.

   ```
   1 def get_nb_rando(parking, randos):
   2     nb = 0
   3     for rando in randos:
   4         if ...:
   ```

```
5             nb = nb + 1
6     return nb
```

Par exemple, `get_nb_rando(1, [Rando(1, 1, 1), Rando(2, 2, 1), Rando(3, 1, 2)])` renvoie 2.

10. Écrire une fonction `nb_rando_par_parking` qui prend en paramètre une liste de randonnées et qui renvoie un dictionnaire qui associe à chaque identifiant de parking le nombre de randonnées qui partent de ce parking.

    Par exemple, `nb_rando_par_parking([Rando(1, 1, 1), Rando(2, 2, 1), Rando(3, 1, 2)])` renvoie `{1:2, 2:1}`.