

Devoir d'informatique numéro 1

corrigé

Base de données « Vidéo-club »

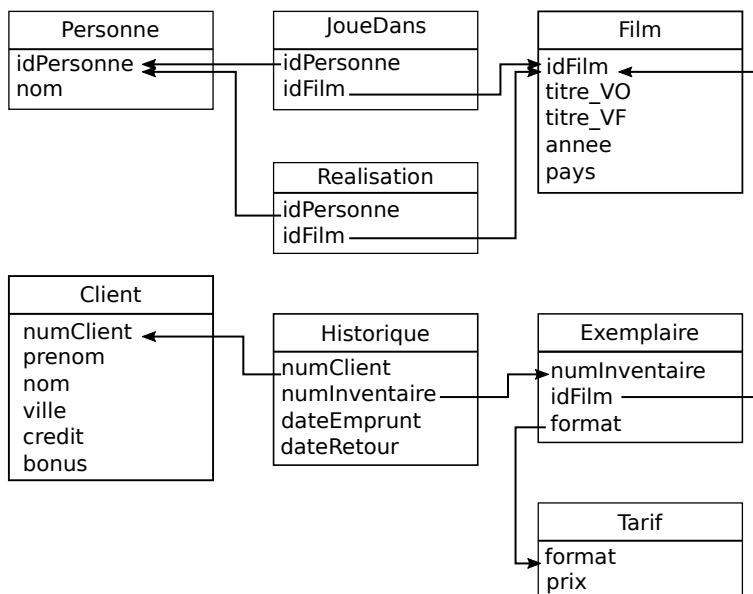
1. Les clés primaires sont les suivantes :

- pour la table *JoueDans* : les attributs *idPersonne*, *idFilm*;
- pour la table *Realisation* : les attributs *idPersonne*, *idFilm*;
- pour la table *Historique* : les attributs *numClient*, *numInventaire*, *dateEmprunt*.

2. L'association « une personne *réalise* un film » est de classe un-à-plusieurs. Dans la BDD, elle est prise en compte par l'attribut *idFilm* dans la table *Realisation*. Il constitue une clé étrangère vers la table *Film*.

3. L'association « une personne *joue dans* un film » est de classe plusieurs-à-plusieurs et elle met en relation les tables *Personne* et *Film*. Elle est prise en compte par l'introduction de la table *JoueDans* dont les deux attributs *idPersonne* et *idFilm* sont des clés étrangères vers les tables *Personne* et *Film*. L'association « une personne *joue dans* un film » est ainsi décomposée en deux associations un-à-plusieurs, la première entre *Personne* et *JoueDans*, la seconde entre *Film* et *JoueDans*.

Pour répondre aux questions qui suivent, il est bon d'avoir en tête l'ensemble des clés étrangères et des tables auxquelles elles font référence. Représenter au brouillon le schéma de la BDD peut y aider.



4. `SELECT Prenom, Nom FROM Client ORDER BY nom, prenom ;`

5. `SELECT nom, ville FROM client WHERE 5 <= credit <= 10 AND bonus > 0 ;`

6. `SELECT AVG(bonus) FROM Client WHERE ville = 'Gif-sur-Yvette' ;`

7. `SELECT film.titre_VO FROM film WHERE pays = 'France' AND 1960 <= annee <= 2000 ;`

8. `SELECT JoueDans.idPersonne FROM JoueDans
INTERSECT
SELECT Realisation.idPersonne FROM Realisation ;`

9. `SELECT Film.titre_VO, MIN(annee) FROM Film ;`

10. `SELECT COUNT(*) AS 'Nb_de_Films_au_XXeme' FROM Film WHERE 1901 <= annee <= 2000 ;`

11. `SELECT COUNT(DISTINCT idPersonne) FROM JoueDans ;`

12. `SELECT Film.titre_VO
FROM Film
JOIN JoueDans ON Film.idFilm = JoueDans.idFilm
JOIN Personne ON JoueDans.idPersonne = Personne.idPersonne
WHERE Personne.nom = 'Jean-Paul_Belmondo' ;`

4

13. `SELECT Film.titre_VO
FROM Film
JOIN JoueDans ON Film.idFilm = JoueDans.idFilm
JOIN Personne ON Personne.idPersonne = JoueDans.idPersonne
WHERE Personne.nom = 'Simone_Signoret'
INTERSECT
SELECT Film.titre_VO
FROM Film
JOIN JoueDans ON Film.idFilm = JoueDans.idFilm
JOIN Personne ON Personne.idPersonne = JoueDans.idPersonne
WHERE Personne.nom = 'Alain_Delon' ;`

5

10

14. `SELECT Personne.nom
FROM Personne JOIN Realisation ON Personne.idPersonne = Realisation.idPersonne
JOIN Film ON Film.idFilm = Realisation.idFilm
WHERE Film.titre_VO = 'Gran_Torino' ;`

4

15. `SELECT pays, annee, COUNT(*) AS 'Nb'
FROM Film
GROUP BY pays, annee
ORDER BY Nb DESC;`

1

16. `SELECT DISTINCT Client.nom, Client.Prenom
FROM Client
JOIN Historique ON Client.numClient = Historique.numClient
WHERE 2005-01-01 < Historique.dateEmprunt < 2005-12-31 ;`

1

17. `SELECT COUNT(*)
FROM Exemple JOIN FILM ON Exemple.idFilm = Film.idFilm
WHERE Film.titre_VO = 'Pinocchio' ;`

1

18. `SELECT Exemple.format, COUNT(DISTINCT Film.idFilm) -- Attention à DISTINCT
FROM Exemple
JOIN Film ON Exemple.idFilm = Film.idFilm
GROUP BY Exemple.format ;`

2

19. `SELECT Personne.nom, Film.titre_VO
FROM Film JOIN Realisation ON Realisation.idFilm = Film.IdFilm
JOIN JoueDans ON JoueDans.idFilm = Film.idFilm
JOIN Personne ON Personne.idPersonne = JoueDans.idPersonne
WHERE JoueDans.idPersonne = Realisation.idPersonne ;`

1

20. `SELECT Film.titre_VO FROM Film JOIN Exemple ON Film.idFilm = Exemple.idFilm
JOIN Historique ON Historique.numInventaire = Exemple.numInventaire
JOIN Client ON Client.numClient = Historique.numClient
WHERE Client.nom = 'ROUX' AND Client.prenom = 'Jules' ;`

```
21. SELECT Client.Nom, COUNT(*) AS NB
FROM Historique
JOIN Client ON Client.numClient = Historique.numClient
GROUP BY Client.numClient ;
```

```
22. SELECT COUNT(*)
FROM Historique
JOIN Exempleaire ON Historique.numInventaire = Exempleaire.numInventaire
JOIN FILM ON Exempleaire.idFilm = Film.idFilm JOIN JoueDans ON Film.idFilm =
JoueDans.idFilm
JOIN Personne ON Personne.idPersonne = JoueDans.idPersonne
WHERE Personne.nom = 'Catherine_Deneuve' ;
```

```
23. SELECT Personne.nom, COUNT(*) AS NB
FROM Historique
JOIN Exempleaire ON Historique.numInventaire = Exempleaire.numInventaire
JOIN FILM ON Exempleaire.idFilm = Film.idFilm JOIN JoueDans ON Film.idFilm =
JoueDans.idFilm
JOIN Personne ON Personne.idPersonne = JoueDans.idPersonne
GROUP BY Personne.idPersonne
ORDER BY NB DESC ;
```

```
24. SELECT Personne.nom, COUNT(*) AS NbFilm
FROM Personne
JOIN Realisation ON Personne.idPersonne = Realisation.idPersonne
GROUP BY Personne.idPersonne
HAVING NbFilm >= 10 ;
```

```
25. SELECT SUM(Tarif.prix)
FROM Tarif
JOIN Exempleaire ON Tarif.format = Exempleaire.format ;
```

```
26. SELECT SUM(Tarif.prix)
FROM Tarif
JOIN Exempleaire ON Exempleaire.format = Tarif.format
JOIN Historique ON Historique.numInventaire = Exempleaire.numInventaire
WHERE 2010-01-01 <= Historique.dateEmprunt <= 2010-12-31
```

```
27. SELECT Client.nom, SUM(Tarif.prix)
FROM CLIENT
JOIN Historique ON Client.numClient = Historique.numClient
JOIN Exempleaire ON Exempleaire.numInventaire = Historique.numInventaire
JOIN Tarif ON Tarif.format = Exempleaire.format
GROUP BY Client.numClient
HAVING COUNT(*) >= 5;
```

```
28. SELECT Film.pays, COUNT(*) AS NB
FROM Film
JOIN Exempleaire ON Film.idFilm = Exempleaire.idFilm
JOIN Historique ON Historique.numInventaire = Exempleaire.numInventaire
GROUP BY Film.pays ORDER BY Nb DESC LIMIT 3
```

```
29. SELECT Personne.nom, SUM(Tarif.prix) AS CA, COUNT(DISTINCT Film.idFilm) AS NbFilm
FROM Tarif
JOIN Exempleaire ON Tarif.format = Exempleaire.format
JOIN Film ON Film.idFilm = Exempleaire.idFilm
JOIN Realisation ON Realisation.idFilm = Film.idFilm
```

```

JOIN Personne ON Personne.idPersonne = Realisation.idPersonne
JOIN Historique ON Historique.numInventaire = Exempleaire.numInventaire
GROUP BY Realisation.idPersonne
HAVING NbFilm >= 3
ORDER BY CA DESC LIMIT 10; ;

```

10

```

30. SELECT C1.nom, C1.prenom, C2.nom, C2.prenom
FROM Client AS C1 JOIN Client AS C2
ON C1.nom = C2.nom AND C1.prenom != C2.prenom AND C1.prenom < C2.prenom ;

```

```

31. SELECT P1.nom
FROM Personne AS P1
JOIN Realisation ON Realisation.idPersonne = P1.idPersonne
JOIN FILM ON Film.idFilm = Realisation.idFilm
JOIN JoueDans ON JoueDans.idFilm = Film.idFilm
JOIN Personne AS P2 ON P2.idPersonne = JoueDans.idPersonne
WHERE P2.nom = 'George Clooney' ;

```

2

7

```

32. SELECT DISTINCT P1.nom AS 'Réalisateur', P2.nom AS 'Acteur'
FROM Personne AS P1
JOIN Realisation ON Realisation.idPersonne = P1.idPersonne
JOIN Film ON Film.idFilm = Realisation.idFilm
JOIN JoueDans ON JoueDans.idFilm = Film.idFilm
JOIN Personne AS P2 ON P2.idPersonne = JoueDans.idPersonne ;

```

3

Modélisation d'un matériau magnétique

```

1. from math import exp, tanh
from random import random, randrange

```

```

10. def initialisation() :
    return [1 for i in range(n)]

```

```

11. def initialisation_anti() :
    s = []
    for a in range(h//2) :
        s = s + [1 for i in range(h)]
        s = s + [-1 for i in range(h)]
    return s

```

3

```

12. def repliement(s) :
    L = []
    i = 0
    for a in range(h) : # on parcourt les lignes
        Z = L[a*h:a*h+h]
        L.append(Z)
    return L

```

4

```

13. def liste_voisins(i) :
    ligne, col = i // h, i % h
    # gauche
    if col == 0 :
        ig = i + h - 1 # on avance de h-1 colonnes
    else :
        ig = i - 1
    # droite
    if col == h - 1 : # on recule de h-1 colonnes

```

5

```

        idr = i - h + 1
    else :
        idr = i + 1
    # bas
    if ligne == h - 1 :
        ib = i - (h-1) * h # on remonte (h-1) lignes
    else :
        ib = i + h
    # haut
    if ligne == 0 :
        ih = i + (h-1) * h # on descend (h-1) lignes
    else :
        ih = i - h

    return [ig, idr, ib, ih]

```

14. `def energie(s) :`
 `A = 0`
 `for i in range(n) :`
 `LV = liste_voisins(i)`
 `for j in LV :`
 `A += s[i] * s[j]`
 `return - A / 2`

15. On utilise `random` et une comparaison à p pour simuler une loi de Bernoulli de paramètre p . Le cas $\Delta e \leq 0$ pourrait être groupé avec l'autre : on obtiendrait $\exp(-\Delta e/(k_B T)) \geq 1$ et le test `x < p` donnerait toujours `True`.

```

def test_boltzmann(delta_e, T) :
    if delta_e <= 0 :
        return True
    else :
        p = exp(- delta_e / T)
        x = random()
        if x < p :
            return True
        else :
            return False

```

16. Dans la fonction `calcule_delta_e1`, on calcule la nouvelle énergie en parcourant tous les spins de la grille, avec une complexité en $O(n)$. Dans la fonction `calcule_delta_e2`, on calcule la nouvelle énergie par modification de l'ancienne, en considérant seulement les modifications dans le voisinage du spin d'indice i , avec une complexité en $O(1)$. Cette seconde méthode est donc plus efficace.

17. `def monte_carlo(s, T, n_tests) :`
 `for test in range(n_tests) :`
 `i = randrange(n)`
 `delta_e = calcule_delta_e2(s, i)`
 `if test_boltzmann(delta_e, T) :`
 `s[i] = - s[i]`
 `return`

18. `def aimantation_moyenne(n_tests, T) :`
 `s = initialisation()`
 `monte_carlo(s, T, n_tests)`
 `total = 0`
 `for spin in s :`
 `total += spin`
 `return total / n`

19. L'initialisation est en $O(n)$, l'appel à `monte_carlo` est en $O(n_tests)$, la boucle ligne 5 et 6 est en $O(n)$.

Finalement, la fonction `aimantation_moyenne` a une complexité asymptotique en $O(n + n_tests)$.

20. Si on voulait prendre en compte toutes les interactions entre spins, la fonction `calcule_delta_e2` serait alors en $O(n)$ de sorte que `aimantation_moyenne` aurait alors une complexité en $O(n_tests \times n)$.

21. On voit apparaître des domaines de spin uniforme. À basse température, ces domaines sont assez grand : leur taille est comparable à celle de l'échantillon entier. Plus la température augmente, plus ces échantillons deviennent petits, si petits qu'on ne parvient plus à les distinguer. Dans une même région de taille mésoscopique, on a alors des spins dans les deux sens, d'où une aimantation nulle. On retrouve le rôle de la température annoncé dans l'introduction : lorsque T augmente et franchit la température de Curie, le matériau se désaimante.

22. Cette question et les suivantes se rapprochent du parcours d'un graphe. On peut considérer les cases comme les nœuds d'un graphe et dire que deux nœuds sont connectés s'ils sont voisins et possèdent le même spin. Les domaines de Weiss sont les composantes connexes de ce graphe ; il s'agit de les déterminer et de les numéroté. La méthode adoptée est celle du parcours d'un graphe en profondeur ; on peut la coder de manière récursive ou en utilisant une pile explicite.

```
def explorer_voisinage(s, i, weiss, num) :
    LV = liste_voisins(i)
    for j in LV :
        if s[j] == s[i] and weiss[j] == -1 :
            weiss[j] = num
            explorer_voisinage(s, j, weiss, num)
```

5

23. La décomposition du travail entre les questions 23 et 24 ne me paraît pas très claire ; il est possible de l'envisager différemment de ce que je propose ci-dessous.

```
def explorer_voisinage_pile(s, i, weiss, num, pile) :
    j = pile.pop()
    weiss[j] = num # On marque ce spin lorsqu'on le dépile (cf énoncé)
    LV = liste_voisins(j)
    for u in LV : # On ajoute à la pile les spins qui doivent l'être
        if s[u] == s[i] and weiss[u] == -1 :
            pile.append(u)
```

4

```
24. def construire_domaine_weiss(s) :
    num = 0
    weiss = [-1 for i in range(n)]
    for i in range(n) :
        if weiss[i] != -1 :
            pile = [i] # Par construction, la pile ne contiendra que des nœuds du même domaine
            while pile != [] :
                explorer_voisinage(s, i, weiss, pile)
            num += 1
    return weiss
```

3

8