DS 5

Informatique pour tous, première année

Julien Reichert

Ce devoir consiste en un exercice (difficile) de programmation et un certain nombre de questions sur une unique base de données fournie. Les requêtes à écrire ne sont pas forcément de difficulté croissante (d'après la préparation du corrigé).

Exercice 1 : Écrire une fonction effectuant un tirage au sort biaisé de valeurs entières dans un intervalle précisé. La spécification de la fonction sera la suivante : on donne comme premier argument un entier `n` donnant la taille de la liste à renvoyer et comme deuxième argument un entier naturel non nul `k` tel que les éléments de la liste soient entre `0` et `k-1` ; inutile de vérifier que les arguments sont cohérents, par ailleurs. En sortie : une liste respectant les contraintes annoncées ci-avant. Les contraintes du tirage au sort : si une valeur `i` a déjà été piochée `ni` fois et que le plus petit nombre d'occurrences d'une valeur est `m`, alors on donne un coefficient `1 / (ni - m + 1)` à la valeur `i`, et la probabilité pour chaque valeur d'être piochée doit être égale au coefficient de la valeur divisé par la somme des coefficients.

Après tests, l'écart-type est sensiblement le même qu'en suivant la loi uniforme, mais engendrer 10000 nombres entre 0 et 99 donne régulièrement des nombres d'occurrences toujours entre 96 et 104 au lieu de 60 à 140 potentiellement observés en suivant la loi uniforme...

On pourra se servir de la fonction `random` du module éponyme, qui engendre un flottant entre `0` et `1`. Toutes les autres fonctions s'en déduisent, et il n'est pas recommandé d'utiliser `randint` vu que les valeurs manipulées sont a priori des flottants (ou alors il faut faire une mise au même dénominateur, mais la complexité a toutes les chances d'en pâtir).

Voici la base de données pour le reste du devoir. Il s'agit de gérer des paris[1] sur des matchs du championnat d'Europe, avec un calcul de score fait par ailleurs (Python / PHP / etc.) en général. Pour chaque match, annoncer le bon résultat (équipe 1 gagne / match nul / équipe 2 gagne) donne un point au parieur, annoncer la bonne différence de buts hors match nul donne un point supplémentaire, annoncer le bon nombre de buts d'une équipe donne un point même si le bon résultat n'est pas donné, mais annoncer le bon score donne trois points plutôt que quatre si on respectait cette formule. On ne considère pas de prolongations ici.

La base de donnée est constituée de cinq tables :

- `Equipes`, dont les attributs sont `Id_equipe` (entier) et `Pays` (chaîne de caractères) ;
- `Matchs`, dont les attributs sont `Id_match` (entier), `Date` (chaîne de caractères au format `"MM/JJ"`), `Phase` (chaîne de caractères pouvant valoir `"Groupe A"`, `"Quart de finale"`, ...), `Equipe1` (entier, l'équipe dite « receveuse ») et `Equipe2` (entier, l'équipe dite « visiteuse ») ;
- `Parieurs`, dont les attributs sont `Id_parieur` (entier) et `Nom_parieur` (chaîne de caractères) ;
- `Paris`, dont les attributs sont `Parieur` (entier), `Id_match` (entier), `Score1` (entier) et `Score2` (entier).
- `Resultats`, dont les attributs sont `Id_match` (entier), `Buts1` (entier) et `Buts2` (entier).

Exemples d'enregistrements (avec les attributs dans l'ordre) pour chaque table avec explications si besoin :

- Table `Equipes` : `(1, "France"), mais aussi (24, "Hongrie")`. Plutôt intuitif. L'identifiant est arbitraire.
- Table `Matchs` : `(22, "06/19", "Groupe F", 24, 1)`. Allez les Bleus !
- Table `Parieurs` : `(1, "Julien")`. Intuitif aussi.
- Table `Paris` : `(1, 22, 0, 3)`. Je pense que la France va gagner contre la Hongrie 3 à 0.
- Table `Resultats` : `(22, 0, 4)`. Wow, encore mieux ! 2 points pour moi : puisque le résultat est de 4 à 0, l'équipe annoncée a gagné et le nombre de buts d'une équipe est correct.

Exercice 2 : Décrire pour chaque table des clés pertinentes, en mentionnant les clés étrangères.

Exercice 3 : Pourquoi les points marqués par les parieurs ne figurent-ils dans aucune table ?

Exercice 4 : Serait-il envisageable de rassembler des tables ? Discuter des avantages et des inconvénients.

Exercice 5 : Écrire une requête permettant de connaître le nom de l'équipe d'identifiant `19` et une requête permettant de connaître l'identifiant de l'Autriche.

Exercice 6 : Écrire une requête permettant de connaître le nom des équipes jouant la finale.

Exercice 7 : Écrire une requête permettant de lister tous les paris sur le match Hongrie-France (numéro 22, cette information peut être utilisée pour s'épargner des jointures). Le résultat devra être formé de trois attributs : le nom du parieur, le score de la Hongrie et le score de la France, ces attributs devant être renommés pour être explicites.

Exercice 8 : Écrire une requête permettant de connaître le nom des équipes jouant le 30 juin. Un seul attribut devra figurer par enregistrement dans le résultat de la requête. On signale qu'il n'est pas nécessaire de traiter les doublons, car une équipe ne joue jamais deux fois dans la même journée.

Exercice 9 : Écrire une requête déterminant pour chaque équipe son nom, le premier jour où elle a un match et le dernier jour où elle a un match. Les dates sont ordonnées naturellement vu le format annoncé.

Exercice 10 : Écrire une requête déterminant tous les paris (en tant que couples formés du parieur et du match) ayant donné trois points.

Exercice 11 : Écrire une requête déterminant tous les paris (idem) ayant donné exactement un point. On prendra bien soin de mentionner d'abord précisément tous les cas menant à l'obtention d'un point exactement.

Exercices non donnés : idem deux points, puis récupérer le score total d'un parieur puis le classement des parieurs. Comme annoncé, laissons agir un langage où les boucles sont implémentées, cela facilite grandement les choses. À ce stade, même les exercices 10 et 11 sont alors remplacés par un traitement à partir d'une récupération brute des données.

1. amicaux, ne pas voir le pognon partout !