Math93.com

BAC NSI - Correction Métropole 2023 jour 1 - NSI

Thèmes des exercices (sur 12 points)

- Exercice 1 (3 points) : Base de données relationnelle et le langage SQL.
- Exercice 2 (3 points) : Les réseaux.
- Exercice 3 (6 points) : programmation orientée objet en Python et d'algorithmique.
- Lien vers le sujet

Exercice 1. Base de données relationnelle et le langage SQL. 3points

On pourra utiliser les mots-clés SQL suivants :

```
AND, FROM, INSERT, INTO, JOIN, ON, SELECT, SET, UPDATE, VALUES, WHERE
```

Un grand magasin de meubles propose à ses clients un large choix de meubles. Les informations correspondantes sont rangées dans une base de données composée de trois relations. Voici le schéma de deux de ces relations :

- *Clients (id, nom, prenom, adresse, ville)*
- *Commandes (id, #idClient, #idMeuble, quantite, date)*

Dans ce schéma :

- la clé primaire de chaque relation est définie par les attributs soulignés ;
- les attributs précédés de # sont les clés étrangères.

La troisième relation est appelée Meubles et concerne les meubles du magasin. Le tableau de la figure 1 ci-dessous en présente un extrait :

id	intitule	prix	stock	description
62	'skap'	69.99	2	'Armoire blanche 3 portes'
63	'skap'	69.99	3	'Armoire noire 3 portes'
74	'stol'	39.99	10	'Chaise en bois avec tissu bleu'
98	'hylla'	99.99	0	'Bibliothèque 5 étages blanche'

Figure 1 – Extrait de la relation `Meubles`

1. Dans cette question, on s'intéresse au modèle relationnel.
   (a) Donner la caractéristique qu'un attribut doit avoir pour être choisi comme clé primaire.

Corrigé

Pour être choisi comme clé primaire, un attribut doit avoir les caractéristiques suivantes :

- Unicité : chaque valeur d'un attribut clé primaire doit être unique pour chaque enregistrement dans la table. Cela permet d'identifier de manière unique chaque enregistrement dans la table.
- On peut ajouter Non-nullité et immutabilité mais ce n'était pas demandé.

(b) Expliquer le rôle des deux clés étrangères de la relation Commandes.

Corrigé

- Chaque commande possède 5 attributs.
  - **id** : la clé primaire de la table **Meubles**
  - **#idClient** : une clé étrangère (un attribut) qui pointe vers la clé primaire id de la table **Clients**. Elle permet de connaître le client qui a passé la commande. -
  - **#idMeuble** : clé étrangère, un attribut qui pointe vers la clé primaire id de la table **Meubles** qui permet de connaître le meuble commandé.
  - **quantité** : qui sera le nombre commandé.
  - **date** : la date de cette commande

(c) Donner le schéma relationnel de la relation Meubles en précisant la clé primaire et les éventuelles clés étrangères.

Corrigé

id	intitule	prix	stock	description
62	'skap'	69.99	2	'Armoire blanche 3 portes'
63	'skap'	69.99	3	'Armoire noire 3 portes'
74	'stol'	39.99	10	'Chaise en bois avec tissu bleu'
98	'hylla'	99.99	0	'Bibliothèque 5 étages blanche'

Figure 1 – Extrait de la relation `Meubles`

Le schéma relationnel de la relation Meubles est :

- *Meubles (id : INT, intitule : VARCHAR, prix : FLOAT, stock : INT, description : VARCHAR)*

Il n'y a pas de clé étrangère.

2. En vous basant uniquement sur les données du tableau de la figure 1, donner le résultat de la requête suivante :

```
SELECT id, stock, description
FROM   Meubles
WHERE  intitule = 'skap';
```

Corrigé

En se basant uniquement sur les données du tableau de la figure 1, le résultat de la requête ci-dessus est :

- 62, 2, 'Armoire blanche 3 portes'
- 63, 3, 'Armoire noire 3 portes'

3. Donner la requête SQL permettant d'afficher les noms et prénoms des clients habitant à Paris.

Corrigé

Voici le schéma de la table *Clients* est : :

- *Clients (id, nom, prenom, adresse, ville)*

Donc la requête SQL permettant d'afficher les noms et prénoms des clients habitant à Paris est :

```
SELECT nom, prenom
FROM   Clients
WHERE  ville = 'Paris';
```

4. Le magasin vient de recevoir des meubles dont l'intitulé est 'hylla' et dont l'attribut id dans la relation Meubles vaut 98. Le stock de ces meubles est alors de 50. Recopier et compléter la requête SQL ci-dessous qui permet de mettre à jour la base de données.

Corrigé

id	intitule	prix	stock	description
62	'skap'	69.99	2	'Armoire blanche 3 portes'
63	'skap'	69.99	3	'Armoire noire 3 portes'
74	'stol'	39.99	10	'Chaise en bois avec tissu bleu'
98	'hylla'	99.99	0	'Bibliothèque 5 étages blanche'

Figure 1 – Extrait de la relation `Meubles`

On doit donc UPDATE l'id 98 de la table *Meubles* par la requête SQL ci-dessous :

```
UPDATE Meubles
SET    stock = 50
WHERE  id = 98 ;
```

5. Le magasin vient d'ajouter à son catalogue un nouveau meuble dont les caractéristiques sont les suivantes :

id	intitule	prix	stock	description
65	'matta'	95.99	25	'Tapis vert à pois rouges'

Donner la requête SQL qui permet d'ajouter cet article à la relation Meubles.

```
INSERT INTO Meubles
VALUES      (65,'matta',95.99,25,'Tapis vert à pois rouges')
```

6. Donner la requête SQL permettant de récupérer le nom et le prénom des différents clients qui ont passé une commande le 30 avril 2021. On précise que, dans la relation Commandes, les dates sont des chaînes de caractères, par exemple '21/08/2002'.

Corrigé

Voici le schéma des tables *Commandes* et *Clients* :

- *Clients (id, nom, prenom, adresse, ville)*
- *Commandes (id, #idClient, #idMeuble, quantite, date)*

Donc la requête SQL demandées est :

```
SELECT nom, prenom
FROM   Clients JOIN Commandes
WHERE  Commandes.idClient = Clients.id
AND    date = '30/04/2001'
```

Exercice 2. Les réseaux. 3points

Le réseau d'une entreprise dispose de quatre sites (SiteA, SiteB, SiteC et SiteD) et de cinq routeurs (R1, R2, R3, R4 et R5). La figure 1 en donne une représentation.

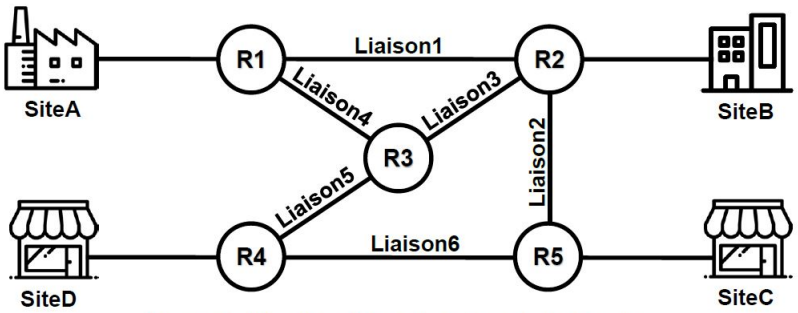

Figure 1 – Représentation du réseau de l'entreprise

1. Justifier brièvement que ce réseau sera encore fonctionnel même si l'une des liaisons (Liaison1, Liaison2,..., Liaison6) est coupée.

Corrigé

Si une seule des liaisons est coupée, aucun routeur ni aucun site ne sera isolé.

2. Dans un premier temps, les tables de routage des routeurs sont configurées manuellement (voir figure 2).

Routeur R1	
Destination	**Suivant**
SiteA	Local
SiteB	R2
SiteC	R3
SiteD	R3

Routeur R2	
Destination	**Suivant**
SiteA	R1
SiteB	Local
SiteC	R3
SiteD	R3

Routeur R3	
Destination	**Suivant**
SiteA	R1
SiteB	R2
SiteC	R4
SiteD	R4

Routeur R4	
Destination	**Suivant**
SiteA	R3
SiteB	R3
SiteC	R5
SiteD	Local

Routeur R5	
Destination	**Suivant**
SiteA	R4
SiteB	R4
SiteC	Local
SiteD	R4

Figure 2 – Tables de routage des routeurs R1, R2, R3, R4 et R5

Indiquer le chemin suivi par les paquets lorsqu'une information est envoyée de SiteB à SiteC.

Corrigé

Le chemin serait SiteB - R3 - R4 - R5 - SiteC.

3. Afin d'optimiser la maintenance du réseau, les tables de routage sont configurées automatiquement en utilisant le protocole RIP.

   Pour le protocole RIP, le chemin est construit de façon à minimiser le nombre de routeurs traversés. Recopier et compléter la table de routage RIP du routeur R1 (voir figure 3).

Routeur R1 (RIP)		
Destination	**Suivant**	**Nombre de sauts**
SiteA	Local	0
SiteB		
SiteC		
SiteD		

Figure 3 – Table de routage RIP du routeur R1

Corrigé

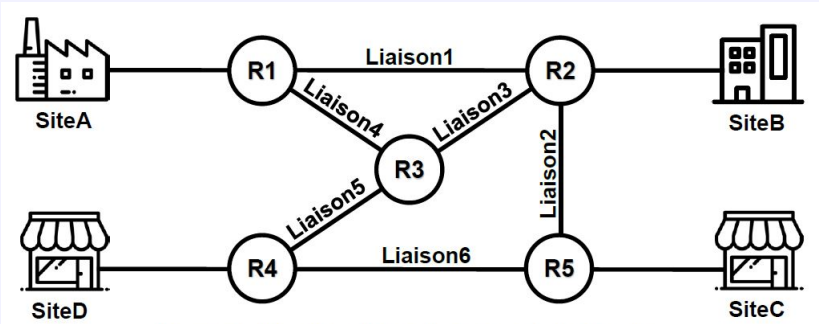

Figure 1 – Représentation du réseau de l'entreprise

Protocole RIP (*Routing Information Protocol*)

Routeur R1 (RIP)		
Destination	**Suivant**	**Nombre de sauts**
SiteA	Local	0
SiteB	R2	1
SiteC	R2	2
SiteD	R3	2

4. La liaison Liaison2 a un débit très inférieur aux autres liaisons. Expliquer pourquoi le choix du protocole RIP n'est pas judicieux.

Corrigé

Le protocole RIP ne prend pas en compte le débit de chaque liaison mais uniquement le nombre de saut minimum entre deux routeurs.

Donc même si un chemin a plus de sauts mais est plus rapide qu'un chemin ayant moins de saut, le protocole RIP ne lempruntera pas.

Il peut être judicieux dutiliser le protocole OSPF (*Open Shortest Path First*).

5. On considère maintenant que les tables de routage sont configurées en utilisant le protocole OSPF. Pour le protocole OSPF, le chemin est construit de façon à minimiser le coût. Le coût d'un chemin est la somme des coûts des liaisons à parcourir. Pour une liaison, la relation entre le coût (sans unité) et le débit D (en bit/s) est donnée par :

$$\text{coût} = \frac{10^{10}}{D}$$

Par convention, le coût d'une liaison directe entre un routeur et un site est 0. On donne le coût des liaisons dans la figure 4.

Liaison	Coût
Liaison1	100 000
Liaison2	1 000 000
Liaison3	5
Liaison4	50 000
Liaison5	5
Liaison6	10

Figure 4 – Table des coûts des liaisons

(a) Indiquer la liaison dont le débit est le plus faible.

Corrigé

La liaison dont le débit est le plus faible est celle dont le coût est le plus grand.

C'est donc la **Liaison2** qui a le débit le plus faible.

(b) Donner la liste des quatre chemins possibles pour aller de SiteA à SiteC sans utiliser deux fois le même routeur et calculer le coût de chacun de ces chemins.

Corrigé

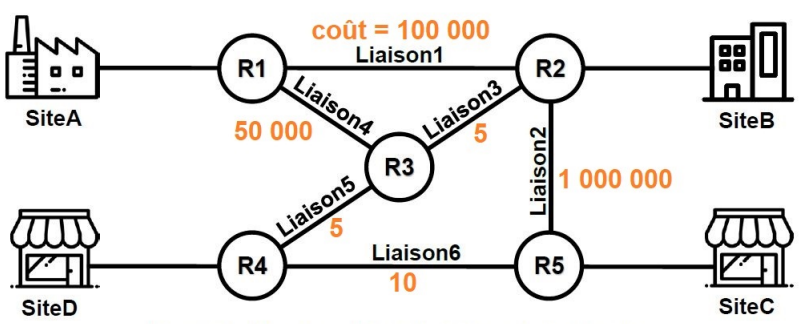

Figure 1 – Représentation du réseau de l'entreprise

Les quatres chemins possibles pour aller de SiteA à SiteC sont :

- SiteA - R1 - R2 - R5 - SiteC coût = 1 100 000
- SiteA - R1 - R3 - R4 - R5 - SiteC coût = 50 015
- SiteA - R1 - R3 - R2 - R5 - SiteC coût = 1 050 005
- SiteA - R1 - R2 - R3 - R4 - R5 - SiteC coût = 100 020

(c) Recopier et compléter la table de routage OSPF du routeur R1 (voir figure 5).

Routeur R1 (OSPF)		
Destination	**Suivant**	**Coût total du chemin**
SiteA	Local	0
SiteB		
SiteC		
SiteD		

Figure 5 – Table de routage OSPF du routeur R1

Corrigé

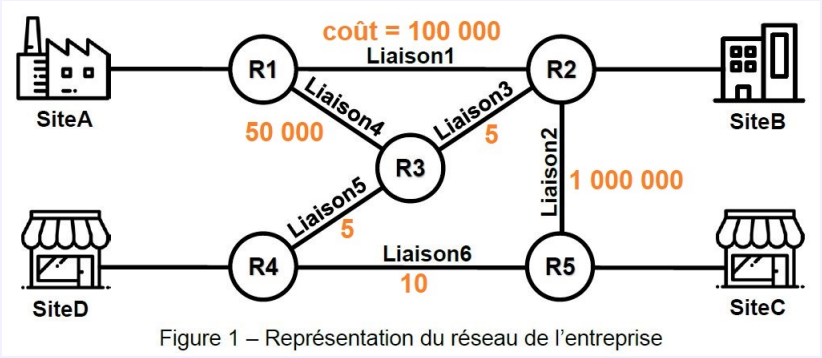

Figure 1 – Représentation du réseau de l'entreprise

Routeur R1 (OSPF)		
Destination	**Suivant**	**Coût total du chemin**
SiteA	Local	0
SiteB	R3	50 005
SiteC	R3	50 0015
SiteD	R3	50 005

Exercice 3. programmation orientée objet en Python et d'algorithmique. 6points

Un pays est composé de différentes régions. Deux régions sont voisines si elles ont au moins une frontière en commun. L'objectif est d'attribuer une couleur à chaque région sur la carte du pays sans que deux régions voisines aient la même couleur et en utilisant le moins de couleurs possibles. La figure 1 ci-dessous donne un exemple de résultat de coloration des régions de la France métropolitaine.

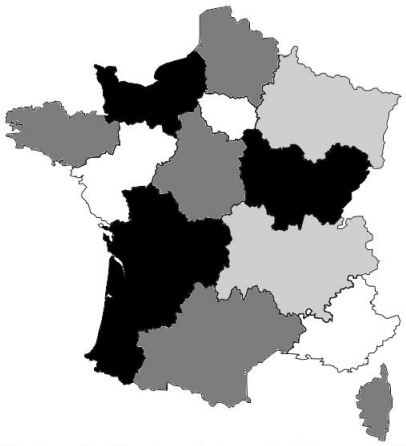

Figure 1 – Carte coloriée des régions de France métropolitaine

Partie 1

On considère la classe *Region* qui modélise une région sur une carte et dont le début de l'implémentation est :

```
class Region:
    '''Modélise une région d'un pays sur une carte.'''
    def __init__(self, nom_region): # methode 1 : constructeur
        '''
        initialise une région : param nom_region (str) le nom
        de la région
        '''
        self.nom = nom_region # 1er attribut

        # tableau des régions voisines, vide au départ
        self.tab_voisines = [] # 2e attribu

        # tableau des couleurs disponibles pour colorier la région
        self.tab_couleurs_disponibles = ['rouge', 'vert', 'bleu', 'jaune',
        'orange', 'marron']

        # couleur attribuée à la région et non encore choisie au départ
        self.couleur_attribuee = None # 4e attribut
```

1. Associer, en vous appuyant sur l'extrait de code précédent, les noms :
   nom, tab_voisines, tab_couleurs_disponibles et couleur_attribuee
   au terme qui leur correspond parmi : objet, attribut, méthode ou classe.

Corrigé

nom, tab_voisines, tab_couleurs_disponibles et couleur_attribuee
sont 4 attributs de la classe *Region*.

2. Indiquer le type du paramètre **nom_region** de la méthode **__init__** de la classe *Region*.

Corrigé

D'après la documentation, le paramètre **nom_region** est de type chaîne de caractères (string ou str)

3. Donner une instruction permettant de créer une instance nommée *ge* de la classe *Region* correspondant à la région dont le nom est « Grand Est » .

Corrigé

Une instruction permettant de créer une instance nommée **ge** de la classe *Region* correspondant à la région dont le nom est « Grand Est » est :

```
ge = Region('Grand Est')
```

4. Recopier et compléter la ligne 6 de la méthode de la classe *Region* ci-dessous :

Corrigé

On obtient :

```
1 def renvoie_premiere_couleur_disponible(self): # methode 2 :
2         '''
3         Renvoie la première couleur du tableau des couleurs disponibles
4         supposé non vide.
5         :  return (str)
6         '''
7         return self.tab_couleurs_disponibles[0]
```

5. Recopier et compléter la ligne 6 de la méthode de la classe Region ci-dessous :

Corrigé

On obtient :

```
1 def renvoie_nb_voisines(self) :
2         '''
3         Renvoie le nombre de régions voisines.
4         : return (int)
5         '''
6         return len(self.tab_voisines)
```

6. Compléter la méthode de la classe Region ci-dessous à partir de la ligne 6 :

Corrigé

On obtient :

```
1  def est_coloriee(self):
2          '''
3          Renvoie True si une couleur a été attribuée à cette région
4          et False sinon. : return (bool)
5          '''
6          if self.couleur_attribuee == None:
7              return False
8          else:
9              return True
10
11 # ou encore
12
13 def est_coloriee(self):
14         '''
15         Renvoie True si une couleur a été attribuée à cette région
16         et False sinon. : return (bool)
17         '''
18         return self.couleur_attribuee != None
19
20
```

7. Compléter la méthode de la classe Region ci-dessous à partir de la ligne 8 :

Corrigé

On obtient :

```
1  def retire_couleur(self, couleur):
2          '''
3          Retire couleur du tableau de couleurs disponibles de la région si
4          elle est dans ce tableau. Ne fait rien sinon.
5          : param couleur (str)
6          : ne renvoie rien : effet de bord sur le tableau des couleurs
7          disponibles'''
8          if couleur in self.tab_couleurs_disponibles:
9              self.tab_couleurs_disponibles.remove(couleur)
```

8. Compléter la méthode de la classe Region ci-dessous, à partir de la ligne 7, en utilisant une boucle :

Corrigé

On obtient :

```
1  def est_voisine(self, region):
2          '''
3          Renvoie True si la region passée en paramètre est une voisine
4          et False sinon.
5          : param region (Region) : return (bool)
6          '''
7          for region_voisine in self.tab_voisines:
8              if region == region_voisine:
9                  return True
10         return False
```

ou encore si on n'avait pas imposé de boucle, plus simple dans le genre pythonnade :

```
1  def est_voisine(self, region):
2          '''
3          Renvoie True si la region passée en paramètre est une voisine
4          et False sinon.
5          : param region (Region) : return (bool)
6          '''
7          return region in self.tab_voisines
8
```

Partie 2

On a créé une classe *Pays* :

- cette classe modélise la carte d'un pays composé de régions ;
- lunique attribut tab_regions de cette classe est un tableau (type list en Python) dont les éléments sont des instances de la classe *Region*.

9. Recopier et compléter la méthode de la classe Pays ci-dessous à partir de la ligne 7 :

Corrigé

On obtient :

```
1   def renvoie_tab_regions_non_coloriees(self):
2          '''
3          Renvoie un tableau dont les éléments sont les régions
4          du pays sans couleur attribuée.
5          : return (list) tableau dinstances de la classe Region
6          '''
7          tableau = []
8          for region in self.tab_regions:
9              if not region.est_coloriee():
10                 tableau.append(region)
11         return tableau
```

10. On considère la méthode de la classe Pays ci-dessous.

```
def renvoie_max(self):
        nb_voisines_max = -1
        region_max = None
        for reg in self.renvoie_tab_regions_non_coloriees():
            if reg.renvoie_nb_voisines() > nb_voisines_max:
                nb_voisines_max = reg.renvoie_nb_voisines()
                region_max = reg
        return region_max
```

(a) Expliquer dans quel cas cette méthode renvoie None.

Corrigé

La méthode **renvoie_max()** renvoie None lorsque toutes les régions ont été colorisées (également lorsque la liste des région du pays est vide)

(b) Indiquer, dans le cas où cette méthode ne renvoie pas None, les deux particularités de la région renvoyée

Corrigé

La région renvoyée est la région **non coloriée** contenant le plus de régions voisines.

11. Coder la méthode **colorie(self)** de la classe **Pays** qui choisit une couleur pour chaque région du pays de la façon suivante :
    - On récupère la région non coloriée qui possède le plus de voisines.
    - Tant que cette région existe :
        - La couleur attribuée à cette région est la première couleur disponible dans son tableau de couleurs disponibles.
        - Pour chaque région voisine de la région :
            - si la couleur choisie est présente dans le tableau des couleurs disponibles de la région voisine alors on la retire de ce tableau.
        - On récupère à nouveau la région non coloriée qui possède le plus de voisines.

. Corrigé

```
def colorie(self):
        ''' colorie toutes les régions du pays
        : Ne renvoie rien (effet de bord sur les régions du pays)
        '''
        reg = self.renvoie_max()
        while reg != None :
            reg.couleur_attribuee = reg.renvoie_premiere_couleur_disponible()
            for voisine in reg.tab_voisines :
                voisine.retire_couleur(reg.couleur_attribuee)
            reg = self.renvoie_max()

```

↤ **Fin du devoir** ↦