Correction du DS 5

Julien Reichert

Exercice 1

```
import random

def tirage(n, k):
    l = []
    occurrences = [0] * k
    scores, score_total = [1] * k, k # coefficients et total des coefficients, utiles
    min_occ, nb_min_occ = 0, k # nombre minimal d'occurrences et combien ont ce minimum, utiles aussi
    for _ in range(n):
        alea = score_total * random.random()
        indice = 0
        while indice < k-1 and alea > scores[indice]: # sécurité pour ne pas déborder
        # débordement impossible en théorie mais la précision des flottants...
            alea -= scores[indice]
            indice += 1
        l.append(indice)
        occurrences[indice] += 1
        if occurrences[indice] == min_occ+1: # donc était min_occ
            nb_min_occ -= 1
            if nb_min_occ == 0:
                min_occ += 1
                score_total = 0
                for ind in range(k):
                    scores[ind] = 1 / (occurrences[ind] - min_occ + 1)
                    score_total += scores[ind]
                    if occurrences[ind] == min_occ:
                        nb_min_occ += 1
        else:
            nouveau_score = 1 / (occurrences[indice] - min_occ + 1)
            score_total = score_total - scores[indice] + nouveau_score
            scores[indice] = nouveau_score
    return l
```

Exercice 2

- `Equipes` : chaque attribut est une clé, mais si déjà on crée un identifiant, ce sera la clé primaire.
- `Matchs` : `Id_match` est une clé (clé primaire), `Date` ne sera pas une clé car il y a des jours avec plusieurs matchs (non pénalisé car tout le monde ne peut pas savoir), même en ajoutant `Phase` (qui n'est clairement pas une clé !), et le couple `(Equipe1, Equipe2)` non plus (Portugal-Grèce en 2004, par exemple). Cependant, il est vrai que le couple `(Date, Equipe1)` (idem avec `Equipe2`), ainsi que le triplet `(Phase, Equipe1, Equipe2)` (pas de match retour) est une clé. Les attributs `Equipe1` et `Equipe2`, au vu du type mentionné, sont des clés étrangères vers la table `Equipes`.
- `Parieurs` : cf. `Equipes`.
- `Paris` : seul le couple `(Parieur, Id_match)` a du sens. Il est formé de deux clés étrangères (tables intuitives).
- `Resultats` : l'attribut `Id_match` est une clé primaire, qui est par ailleurs une clé étrangère vers la table

`Matchs`.

Exercice 3

Comme on le voit sur l'énoncé des dernières questions, cette information se déduit du contenu des autres tables. Ceci étant, il n'est pas interdit qu'il y ait une redondance dans une base de données, donc la table `Paris` aurait pu avoir un attribut `Score_pari` valant `NULL` jusqu'à ce que le résultat du match soit connu, occasionnant une mise à jour en même temps que l'insertion du résultat dans la table `Resultats`.

Exercice 4

Les tables à deux attributs (identifiant et valeur non numérique) peuvent très bien disparaître au profit du remplacement dans toutes les tables où l'identifiant est mentionné en tant que clé étrangère de la valeur. Ceci permet une meilleure lecture par l'humain des données de ces tables, mais cette lecture n'est en général pas faite directement par l'utilisateur, et le surcoût en espace engendré suggère de maintenir la structure telle qu'elle a été présentée.

En outre, le résultat d'un match pourrait donner lieu à d'autres attributs dans la table des matchs, quitte à mettre à jour ces champs, initialement à `NULL` au moment de la création de l'enregistrement. C'est ici un choix personnel.

Exercice 5

```
SELECT Pays FROM Equipes WHERE Id_equipe = 19;

SELECT Id_equipe FROM Equipes WHERE Pays = "Autriche";
```

Ne pas oublier les guillemets !

Exercice 6

Un résultat avec les deux équipes (apprendre à faire des auto-jointures) :

```
SELECT Equipes.Pays AS Finaliste_1, Equipes2.Pays AS Finaliste_2
FROM Equipes JOIN Matchs ON Equipes.Id_equipe = Matchs.Equipe1
JOIN Equipes AS Equipes2 ON Equipes2.Id_equipe = Matchs.Equipe2
WHERE Phase = "Finale"
```

Deux résultats, un par équipe (plus simple) :

```
SELECT Pays FROM Equipes WHERE Id_equipe IN
(SELECT Equipe1 AS Finaliste FROM Matchs WHERE Phase = "Finale"
UNION
SELECT Equipe2 AS Finaliste FROM Matchs WHERE Phase = "Finale")
```

Il est aussi possible de faire une réunion de deux tables obtenues par une jointure (principe analogue).

Exercice 7

```
SELECT Nom_parieur AS Joueur, Score1 AS Score_Hongrie, Score2 AS Score_France
FROM Parieurs JOIN Paris ON Id_parieur = Parieur WHERE Id_match = 22
```

Version complète :

```
SELECT Nom_parieur AS Joueur, Score1 AS Score_Hongrie, Score2 AS Score_France
```

```
FROM Parieurs
JOIN Paris ON Id_parieur = Parieur
JOIN Matchs ON Paris.Id_match = Matchs.Id_match
JOIN Equipes ON Equipe1 = Equipes.Id_equipe
JOIN Equipes AS Equipes2 ON Equipe2 = Equipes2.Id_equipe
WHERE Equipes.Pays = "Hongrie" AND Equipes2.Pays = "France"
```

Exercice 8

```
SELECT Pays FROM Equipes WHERE Id_equipe IN
(SELECT Equipe1 FROM Matchs WHERE Date="06/30"
UNION
SELECT Equipe2 FROM Matchs WHERE Date="06/30")
```

Comme d'habitude, une jointure peut remplacer l'opérateur `IN`.

Exercice 9

```
SELECT Pays, MIN(Date), MAX(Date) FROM Equipes JOIN Matchs
WHERE Id_equipe = Equipe1 OR Id_equipe = Equipe2
GROUP BY Pays
```

Ici, le `WHERE` peut être remplacé par `ON`, mais cela dépasse le cadre autorisé des jointures au programme.

Exercice 10

```
SELECT Parieur, Paris.Id_match FROM Paris JOIN Resultats ON Paris.Id_match = Resultats.Id_match
WHERE Score1 = Buts1 AND Score2 = Buts2
```

Exercice 11

Cas où un seul point est marqué :

- Une équipe a gagné et on a deviné laquelle, mais ni le premier score, ni le deuxième, ni la différence n'est correcte.
- Conformément au pari, il y a eu un match nul, mais le nombre de buts est incorrect.
- L'issue du match n'est pas celle sur laquelle on a misé, mais un des deux scores était correct.

On va donc faire une réunion de ces cas ou une disjonction des conditions.

```
SELECT Parieur, Paris.Id_match FROM Paris JOIN Resultats ON Paris.Id_match = Resultats.Id_match
WHERE
(Score1 > Score2 AND Buts1 > Buts2 AND Score1 <> Buts1
  AND Score2 <> Buts2 AND Score1 - Score2 <> Buts1 - Buts2)
OR (Score1 = Score2 AND Buts1 = Buts2 AND Score1 <> Buts1)
OR
(
  (
    (Score1 > Score2 AND Buts1 <= Buts2) OR (Score1 < Score2 AND Buts1 >= Buts2)
      OR (Score1 = Score2 AND Buts1 <> Buts2)
  )
    AND (Score1 = Buts1 OR Score2 = Buts2)
)
```