DS 4

Informatique pour tous, première année

Julien Reichert

Exercice 1 : Ingéniérie numérique et simulation

Question 1.1 : Écrire un programme implémentant la méthode du pivot de Gauss en Python. La structure de données (liste de listes, tableau, voire autre chose) est laissée au choix. Il est possible d'écrire des sous-fonctions annexes et d'utiliser ou non `numpy` et `scipy`. Il faudra néanmoins que l'algorithme du pivot soit entièrement écrit.

Question 1.2 : Écrire sur la base de l'algorithme précédent un algorithme permettant de déterminer le rang d'une matrice. On ne vérifiera pas que l'argument représente bien une matrice carrée (là aussi en tant que structure de données au choix), et on admettra que ce sera le cas.

Il est possible de ne pas écrire le programme en entier, à condition de bien baliser les lignes à changer par rapport au programme de la question précédente. Bien entendu, l'utilisation de sous-fonctions à la question précédente montre l'avantage de la programmation modulaire !

Rappel mathématique : le rang d'une matrice se calcule entre autres en se ramenant par des opérations élémentaires **sur les lignes ou les colonnes** à une matrice diagonale dont le nombre d'éléments non nuls est précisément le rang.

Exercice 2 : Bases de données

Pour cet exercice, nous utiliserons une version simplifiée d'une base de données que j'ai créée pendant le confinement pour recenser les spectacles musicaux auxquels j'ai assisté, ainsi que diverses informations à leur propos.

- Table Spectacle, avec les attributs Id_spectacle (clé primaire avec auto-incrémentation), Date (chaîne de caractères [1]), Titre (chaîne de caractères), Compositeur (chaîne de caractères), Type (chaîne de caractères), Pays (chaîne de caractères), Ville (chaîne de caractères) et Salle (chaîne de caractères pouvant être NULL). Le couple (Date, Titre) est aussi une clé, mais pas l'attribut Date [2].
- Table Distribution, avec les attributs Id_spectacle (clé étrangère pour la table Spectacle), Sur_scene (booléen, assimilable à un entier valant forcément 0 ou 1, respectivement pour faux et vrai), Role (chaîne de caractères) et Personne (chaîne de caractères). Le couple (Id_spectacle, Role) est une clé.

Pour clarifier, les valeurs possibles de l'attribut Type de la table Spectacle sont `"Opéra"`, `"Ballet"`, `"Comédie musicale"` et `"Concert"`, ce qui permet de ne pas se tromper quand ces valeurs apparaitront dans une question. De plus, l'attribut Sur_scene est à faux pour des valeurs de Role telles que `"Chef d'orchestre"`, `"Metteur en scène"`, etc. On en déduit que Personne est l'attribut correspondant à l'identité des chanteurs / acteurs / directeurs...

Question 2.1 : Écrire une requête permettant d'obtenir la liste des pays où j'ai assisté à au moins un spectacle.

Question 2.2 : Écrire une requête permettant d'obtenir le nombre d'opéras auxquels j'ai assisté.

1. ... au format JJ/MM/AAAA, pas très idéal pour trier par ordre chronologique, heureusement qu'il y a l'identifiant pour cela !
2. Je peux être très motivé certains jours...

Question 2.3 : Écrire une requête permettant d'obtenir le nom des chefs d'orchestre des productions de "La Traviata" que j'ai vues. [3]

Question 2.4 : Écrire une requête permettant d'obtenir le titre du spectacle avec le plus de personnes enregistrés dans la base de données au niveau de la distribution. En cas d'égalité, on pourra au choix renseigner un titre arbitraire ou tous.

Question 2.5 : Même question en se restreignant aux rôles sur scène.

Question 2.6 : Écrire une requête permettant d'obtenir le nom de la ville où il y a eu le plus grand nombre de salles différentes dans lesquelles j'ai assisté à au moins un spectacle. Même consigne quant aux égalités éventuelles. [4]

Question 2.7 : Expliquer ce que font les trois requêtes ci-dessous.

```
SELECT Personne, COUNT(*)
FROM Spectacle JOIN Distribution
ON Spectacle.Id_spectacle=Distribution.Id_spectacle
WHERE Role="Chef d'orchestre" AND Sur_scene=0
GROUP BY Personne
HAVING COUNT(*) > 1
ORDER BY COUNT(*) DESC
```

```
SELECT Personne, COUNT(DISTINCT Ville)
FROM Spectacle JOIN Distribution
ON Spectacle.Id_spectacle=Distribution.Id_spectacle
WHERE Sur_scene=1
GROUP BY Personne
HAVING COUNT(DISTINCT Ville) > 1
ORDER BY COUNT(DISTINCT Ville) DESC
```

```
SELECT Compositeur, COUNT(*)
FROM Spectacle
WHERE Type="Opéra"
GROUP BY Compositeur
HAVING COUNT(*) =
(
  SELECT COUNT(*)
  FROM Spectacle
  WHERE Type="Opéra"
  GROUP BY Compositeur
  ORDER BY COUNT(*) DESC
  LIMIT 1
)
```

3. Pas besoin de gérer d'éventuels doublons, il n'y en a d'ailleurs pas à l'heure actuelle.
4. La réponse SELECT "Paris" n'est pas autorisée !