OBJECTIF

Objectif :

- Réaliser des fonctions sous postgres qui répondent à des règles de gestions.
- Tester ces fonctions en les utilisant dans une requête ou en l'appelant directement.

Ressources :

- Vous disposez du diagramme de classe et du modèle physique de la base *département* sur laquelle vous devez réaliser les fonctions.
- Du diaporama de cours.

BASE UTILISEE : FILM

1. Lancez la console d'administration du SGBD PostGres :
   http://a-pedagoarles-postgresql.aix.univ-amu.fr/phppgadmin/
   login AMU / mot de passe AMU

2. Récupérez le script de création des tables (*BaseFilmsPostGres.sql*), copier/coller dans la fenêtre SQL
3. lancer le script
4. Vos tables sont "normalement" créées et renseignées dans le schéma de données *TP_bd_film,* vérifiez le en utilisant la partie gauche de phppgadmin

Vous pouvez commencer le TP, à vous de jouer!!

TRAVAIL A FAIRE

1. Fonction retournant la liste de tous les films d'un réalisateur passé en paramètre (son nom et prénom) Erreur si le réalisateur n'existe pas

   Tester de la fonction avec les valeurs suivantes :
   'Hitchck', 'Alfred' -- erreur
   'Hitchcock', 'Alfred' **-- retourne la liste**

2. Fonction retournant le réalisateur (nom et prénom sur une seule colonne) d'un film d'après son titre. Erreur si le film n'existe pas.

   Tester de la fonction avec les valeurs suivantes :
   'film qui n'existe pas' **-- renvoie une erreur**
   'Pulp fiction' **-- renvoie Tarentino Quentin**

3. Fonction qui prend un identifiant d'artiste en entrée, et nous dit si c'est un réalisateur et/ou un acteur. Erreur l'identifiant n'existe pas
   Tester de la fonction avec les valeurs suivantes :

   37 -- acteur et réalisateur
   3 -- réalisateur
   18 -- acteur

4. Ecrire un trigger sur l'ajout d'une note, le trigger renverra un message d'erreur si :

   a. le mail de l'internaute n'existe pas
   b. le film n'existe pas
   c. l'internaute déjà donné une note pour ce film.

Tester de le trigger avec les requêtes suivantes :
insert into notation values (54, 'eeeeee@eee.fr', 4) -- ERREUR internaute n'existe pas insert into notation values (540, 'davy@bnf.fr', 4) -- ERREUR le film n'existe pas
insert into notation values (5, 'davy@bnf.fr', 4) -- ERREUR déjà noté
insert into notation values (54, 'davy@bnf.fr', 4) -– insertion correcte

5. Ecrire une fonction qui retourne la moyenne et le nombre de notes pour un film passé en paramètre (le titre du film). Erreur si l'identifiant du film n'existe pas dans la table *notation.* Retourne une ligne sous la forme ***nombre de notes / moyenne ; Exemple : 2 note(s) / 2,5***

Tester de la fonction avec les valeurs suivantes :
'La mort aux trousses' **-- retourne *2 note(s) / 4.00***
'La mort aux trous **-- erreur**

6. Ecrire une fonction qui permet de modifier le mail d'un internaute, cette modification doit être répercutée sur la table notation. Les paramètres de la fonction seront l'ancien et le nouveau mail.

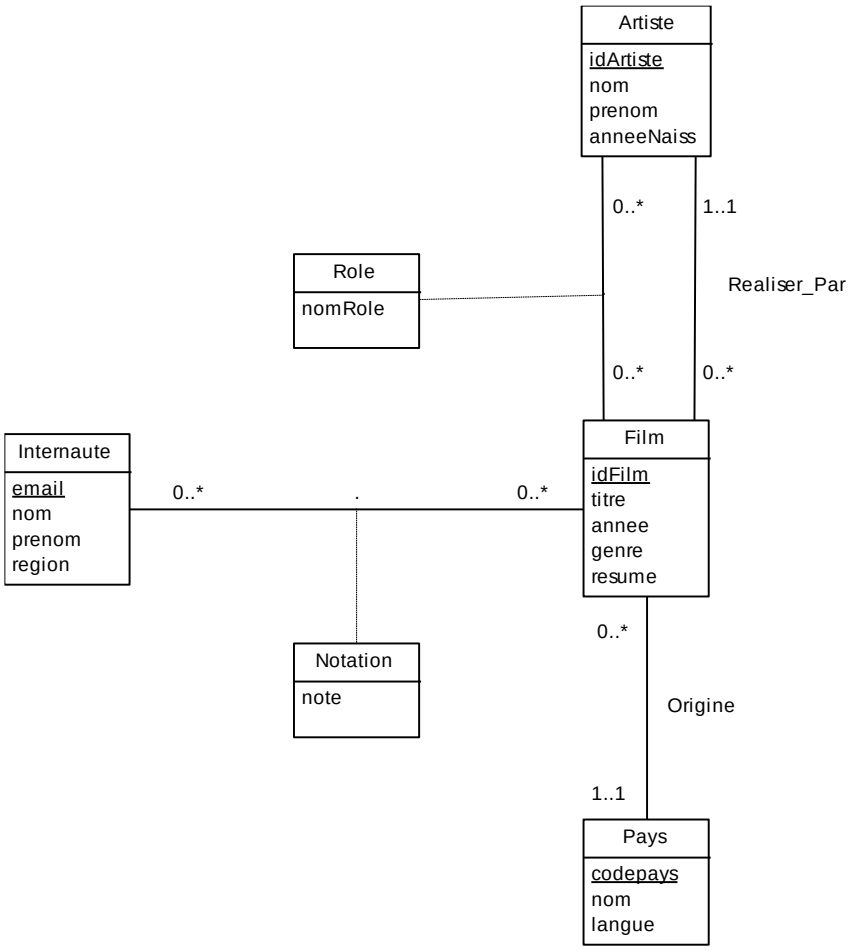
   Le principe : s'il y a une violation de la clé étrangère (FOREIGN_KEY_VIOLATION) il faut récupérer le nom de la clé étrangère qui fait référence à la table internaute, modifier le mail dans la table notation puis dans la table internaute avant de rétablir la clé étrangère.

   L'utilisation de la gestion des erreurs via les exceptions est fortement conseillée voir obligatoire.

   La requête suivante permet de récupérer le nom de la contrainte de clé étrangère de la table notation qui fait référence à la table internaute

```
SELECT  conname
FROM  pg_class PC join pg_namespace PN On PN.oid=PC.relnamespace
            join pg_constraint PCo ON PCo.conrelid=PC.oid
where nspname='tp_bd_film'
and contype='f' and relname='notation'
and confrelid=(select oid from pg_class where relname='internaute');
```

ANNEXE 1 : DIAGRAMME DE CLASSE DE LA BASE FILM



ANNEXE 2 : MPD DE LA BASE FILM