2021 - J2 - Polynésie

Exercice 5

A.1

- La clé primaire de la relation `Fims` est *idFilm*.
- La clé primaire de la relation `Abonnes` est *idAbonne*.

A.2

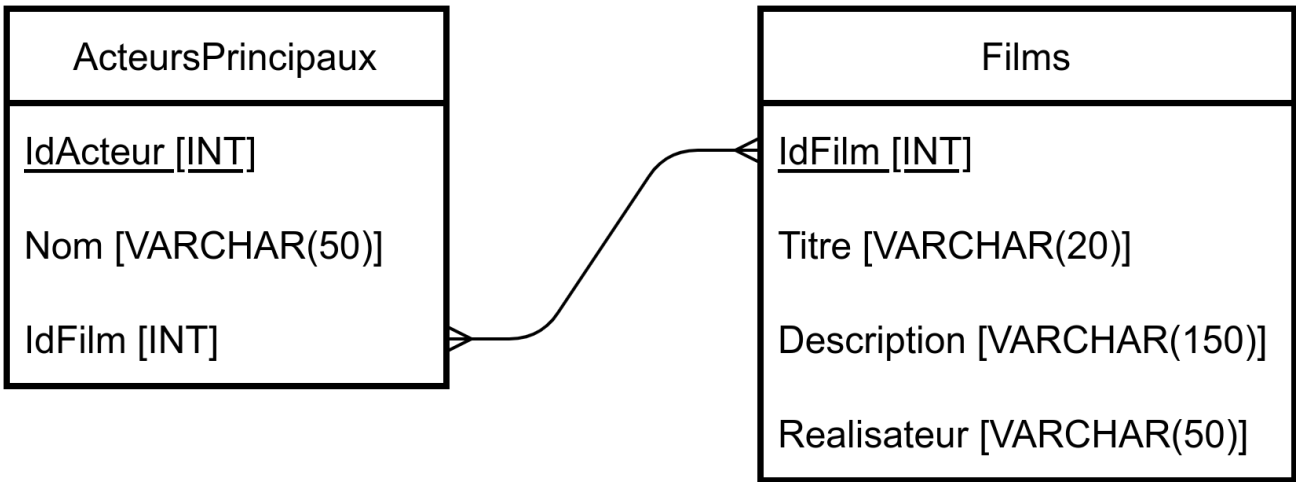
- `idFilm` est un nombre entier (`INT`).
- `Description` est une chaîne de caractère de taille variable d'au plus 150 caractères (`VARCHAR(150)`).

A.3 La relation `ComptesAbonnes` admet la clé primaire `idCpt` et la clé étrangère `IdAbonne`. On peut d'ailleurs reconnaître les clés étrangères au fait qu'elles font référence à la clé primaire d'autres relations.

A.4 Pour stocker les acteurs principaux de chaque film, il faut modifier le modèle relationnel (le *schéma relationnel*). Il faut donc créer une relation `ActeursPrincipaux` qui admet une clé primaire (un identifiant unique par acteur), des informations sur l'acteur (par exemple `Nom`, `DateNaissance`, etc.).

Pour associer les relations `ActeursPrincipaux` et `Films`, il faut ajouter un attribut `idFilm` de type clé étrangère reliant ces deux relations.

Enfin, pour savoir si les valeurs partagées doivent être unique, zéro, une ou plusieurs fois, il faut réaliser qu'*un acteur peut jouer dans plusieurs films* et qu'*un film possède plusieurs acteurs*. Et donc il n'y a pas d'unicité : il faut une cardinalité de zéro, un ou plusieurs du côté `ActeursPrincipaux.idFilm` et du côté `Films.idFilm`.



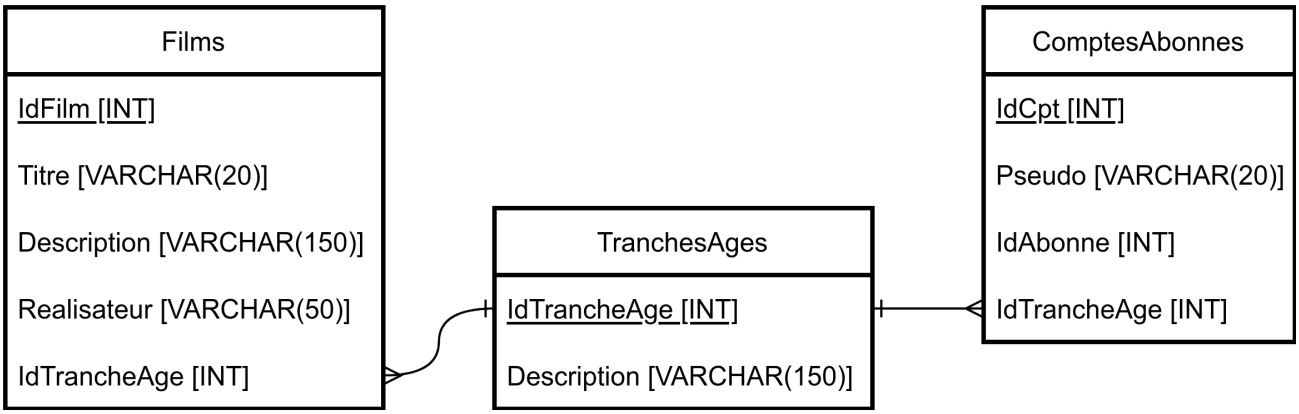
A.5 Pour ajouter une tranche d'âge, il faut commencer par créer la relation `TranchesAges`. Il faut au moins un attribut identifiant unique `idTrancheAge` de domaine `INT`. On peut ajouter aussi une `Description` de domaine `VARCHAR(150)`.

Ensuite il faut associer `TranchesAges` à la relation `ComptesAbonnes` car l'énoncé indique *associer une tranche d'âge à chacun des comptes d'un abonné*. Pour cela, il faut ajouter un attribut `idTrancheAge` dans la relation `ComptesAbonnes`. Ce nouvel attribut sera une clé étrangère.

Puis en déterminer sa cardinalité : *un compte possède une unique tranche d'âge* et *une tranche d'âge est associée à zéro, un ou plusieurs comptes*. L'unicité est du côté de la tranche d'âge et la multiplicité du côté du compte de l'abonné.

Maintenant, il faut associer `TranchesAges` à la relation à `Film`. Pour cela il faut ajouter un nouvel attribut `idTrancheAge` dans la relation `Film`. Cet attribut sera une clé étrangère.

Enfin, pour en déterminer sa cardinalité, on remarque qu'*un film possède une unique tranche d'âge* et qu'*une tranche d'âge est associée à zéro, un ou plusieurs films*. Ainsi l'unicité est du côté de la tranche d'âge et la multiplicité du coté du film.



B.1 Requête permettant de récupérer l'identifiant et le pseudo de tous les comptes rattachés à l'abonné 237 :

```
SELECT idCpt, Pseudo
FROM ComptesAbonnes
WHERE idAbonne = 237;
```

B.2 La requête ci-dessous permet de calculer la moyenne des notes données par les comptes d'abonnés pour le film d'identifiant 1542.

```
SELECT AVG(NbEtoiles)
FROM Votes
WHERE IdFilm = 1542;
```

B.3 La requête ci-dessous permet d'afficher par ordre de note décroissante l'identifiant, le titre et la note des films pour lesquels le compte 508 a voté.

```
SELECT u.IdFilm, u.Titre, v.NbEtoiles
FROM Films as u JOIN Votes as v
ON u.IdFilm = v.IdFilm
WHERE v.IdCpt = 508
ORDER BY v.NbEtoiles DESC;
```

B.4 Requête permettant de modifier le pseudo du compte 508 pour lui attribuer la valeur « Champion » :

```
UPDATE TABLE ComptesAbonnes
SET Pseudo = 'Champion'
```

```
WHERE IdAbonne = 508;
```

C.1 La fonction commence par vérifiée que l'argument `listeFilms` était bien un tableau (de type `list`) et qu'il était non vide (de longueur différente de 0).

Ensuite un accumulateur `somme_ecarts` et une boucle sont mis en place pour parcourir tous les films de la liste. Pour chaque film, la fonction calcule la valeur absolue (`abs()`) de la différence des notes mises par chacun des deux comptes passés en arguments. Cette différence est accumulée dans `somme_ecarts`.

Par exemple, si les deux comptes ont mis la même note, cette différence est nulle. Puis, au plus les notes sont différentes, au plus la différence est grande.

À la sortie de la boucle, la variable `somme_ecarts` contient la somme de toutes les différences de notes mises par chacune des comptes.

En divisant cette somme par le nombre total de films, on obtient la *somme des écarts par film*.

Cette fonction renvoie donc la moyenne des écarts de notes entre les deux comptes.

C.2

```
def conseilsFilms (idCpt):
    conseils = []
    filmsMieuxNotes = podiumCompte(idCpt)
    spectateursFilms = spectateurs(films_mieux_notes)

    for idSpectateur in spectateursFilms:
        if distance(idCpt, id, filmsMieuxNotes) < 10:
            conseil = conseil + podiumCompte(idSpectateur)[:3]

    return conseil
```

Le code précédent est écrit avec une syntaxe homogène, mais non conseillée en

Python : **CamelCase** Voici une syntaxe non homogène mais conseillée en Python : **snake_case**

```
def conseils_films (idCpt):
    conseils = []
    film_mieux_notes = podiumCompte(idCpt)
    spectateurs_films = spectateurs(films_mieux_notes)

    for id_spectateur in spectateurs_films:
        if distance(idCpt, id, films_mieux_notes) < 10:
            conseil = conseil + podiumCompte(idSpectateur)[:3]

    return conseil
```