DS Info 6 – Corrigé

D.Malka – MPSI 2016-2017 – Lycée Saint-Exupéry

19.06.2017

Exercice 1 – Données générales sur le monde

Requêtes fournissant :

1. le nom et le GNP de tous les pays dont le GNP est supérieur à 100 000 ;

```
SELECT co.name, co.GNP
FROM Country co
WHERE
  co.GNP>100 000;
```

2. le nom et la population de toutes les villes françaises référencées dans la base de données ;

```
SELECT ci.name,ci.Population
FROM City ci
WHERE
  Country_code=FRA;
```

   ou si on ne connaît pas le code de la France :

```
SELECT ci.name,ci.Population
FROM City ci
  JOIN Country co ON ci.CountryCode=co.Code
WHERE
  co.Name="France";
```

3. le nombre de personnes parlant anglais dans le monde :

```
SELECT SUM(co_l.percentage*co.Population)
FROM CountryLanguage co_l
  JOIN Country co ON co_l.CountryCode=co.Code
WHERE
  co_l.language="English";
```

4. la population totale de chaque continent :

```
SELECT co.Continent, SUM(co.Population) as Population
FROM Country co
GROUP BY Continent;
```

5. le nom et la superficie des pays dont la population moyenne dans les grandes villes est supérieure à 100 000 :

```
SELECT co.Name, co.SurfaceArea
FROM Country co
  JOIN City ci ON ci.CountryCode=co.Code
GROUP BY co.Code
HAVING AVG(ci.Population)>100000;
```

Code	Name	Continent	SurfaceArea	Population	LifeExpectancy	GNP	Capital
DOM	Dominican Republic	North America	48511.00	8495000	73.2	15846.00	587
DZA	Algeria	Africa	2381741.00	31471000	69.7	49982.00	35
ECU	Ecuador	South America	283561.00	12646000	71.1	19770.00	594
EGY	Egypt	Africa	1001449.00	68470000	63.3	82710.00	608
ERI	Eritrea	Africa	117600.00	3850000	55.8	650.00	652
ESH	Western Sahara	Africa	266000.00	293000	49.8	60.00	2453
ESP	Spain	Europe	505992.00	39441700	78.8	553233.00	653
EST	Estonia	Europe	45227.00	1439200	69.5	5328.00	3791
ETH	Ethiopia	Africa	1104300.00	62565000	45.2	6353.00	756
FIN	Finland	Europe	338145.00	5171300	77.4	121914.00	3236
FJI	Fiji Islands	Oceania	18274.00	817000	67.9	1536.00	764
FLK	Falkland Islands	South America	12173.00	2000	*NULL*	0.00	763
FRA	France	Europe	551500.00	59225700	78.8	1424285.00	2974
FRO	Faroe Islands	Europe	1399.00	43000	78.4	0.00	901
FSM	Micronesia, Federated States of	Oceania	702.00	119000	68.6	212.00	2689
GAB	Gabon	Africa	267668.00	1226000	50.1	5493.00	902
GBR	United Kingdom	Europe	242900.00	59623400	77.7	1378330.00	456
GEO	Georgia	Asia	69700.00	4968000	64.5	6064.00	905
GHA	Ghana	Africa	238533.00	20212000	57.4	7137.00	910
GIB	Gibraltar	Europe	6.00	25000	79.0	258.00	915

(a) Table `Country`

CountryCode	Language	IsOfficial	Percentage
ABW	Dutch	T	5.3
ABW	English	F	9.5
ABW	Papiamento	F	76.7
ABW	Spanish	F	7.4
AFG	Balochi	F	0.9
AFG	Dari	T	32.1
AFG	Pashto	T	52.4
AFG	Turkmenian	F	1.9
AFG	Uzbek	F	8.8
AGO	Ambo	F	2.4
AGO	Chokwe	F	4.2
AGO	Kongo	F	13.2
AGO	Luchazi	F	2.4
AGO	Luimbe-nganguela	F	5.4
AGO	Luvale	F	3.6
AGO	Mbundu	F	21.6
AGO	Nyaneka-nkhumbi	F	5.4
AGO	Ovimbundu	F	37.2
AIA	English	T	0.0
ALB	Albaniana	T	97.9

(b) Table `CountryLanguage`

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabol	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilburg	NLD	Noord-Brabant	193238
11	Groningen	NLD	Groningen	172701
12	Breda	NLD	Noord-Brabant	160398
13	Apeldoorn	NLD	Gelderland	153491
14	Nijmegen	NLD	Gelderland	152463
15	Enschede	NLD	Overijssel	149544
16	Haarlem	NLD	Noord-Holland	148772
17	Almere	NLD	Flevoland	142465
18	Arnhem	NLD	Gelderland	138020
19	Zaanstad	NLD	Noord-Holland	135621
20	's-Hertogenbosch	NLD	Noord-Brabant	129170

(c) Table `City`

FIGURE 1 – Extraits de la base de données `World`

Exercice 2 – Gestion du trafic aérien

1. PLAN DE VOL

1.1 Requête SQL qui fournit le nombre de vols qui doivent décoller dans la journée du 2 mai 2016 avant midi.

```
SELECT COUNT(*)
FROM vol v
WHERE
  v.jour = '2016-05-02' AND
  v.heure<'12:00';
```

1.2 Requête SQL qui fournit la liste des numéros de vols au départ d'un aéroport desservant Paris le 2 mai 2016.

```
SELECT v.id_vol
FROM vol v
  JOIN aeroport a ON v.arrivee=a.id_aero
WHERE
  v.arrivee='Paris'
  v.jour = '2016-05-02';
```

1.3 La requête

```
SELECT id_vol
FROM vol
  JOIN aeroport AS d ON d.id_aero = depart
  JOIN aeroport AS a ON a.id_aero = arrivee
WHERE
  d.pays = 'France' AND
  a.pays = 'France' AND
  jour = '2016-05-02';
```

renvoie la liste des vols intérieurs à la France le 2 mai 2016.

1.4 Requête SQL qui fournit la liste des couples (Id1 , Id2) des identifiants des vols dans cette situation.

L'idée est de réaliser une jointure de la table `vol` avec elle même afin de pouvoir comparer le départ et l'arrivée des différents vols. On ne conserve que les n-uplets correspondant à des risques de conflit. La condition `v1.id_vol>v2.id_vol` sert à éliminer les doublons, (Id_1, Id_2) et (Id_2, Id_1) représentant le même conflit.

```
SELECT v1.id_vol, v2.id_vol
FROM vol v1
  JOIN vol v2 ON v1.depart=v2.arrivee AND
           v1.arrivee=v2.depart AND
           v1.niveau=v2.niveau AND
           v1.jour=v2.jour AND
WHERE
  v1.id_vol>v2.id_vol;
```

2. ALLOCATION DES NIVEAUX DE VOL

2.1 Implantation du problème

2.1.1 Fonction `nb_conflits()` sans paramètre qui renvoie le nombre de conflits potentiels, c'est-à-dire le nombre d'arêtes de valuation non nulle du graphe.

```
def nb_conflits():
    nb=0
    n=len(conflit)
    for i in range(1,n):#on demarre a 1 car pas de conflit entre un vol et lui meme
        for j in range(i+1,n):#par symetrie on ne parcourt que le triangle superieur
              de la matrice
            if conflit[i][j]!=0:
                nb=nb+1
    return nb
```

2.1.2 Complexité : parcours de l'ordre d'environ $\frac{1}{2}.(3n)^2$ arêtes du graphes. On prenant l'addition comme coût élémentaire, on trouve une complexité quadratique : $O(n^2)$.

2.2 Régulation

2.2.1 Fonction `nb_vol_par_niveau_relatif(regulation)` qui prend en paramètre une régulation (liste de n entiers) et qui renvoie une liste de 3 entiers $[a, b, c]$ dans laquelle a est le nombre de vols à leurs niveaux RFL, b le nombre de vols au-dessus de leurs niveaux RFL et c le nombre de vols au-dessous de leurs niveaux RFL.

```
1 def nb_vol_par_niveau_relatif(regulation):
2     nb_niveau=[0,0,0]
3     for r in regulation:
4         nb_niveau[r]=nb_niveau[r]+1
5     return nb_niveau
```

2.2.2 Coût d'une régulation

2.2.2.1 Fonction `cout_regulation(regulation)` qui prend en paramètre une liste représentant une régulation et qui renvoie le coût de celle-ci.

L'idée est la suivante : les coût des conflits potentiels entre les vols k et p sont contenus dans la sous matrice $3x3$ de sommet supérieur gauche $(k = 3i, p = 3j)$ représentée fig.2.

```
conflit = [ [   0,   0,   0, 100, 100,   0,   0, 150,   0],
            [   0,   0,   0,   0,   0,  50,   0,   0,   0],
            [   0,   0,   0,   0, 200,   0,   0, 300,  50],
            [ 100,   0,   0,   0,   0,   0, 400,   0,   0],
            [ 100,   0, 200,   0,   0,   0, 200,   0, 100],
            [   0,  50,   0,   0,   0,   0,   0,   0,   0],
            [   0,   0,   0, 400, 200,   0,   0,   0,   0],
            [ 150,   0, 300,   0,   0,   0,   0,   0,   0],
            [   0,   0,  50,   0, 100,   0,   0,   0,   0] ]
```

FIGURE 2 – Sous-matrice du conflit potentiels entre les vols k et p.

Il faut donc, dans chacune de ces sous-matrice extraire le coût correspondant à la régulation r. Dans les coordonnées relatives à chaque sous-matrice de ce coût sont $(r[k], r[p])$. En coordonnées absolues : $(k + r[k], p + r[p])$

```
1  def cout_regulation(regulation):
2      c=0
3      n=len(conflit)
4      #On se deplace de sous-matrices 3x3 en sous-matrices 3x3 contenant les 9
           couts potentielss du conflit entre les vols k et p
5      for k in range(1,n,3):
6          r_k=regulation[k]
7          for p in range(p+3,n):
8              r_p=regulation[p]
9              #Le cout du conflit (k,p) resultant de la regulation r a pour
                   coordonnees (k+r[k],p+r[p])
10             c=c+conflit[k+r_k][p+r_p]
11     return c
```

2.2.2.2 Complexité : $O(n^2)$.

2.2.2.3 Fonction `cout_RFL()` qui renvoie le coût de la régulation pour laquelle chaque vol est à son RFL.

```
1 def cout_RFL():
2     n=len(conflit)//3
3     RFL=[0]*n
4     return cout_regulation(RFL)
```

2.2.3 Il existe 3^n régulations de vols. L'évaluation du coût d'une régulation se fait en $O(n^2)$. Une fonction qui rechercherait la régulation de coût minimal par évaluation de toutes les régulations possibles aurait une complexité $O(3^n.n^2)$ absolument rédhibitoire !

Opérations et fonctions Python disponibles

Fonctions

— `exp(x)` calcule l'exponentielle du nombre x
— `randint(n)` (n entier) renvoie un entier aléatoire compris entre 0 et n-1 inclus
— `random()` renvoie un nombre flottant tiré aléatoirement dans [0, 1[suivant une distribution uniforme
— `sqrt(x)` calcule la racine carrée du nombre x
— `time()` renvoie l'heure sous la forme d'un nombre de secondes depuis un instant de référence

Opérations sur les listes

— `u + v` construit une liste constituée de la concaténation des listes u et v : `[1, 2] + [3, 4, 5]` → `[1, 2, 3, 4, 5]`
— `n * u` construit une liste constitué de la liste u concaténée n fois avec elle-même : `3 * [1, 2]` → `[1, 2, 1, 2, 1, 2]`
— `u.append(e)` ajoute l'élément e à la fin de la liste u
— `del(u[i])` supprime de la liste u son élément d'indice i
— `u.insert(i, e)` insère l'élément e à la position d'indice i dans la liste u (en décalant les éléments suivants) ; si `i >= len(u)`, e est ajouté en fin de liste
— `len(u)` donne le nombre d'éléments de la liste u :
— `u[i], u[j] = u[j], u[i]` permute les éléments d'indice i et j dans la liste u