Correction du DS 1

Julien Reichert

Partie 1

```
# Exercice 1.1

def envoie_premier_fin(l):
    x = l[0]
    for i in range(len(l) - 1):
        l[i] = l[i+1]
    l[-1] = x

# Exercice 1.2

def pgd(l):
    rep = l[1] - l[0]
    for i in range(1, len(l) - 1):
        if l[i+1] - l[i] > rep:
            rep = l[i+1] - l[i]
    return rep

# Exercice 1.3

def ppd(l):
    if l[0] == l[-1]:
        assert False
    rep = l[-1] - l[0]
    for i in range(len(l) - 1):
        if l[i+1] != l[i] and l[i+1] - l[i] < rep:
            rep = l[i+1] - l[i]
    return rep

# Exercice 1.4

def verif(l):
    for i in range(1, len(l)):
       for j in range(i):
           if l[j][0] >= l[i][0] and l[j][1] >= l[i][1]:
               return False
    return True
```

```
# Exercice 1.5

def nombre_chiffres(n):
    n = abs(n)
    if n < 10:
        return 1
    rep = 0
    while n > 0:
        rep += 1
        n //= 10
    return rep
```

Partie 2

Exercice 2.1

Dans la table `Commandes`, `id_client` et `id_produit` sont des clés étrangères vers les attributs homonymes.

Exercice 2.2

Les entités concernées sont les clients et les produits. La relation est « Le client commande un produit. ».

Exercice 2.3

```
SELECT nom_produit FROM Produits WHERE stock <= 5
```

Exercice 2.4

```
SELECT Commandes.* FROM Clients JOIN Commandes ON Clients.id_client = Commandes.id_client
WHERE identite = id
```

Exercice 2.5

```
SELECT Commandes.* FROM Produits JOIN Commandes ON Produits.id_produit = Commandes.id_produit
WHERE quantite > stock
```

Exercice 2.6

```
SELECT SUM(stock) AS stock_total FROM Produits GROUP BY categorie ORDER BY SUM(stock) DESC LIMIT 1
```

Exercice 2.7

```
SELECT Produits.id_produit FROM Produits JOIN Commandes ON Produits.id_produit = Commandes.id_produit
GROUP BY Produits.id_produit, stock HAVING SUM(quantite) > stock
```

Exercice 2.8

```
SELECT id_produit FROM Commandes GROUP BY id_produit HAVING COUNT(DISTINCT id_client) =
(SELECT COUNT(*) FROM Clients)
```

Le `DISTINCT` permet de ne pas tenir compte du fait qu'un produit puisse avoir été commandé plusieurs fois par un même client.

Partie 3

Exercice 3.1

Les attributs `ingredient` et `allergene` de la table `INGRALL` sont des clés étrangères respectivement vers les attributs `idingr` de la table `INGREDIENTS` et `idallerg` de la table `ALLERGENES`.

Les attributs `contributeur` de la table `RECETTES` et `utilisateur` de la table `EVALUATIONS` sont tous deux des clés étrangères vers l'attribut `idutil` de la table `UTILISATEURS`.

Les attributs `recette` et `ingredient` de la table `INGREC` sont des clés étrangères respectivement vers les attributs `idingr` de la table `INGREDIENTS` et `idrec` de la table `RECETTES` (l'attribut `recette` de la table `EVALUATIONS` est également une clé étrangère vers celui-ci).

Exercice 3.2

Les entités sont les internautes, les ingrédients, les allergènes et les recettes.

La relation « L'internaute publie une recette. » ne fait pas l'objet d'une table car il s'agit d'une relation 1-* dans notre modèle, donc il est possible de se simplifier la vie en ayant l'attribut correspondant dans la table `RECETTES`.

La relation « L'ingrédient contient un allergène. » fait l'objet d'une table, il s'agit d'une relation *-*.

Idem pour la relation « La recette nécessite un ingrédient. ».

Idem pour la relation « L'utilisateur note une recette. ».

Exercice 3.3

```
SELECT DISTINCT nom FROM RECETTES JOIN UTILISATEURS ON contributeur = idutil WHERE identite = id
```

Exercice 3.4

```
SELECT DISTINCT ALLERGENES.allergene FROM ALLERGENES
JOIN INGRALL ON idallerg = INGRALL.allergene
JOIN INGREC ON INGRALL.ingredient = INGREC.ingredient
JOIN RECETTES ON recette = idrec
WHERE nom = 'Cheesecake au citron'
```

Exercice 3.5

```
SELECT nom FROM INGREDIENTS
JOIN INGREC ON INGREDIENTS.idingr = INGREC.ingredient
JOIN RECETTES ON recette = idrec
WHERE INGREDIENTS.ingredient = 'œuf'

INTERSECT

SELECT nom FROM INGREDIENTS
JOIN INGREC ON INGREDIENTS.idingr = INGREC.ingredient
JOIN RECETTES ON recette = idrec
WHERE INGREDIENTS.ingredient = 'beurre'
```

Exercice 3.6

Version simple :

```
SELECT nom FROM RECETTES JOIN EVALUATIONS ON idrec = recette
GROUP BY idrec, nom ORDER BY AVG(note) DESC LIMIT 1
```

Version tenant compte des égalités :

```
SELECT nom FROM RECETTES JOIN EVALUATIONS ON idrec = recette
GROUP BY idrec, nom HAVING AVG(note) =
  (
    SELECT AVG(note) AS meilleure_moyenne FROM RECETTES JOIN EVALUATIONS ON idrec = recette
    GROUP BY idrec ORDER BY AVG(note) DESC LIMIT 1
  )
```

Exercice 3.7

Version simple :

```
SELECT identite FROM UTILISATEURS JOIN EVALUATIONS ON idutil = utilisateur
GROUP BY idutil, identite ORDER BY COUNT(*) DESC LIMIT 3
```

Version tenant compte des égalités (afin de comprendre pourquoi on ne l'a pas demandée...) :

```
SELECT identite FROM UTILISATEURS JOIN EVALUATIONS ON idutil = utilisateur
GROUP BY idutil, identite HAVING COUNT(*) =
  (
    SELECT COUNT(*) AS troisieme_max FROM UTILISATEURS JOIN EVALUATIONS ON idutil = utilisateur
    GROUP BY idutil, identite ORDER BY COUNT(*) DESC LIMIT 1 OFFSET 2
  )
```

Exercice 3.8

```
SELECT nom FROM RECETTES

EXCEPT

SELECT nom FROM RECETTES
JOIN INGREC ON recette = idrec
JOIN INGRALL ON INGREC.ingredient = INGRALL.ingredient
JOIN ALLERGENES ON INGRALL.allergene = idallerg
WHERE ALLERGENES.allergene = 'fruit à coque'
OR ALLERGENES.allergene = 'crustacé'
```

Exercice 3.9

```
SELECT FLOOR(nbpers * 1000 / quantite) FROM RECETTES
JOIN INGREC ON idrec = recette
JOIN INGREDIENTS ON INGREC.ingredient = idingr
WHERE INGREDIENTS.ingredient = 'farine' AND nom = 'crêpes'
```