SQL et PL/pgSQL

Marie Szafranski

10 janvier 2025

Lisez *entièrement* et *attentivement* l'énoncé. Les exercices peuvent être traités de façon indépendante, tout comme la plupart des questions au sein d'un exercice : ne bloquez pas sur une question, passez à la suivante. Cependant, la progression des (questions dans les) exercices est incrémentale et peut permettre de faciliter la suite.

Documents et logiciels autorisés. Les documents sont autorisés. Les appareils électroniques personnels (téléphones inclus) sont interdits et doivent être rangés dans un sac fermé. Les traducteurs ne permettant pas de communiquer avec des tiers sont autorisés pour les élèves non francophones. L'utilisation d'un navigateur internet ou de visual studio ainsi que tout système d'IA générative sont interdits. La documentation postgresql sera accessible en ligne de commande. Néanmoins, comme annoncé à la rentrée, un document récapitulatif est une bonne idée pour être efficace.
Trigger warning : *Les élèves surpris en train de consulter internet, d'utiliser une IA générative ou d'échanger avec d'autres se verront attribuer une note de 0 et seront exclus de la salle.*

Temps imparti et barème indicatif. Vous disposez de 3h00. L'examen est composé de 3 parties :

E1. Interrogation (`SQL LMD`) [environ 1/2 des points]

E2. Fonctions et triggers (`SQL LMD`, `LDD` et `PL/pgSQL`) [entre 1/6 et 1/4 des points]

E3. Redéfinition d'un schéma (`SQL LDD`) [entre 1/6 et 1/4 des points]

Le barème sera adaptatif et tiendra compte du fait que la dernière séance sur les triggers a du être raccourcie.

Fichiers disponibles. Vous disposez sur l'espace `/pub` de deux fichiers :

F1. Le fichier `create_fill.sql` vous permettra de créer et remplir la base de données selon la procédure habituelle.

F2. Le fichier `tp_note.sql` contient la trame du rendu à effectuer pour l'ensemble des exercices.

Rendus des TP. À la fin des 3h00, vous devrez déposer, sur `http://exam.ensiie.fr`, un *unique* fichier `tp_note.sql` contenant dans l'en-tête votre nom, votre prénom et la salle, accompagné de vos réponses correctement indentées et commentées. Il y a trois dépôts :

D1. Pour les élèves dont le nom de famille est inclus dans l'ensemble {Abahamou, Geneau}, le dépôt est `cbdr11_tp_abahamou_geneau`.

D2. Pour les élèves dont le nom de famille est inclus dans l'ensemble {Gerard, Meslin}, le dépôt est `cbdr11_tp_gerard_meslin`.

D3. Pour les élèves dont le nom de famille est inclus dans l'ensemble {Michelangeli Tiga, Zuo}, le dépôt est `cbdr11_tp_michelangeli_zuo`.

Ces dépôts seront relevés à 17h15 au plus tard [1]. Un fichier non déposé à cette heure sera considéré comme non rendu. Par ailleurs, le nom respect d'une des consignes exprimées ci-dessus entraînera des points de pénalités.

Quelques notations.

Clés primaires.
- Un attribut utilisé comme clé primaire sera noté `#attribut`.
- Une clé primaire composée de `n` attributs sera notée `#attribut_1, ..., #attribut_n`.

Clés étrangères.
- Un attribut utilisé comme clé étrangère sera noté `attribut=>Reference`. Cela signifie que l'attribut nommé `attribut` est une clé étrangère qui référence la clé primaire de la relation nommée `Reference`.
- Une clé étrangère composée de *n* attributs sera notée `(attribut_1, ..., attribut_n)=>Reference`.

1. Un délai supplémentaire est accordé aux élèves bénéficiant d'un tiers temps.

Schéma de la base de données

```
                                                                                    -- schéma
t_Personne(#n_personne, nom, prenom, type_personne)
t_Film(#n_film, titre, genre, n_realisateur=>t_Personne, date_sortie)
t_Role(#n_acteur=>t_Personne, #n_film=>t Film)


                                                                                -- contraintes
(nom, prenom) UNIQUE                                                             -- t_Personne
(nom, prenom) NOT NULL                                                           -- t_Personne
t_Personne IN t_Film (1x min) OR t_Personne IN t_Role (1x min)          -- t_Personne, t_Film et t_Role
```

1 Interrogation (SQL LMD)

Consignes.

- Reportez dans le fichier `tp_note.sql` l'expression ***ainsi que*** le résultat de chaque requête.
- Dans tous les cas, les jointures classiques ***ne devront jamais*** être faites avec des `SELECT` imbriqués.

Questions préliminaires.

V1. Créez une vue `v_Real(n_personne, nom, prenom, n_film, titre)` permettant de lister l'ensemble des réalisateurs.

a) Jointure dans la clause `FROM` (SQL-92)

b) Jointure dans la clause `WHERE` (SQL-89)

→ 17+2 tuples [2]

V2. Créez une vue `v_Act(n_personne, nom, prenom, n_film, titre)` permettant de lister l'ensemble acteurs et actrices.

a) Jointure dans la clause `FROM` (SQL-92)

b) Jointure dans la clause `WHERE` (SQL-89)

→ 3+2 tuples [1].

Les vues pourront être utilisées pour définir plus facilement certaines des requêtes suivantes. Néanmoins, il est toujours possible de définir ces requêtes sans ces vues.

Requêtes.

1.1. Liste des films (`titres`) dont le réalisateur a pour nom `'Tarantino'`.

a) Jointure dans la clause `FROM` (SQL-92)

b) Jointure dans la clause `WHERE` (SQL-89)

c) Avec la vue.

→ 4 tuples

1.2. Liste des acteurs et actrices (`nom, prenom`) du film dont le titre est `'Pulp Fiction'`.

a) Jointure dans la clause `FROM` (SQL-92)

b) Jointure dans la clause `WHERE` (SQL-89)

c) Avec la vue.

→ 5 tuples

1.3. Liste des films (`titre, date_sortie`) dans lesquels l'acteur ayant pour nom `'Clooney'` a joué. La liste devra être ordonnée par ordre chronologique inverse (du plus récent au plus ancien).

→ 5 tuples

1.4. Liste des acteurs et actrices (`nom, prenom`) et nombre de films (colonne nommée `nb_film`) dans lesquels ils ont joué. La liste devra être ordonnée par ordre décroissant sur le nombre de films puis par ordre croissant sur le nom.

→ 1 nom dans 5 films, 2 noms dans 4 films, 3 noms dans 3 films, 6 noms dans 2 films et 7 noms dans 1 film.

2. Voir l'énoncé de l'exercice 2 pour vérifier vos réponses.

1.5. Liste des réalisateurs (`nom, prénom`) et nombre de films (colonne nommée `nb_film`) pour les réalisateurs ayant réalisés plus de 3 films.

→ 2 tuples

1.6. Liste des films (`titre`) dans lesquels l'acteur ayant pour nom `'Clooney'` a joué mais pas l'actrice ayant pour nom `'Zeta-Jones'`.

→ 3 tuples

1.7. Liste des réalisateurs (`nom, prénom`) qui sont aussi acteurs. La requête pourra être réalisée avec un opérateur ensembliste.

→ 2 tuples [1].

1.8. Liste des acteurs et actrices (`n_personne, nom, prenom`) qui ne sont pas réalisateurs. La requête devra être exprimée avec l'opérateur `(NOT) IN` ou `(NOT) EXISTS`.

→ 17 tuples [1].

1.9. Liste des réalisateurs (`n_personne, nom, prenom`) qui ne sont pas acteurs. La requête devra être exprimée avec une jointure externe `(LEFT/RIGHT) OUTER JOIN`.

→ 3 tuples [1].

2 Fonctions et triggers (SQL LMD, LDD et PL/pgSQL)

2.1. Ajoutez une contrainte à la relation `t_Personne` pour faire en sorte que l'attribut `type_personne` soit égal à 1 (acteur ou actrice), 2 (réalisateur) ou 3 (acteur et réalisateur).

2.2. a) Donnez l'instruction permettant d'initialiser, pour la personne dont le numéro est 1 (`n_personne` = 1), l'attribut `type_personne` à 1.

b) Décrivez ce qu'il se passera si on initialise, pour la personne dont le numéro est 1 (`n_personne` = 1), l'attribut `type_personne` à 4.

c) Décrivez ce qu'il se passera si on initialise l'attribut `type_personne` lorsque `n_personne` n'est pas dans la base de données.

2.3. Créez une fonction qui, étant donnés un numéro de personne et un type, initialise le type d'une personne.

2.4. Créez une fonction qui permet d'initialiser l'attribut `type_personne` pour toutes les personnes présentes dans la base de données. On pourra utiliser les requêtes 1.7, 1.8 et 1.9 des questions précédentes pour créer les curseurs correspondants. Si ces requêtes n'ont pas été faites, on pourra faire une requête générale et effectuer des tests sur le numéro de personne dans le parcours de la requête :
— `n_personne` $\in [1,17]$: `type_personne` = 1,
— `n_personne` $\in [18,20]$: `type_personne` = 2,
— `n_personne` $\in [21,22]$: `type_personne` = 3.
On pourra appeler la fonction créée à la question précédente.

2.5. Créez un trigger, qui lorsqu'un nouveau film est inséré, vérifie que l'attribut `type_personne` du réalisateur vaut bien 2 ou 3. Pour tester votre trigger :

a) Insérez le tuple (16, 'A star is born', 'drame', 2) dans `t_film`.

b) Insérez le tuple (16, 'A star is born', 'drame', 22) dans `t_film`.

3 Modification du schéma (SQL LDD)

Reprenez le schéma de la base de données en page 2.

3.1. Que veut dire la dernière contrainte du schéma ? Donnez une explication.

3.2. Sans modfier le schéma existant, donnez les commandes permettant d'implanter :

a) La contrainte d'unicité sur la table `t_Personne.`

b) La contrainte de non nullité sur la table `t_Personne.`

3.3. Avec les deux contraintes définies précédemment, que représente un couple `(nom, prenom)` ?

3.4. Sans modifier le schéma existant, donnez les commandes permettant :

a) D'ajouter les attributs `nom_real` et `prenom_real` à la table `t_Film`.

b) D'ajouter la contrainte référentielle `(nom_real, prenom_real)=>t_Personne`.

Remarque. Si lors des différents essais vous avez modifié le schéma de la base de données avant de finir les exercices 1 et 2 , pensez à rétablir si nécessaire la base de données dans l'état intial avant de reprendre ces exercices.