

I. Interroger une base de données existante (sur ordinateur)

Exercice 1 *Exploiter une base existante (sur machine) : AOP et AOC*

Contexte : Un fichier à copier dans votre répertoire personnel est disponible dans l'espace d'échange des PC :

Enregistrez une copie du fichier **AOP.sqlite** sous un nouveau nom **mesAOP.sqlite** dans votre répertoire personnel.

N'éditez pas le fichier source ! ! ! !

Schéma relationnel retenu :

Les clés primaires des tables suivantes sont soulignées.

CI (Code Indication : département suivi du numéro dans le département)

Département (Département)

Commune (Commune)

Art (article L')

Airegéographique (Dénomination de l'AOP ou AOC)

IDA (Identifiant INSEE)

I) Requêtes

Ouvrez cette base de données avec un éditeur SQL.

1. Déterminez les entrées complètes de toutes les AOC de Quimper.
2. Déterminez toutes les communes d'AOC du Finistère.
3. Déterminez toutes les communes correspondant à l'IDA 1360. De quoi s'agit-il ?
4. Combien de communes du Finistère possèdent une AOP ?

II. Annales d'écrits de concours (sur papier)

Exercice 2 Mines MP-PSI-PC 2017

Base de données

On modélise ici un réseau routier par un ensemble de croisements et de voies reliant ces croisements. Les voies partent d'un croisement et arrivent à un autre croisement. Ainsi, pour modéliser une route à double sens, on utilise deux voies circulant en sens opposés. La base de données du réseau routier est constituée des relations suivantes :

- Croisement(id, longitude, latitude)
- Voie(id, longueur, id_croisement_debut, id_croisement_fin)

Dans la suite on considère c l'identifiant (id) d'un croisement donné.

1. Écrire la requête SQL qui renvoie les identifiants des croisements atteignables en utilisant une seule voie à partir du croisement ayant l'identifiant c .
2. Écrire la requête SQL qui renvoie les longitudes et latitudes des croisements atteignables en utilisant une seule voie, à partir du croisement c .
3. Que renvoie la requête SQL suivante ?

```
SELECT V2.id_croisement_fin
FROM Voie as V1
JOIN Voie as V2
ON V1.id_croisement_fin = V2.id_croisement_debut
WHERE V1.id_croisement_debut = c
```

Exercice 3 Mines MP-PSI-PC 2016

Partie I : tri

Dans le but ultérieur de réaliser des études statistiques, on souhaite se doter d'une fonction tri. On se donne la fonction `tri` suivante, écrite en Python :

```
def tri(L):
    n=len(L)
    for i in range(1,n):
        j=i
        x=L[i]
        while 0<j and x<L[j-1]:
            L[j]=L[j-1]
            j=j-1
        L[j]=x
    return(L)
```

1. Lors de l'appel `tri(L)` lorsque `L` est la liste `[5, 2, 3, 1, 4]`, donner le contenu de la liste `L` à la fin de chaque itération de la boucle `for`.
2. Soit `L` une liste non vide d'entiers ou de flottants. Montrer que « la liste `L[0: i+1]` (avec la convention Python) est triée par ordre croissant à l'issue de l'itération `i` » est un invariant de boucle. En déduire que `tri(L)` trie la liste `L`.
3. Évaluer la complexité dans le meilleur des cas et dans le pire des cas de l'appel `tri(L)` en fonction du nombre d'éléments de `L`. Citer un algorithme de tri plus efficace dans le pire des cas. Quelle est la complexité dans le meilleur et dans le pire des cas ?

On souhaite, partant d'une liste constituée de couples (chaîne,entier), trier la liste par ordre croissant de l'entier associé suivant le fonctionnement suivant :

```
>>> L=[[ ' Bresil ' ,76],[ ' Kenya ' ,26017],[ ' Ouganda ' ,8431]]
>>> tri_chaine(L)
>>> L
[[ ' Bresil ' ,76],[ ' Ouganda ' ,8431],[ ' Kenya ' ,26017]]
```

4. Écrire en Python une fonction `tri_chaine` réalisant cette opération.

Pour suivre la propagation des épidémies, de nombreuses données sont recueillies par les institutions internationales comme l'OMS. Par exemple, pour le paludisme, on dispose de deux tables :

— La table `palu` recense le nombre de nouveaux cas confirmés et le nombre de décès liés au paludisme ; certaines lignes de cette table sont données en exemple (on précise que `iso` est un identifiant unique pour chaque pays) :

nom	iso	annee	cas	deces
Bresil	BR	2009	309 316	85
Bresil	BR	2010	334 667	76
Kenya	KE	2010	898 531	26 017
Mali	ML	2011	307 035	2 128
Ouganda	UG	2010	1 581 160	8 431
...				

— la table `demographie` recense la population totale de chaque pays ; certaines lignes de cette table sont données en exemple :

pays	periode	pop
BR	2009	193 020 000
BR	2010	194 946 000
KE	2010	40 909 000
ML	2011	14 417 000
UG	2010	33 987 000
...		

- Au vu des données présentées dans la table `pa.lu`, parmi les attributs `nom`, `iso` et `annee`, quels attributs peuvent servir de clé primaire ? Un couple d'attributs pourrait-il servir de clé primaire ? (on considère qu'une clé primaire peut posséder plusieurs attributs). Si oui, en préciser un.
- Écrire une requête en langage SQL qui récupère depuis la table `pa.lu` toutes les données de l'année 2010 qui correspondent à des pays où le nombre de décès dus au paludisme est supérieur ou égal à 1 000. On appelle *taux d'incidence d'une épidémie* le rapport du nombre de nouveaux cas pendant une période donnée sur la taille de la population-cible pendant la même période. Il s'exprime généralement en « nombre de nouveaux cas pour 100 000 personnes par année ». Il s'agit d'un des critères les plus importants pour évaluer la fréquence et la vitesse d'apparition d'une épidémie.
- Écrire une requête en langage SQL qui détermine le taux d'incidence du paludisme en 2011 pour les différents pays de la table `pa.lu`.
- Écrire une requête en langage SQL permettant de déterminer le nom du pays ayant eu le deuxième plus grand nombre de nouveaux cas de paludisme en 2010 (on pourra supposer qu'il n'y a pas de pays *ex æquo* pour les nombres de cas). On considère la requête R qui s'écrit dans le langage de l'algèbre relationnelle :

$$R = \pi_{\text{nom, deces}}(\sigma_{\text{annee}=2010}(\text{pa.lu}))$$

On suppose que le résultat de cette requête a été converti en une liste Python stockée dans la variable `deces2010` et constituée de couples (chaîne,entier).

- Quelle instruction peut-on écrire en Python pour trier la liste `deces2010` par ordre croissant du nombre de décès dus au paludisme en 2010 ?

Exercice 4 *Exploitation d'une base de données, info communes Centrale 2015*

À partir de mesures régulièrement effectuées par différents observatoires, une base de données des caractéristiques et des états des corps célestes de notre Système solaire est maintenue à jour. L'objectif de cette partie est d'extraire de cette base de données les informations nécessaires à la mise en oeuvre des fonctions développées dans les parties précédentes, puis de les utiliser pour prévoir les positions futures des différentes planètes. Les données à extraire sont les masses des corps étudiés et leurs états (position et vitesse) à l'instant t_{min} du début de la simulation. Une version simplifiée, réduite à deux tables, de la base de données du Système solaire est donnée figure 3. Les masses sont exprimées en kilogrammes, les distances en unités astronomiques ($1 \text{ au} = 1,5 \times 10^{11} \text{ m}$) et les vitesses en kilomètres par seconde. Le référentiel utilisé pour exprimer les composantes des positions et des vitesses est galiléen, orthonormé et son centre est situé à proximité du Soleil.

CORPS			ETAT							
id_corps	nom	masse	id_corps	datem	x	y	z	vx	vy	vz
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

La table CORPS répertorie les corps étudiés, elle contient les colonnes

- id_corps (clé primaire) entier identifiant chaque corps ;
- nom, chaîne de caractères, désigne le nom usuel du corps ;
- masse de type flottant, contient la masse du corps.

La table ETAT rassemble l'historique des états successifs (positions et vitesses) des corps étudiés. Elle est constituée de huit colonnes :

- id_corps de type entier, identifie le corps concerné ;
- datem est la date de la mesure, sous forme d'un entier donnant le nombre de secondes écoulées depuis un instant d'origine ;
- trois colonnes de type flottant pour les composantes de la position x, y, z ;
- trois colonnes de type flottant pour les composantes de la vitesse vx, vy, vz.

A- Écrire une requête SQL qui renvoie la liste des masses de tous les corps étudiés.

B- Les états des différents corps ne sont pas forcément tous déterminés exactement au même instant. Nous allons assimiler l'état initial (à la date t_{min}) de chaque corps à son dernier état connu antérieur à t_{min} . Dans toute la suite, on supposera que la valeur de t_{min} , sous le format utilisé dans la table ETAT, est accessible à toute requête SQL via l'expression `tmin()`.

B.1- On souhaite d'abord vérifier que tous les corps étudiés disposent d'un état connu antérieur à `tmin()`. Le nombre de corps présents dans la base est obtenu grâce à la requête `SELECT count(*) FROM corps`. Écrire une requête SQL qui renvoie le nombre de corps qui ont au moins un état connu antérieur à `tmin()`.

B.2- Écrire une requête SQL qui renvoie, pour chaque corps, son identifiant et la date de son dernier état antérieur à `tmin()`.

B.3- Le résultat de la requête précédente est stocké dans une nouvelle table `date_mesure` à deux colonnes :
 — id_corps de type entier, contient l'identifiant du corps considéré ;
 — date_der de type entier, correspond à la date du dernier état connu du corps considéré, antérieur à `tmin()`.

Pour simplifier la simulation, on décide de négliger l'influence des corps ayant une masse strictement inférieure à une valeur fixée `masse_min()` et de ne s'intéresser qu'aux corps situés dans un cube, centré sur l'origine du référentiel de référence et d'arête `arete()` donnée. Les faces de ce cube sont parallèles aux plans formés par les axes du référentiel de référence. Écrire une requête SQL qui renvoie la masse et l'état initial (sous la forme `masse, x, y, z, vx, vy, vz`) de chaque corps retenu pour participer à la simulation. Classez les corps dans l'ordre croissant par rapport à leur distance à l'origine du référentiel.