Présentation

Le site internet `http://ourairports.com/` recueille des informations sur les aéroports du monde. Ces informations sont disponibles sous la forme d'une base de donnée SQLite.

En l'état, la base de données est constituée de 4 relations :

- Airport_Frequencies recense les fréquences radio sur lesquelles les aéroports émettent ;
- Countries recense la liste des pays ;
- Airports recense la liste des aéroports ;
- Regions recense une liste des régions.

La relation Airports est constituée des attributs suivants :

- id : un identifiant
- type : un type (heliport, small_airport, seaplane_base ...)
- name : un nom
- des coordonnées géographiques(latitude_deg, latitude_deg, elevation_ft, à savoir les latitudes et longitudes en degrés ainsi que l'altitude en pieds) ;
- ...

Consultation de la base de données

Question 1 *Que permet la requête SQL suivante :*

▪ **SQL**
```
SELECT name FROM Countries;
```

Quelle est sa traduction en algèbre relationnelle ?

Question 2 *On souhaite sélectionner tous les noms de pays européens de la relation Countries. Exprimer la requête dans l'algèbre relationnelle puis en langage SQL. L'attribut des pays européens est désigné par "EU".*

Question 3 *On donne la requête suivante :*

$$\pi_{name}\left(\sigma_{iso\_country="FR"}(Regions)\right)$$

Que signifie-t-elle ? La traduire en langage SQL puis tester le résultat.

Consultation de la base de données en Python

Il est possible d'interroger la base de données en utilisant Python. Pour cela, on utilise le formalisme suivant :

▪ **Python**
```
import sqlite3 # Import des commandes permettant de manipuler la base de données
basesql = u"airports.db3" # Base de données initiale

cnx = sqlite3.connect(basesql)
curseur = cnx.cursor()

requete = "SELECT * FROM airports"
curseur.execute(requete)
```

Le préfixe u permet d'importer les fichiers encodés en UTF-8. Dans un certaine mesure, les caractères spéciaux sont alors pris en compte.

▪ **Python**

Le curseur est un objet contenant le résultat de la requête. Pour visualiser la première entité de la requête, la syntaxe est la suivante :

```
data = curseur.fetchone()
print(data)
```

Attention, à chaque utilisation de *fetchone()*, l'entité est supprimée du curseur. Pour parcourir chacune des entités, on peut utiliser la syntaxe suivante :

```
for cur in curseur:
    print(cur)
```

Attention, cette opération peut s'avérer maladroite si la requête a un grand nombre d'entités.

Question 4 *Que permettent les lignes de code suivantes :*

Python
```
requete = "SELECT * FROM airports"
curseur.execute(requete)
res = []
for cur in curseur:
    res.append(cur)
print(len(res))
```

Question 5 *En utilisant les possibilités de Python, donner le nombre de bases d'hydravion existantes. En utilisant la documentation ou le cours, comment utiliser la fonction d'agrégation COUNT pour obtenir un résultat équivalent ?*

Question 6 *Donner la liste des villes françaises (iso_country='FR') hébergeant de telles bases. Vous donnerez la requête SQL ainsi que son expression en algèbre relationnelle.*

Question 7 *Donner la liste des villes européennes (continent='EU') hébergeant de telles bases ainsi que leur nom et leur pays. Vous donnerez la requête SQL ainsi que son expression en algèbre relationnelle.*

Question 8 *En utilisant une jointure entre les relations Coutries et airports, donner la liste des bases d'hydravion américains (United States).*

Affichage des bases d'hydravion sur une carte

On souhaite afficher toutes les bases d'hydravion européennes sur une carte GoogleEarth.

Question 9 *A partir des résultats d'une requête, mettre dans un tableau les lignes constituées du nom de l'aéroport, de sa longitude, de sa latitude et de son type. Une ligne constituera une seule chaîne de caractère, chaque champ étant séparé par une virgule. La fin de la ligne sera terminée par un \n. De plus, les chaînes de caractères ne devront pas contenir le caractère &.*