TP PostgreSQL n°3

- Requêtes avancées et transactions -

Il est conseillé de travailler le plus possible avec des fichiers sql plutôt qu'avec l'éditeur de PostgreSQL .

PostgreSQL et psql

Connectez-vous à la base de données (tp_intro ou autre bdd déjà existante).

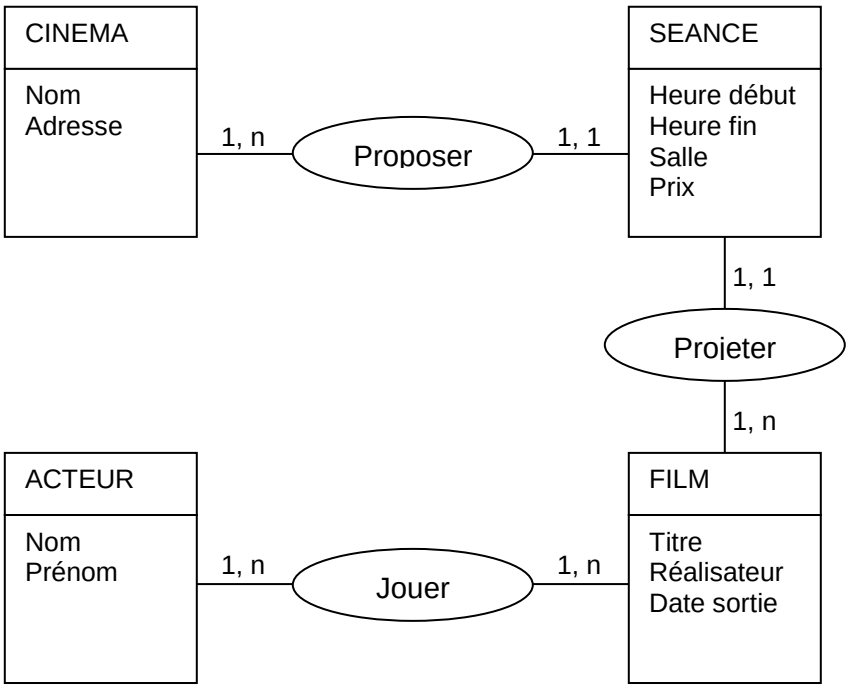
Commandes de base. Les principales commandes de psql sont les suivantes :
– \? : Liste des commandes psql,
– \h : liste des instructions SQL,
– \h <une_instruction> : description de l'instruction SQL <une_instruction>.
Essayez avec l'instruction CREATE TABLE,

– \d : liste des tables (ou relations, cf. partie 2),
– \d <nom_table> : description de la table <nom_table>,
– \i <chemin/nom_fichier_script.sql> : exécution d'un fichier de script SQL,
– \o <chemin/nom_fichier_resultat.sql> : écriture des résultats des instructions dans le fichier passé en paramètre,
– \o : retour à un affichage à l'écran,.
– \q : quitter psql.

Retenez ces commandes, elles seront indispensables pour les TP.

1°) Modèle Entité/Association

Le fichier cinema.sql contient la définition ainsi que des données de test d'une base de données. Cette dernière a été conçue à partir du modèle Entité – Associations suivant :



Charger le fichier /users/prof/fernandes/data/cinema.sql contenant la description des tables et les données. Noter la présence d'une cinquième table permettant de faire la gestion de la relation N-N entre les entités **Acteur** et **Film**.

Les tables suivantes sont donc créées :

Cinema (idcine, nom, adresse)
Film (idfilm, titre, realisateur, datesortie)
Seance (idseance, heuredbt, heurefin, numsalle, ***film***,***cine***,prix)
Acteur (idacteur, nom, prenom)
Casting (idacteur, idfilm, personnage)

Quelle est la commande permettant de voir la liste des tables créées ?

Ex Oracle : En lançant la requête suivante, vous pouvez vérifier que les tables ont bien été créées dans le catalogue : `SELECT * FROM cat`

Quelle est la commande permettant de vérifier la présence des contraintes dans une table ?

Exemple Oracle : La commande suivante permet de vérifier la présence de contraintes sur les tables :

```
SELECT constraint_name, constraint_type, table_namre FROM user_constraints;
```

2°) Requêtes partie 1

Quelles requêtes permettent de répondre aux questions suivantes ? Sauvegarder l'ensemble des requêtes dans un fichier tp3-1.sql

1. Afficher toutes les informations propres aux cinémas
2. Afficher le titre de tous les films
3. Afficher le titre de tous les films par ordre alphabétique
4. Afficher le titre des films sortis en 1995 ou après
5. Afficher le titre des films sortis entre 1980 et 1995 par ordre alphabétique inverse
6. Affiche le titre des films, et leur date de sortie, pour les films antérieur à 1999 et dont le titre contient la chaine 'in'. Renommer le titre de la colonne datesortie en Sortie (opérateur AS).
7. Afficher le nom et le prénom de tous les acteurs
8. Afficher le nom des acteurs et des cinémas
9. Afficher le nom et le prénom des acteurs dont le nom commence par D
10. Afficher le nom (en majuscule) des acteurs et l'initiale de leur prénom suivie d'un point, c'est à dire sous le format NOM P. (opérateurs UPPER et CONCAT). Renommer le nom de la colonne en Nom
11. Afficher les heures de début et de fin de chaque séance sous le format hh-mm
12. Afficher les heures de début et de fin de chaque séance commençant à ou après 19h00 sous le format hh-mm
13. Afficher les heures de début et de fin de chaque séance commençant à ou après 19h00 sous le format hh-mm, en renommant les colonnes Debut et Fin
14.

2°) Requêtes partie 2

15. Afficher les heures de début et de fin de chaque séance ainsi que leur durée en minutes, pour les séances commençant à ou après 21h00. Les heures doivent etre affichées sous le format hh-mm, en renommant les colonnes en Debut,Fin et DureeMn
16. Afficher les horaires des séances de la première salle du premier cinéma
17. Afficher le prix moyen d'une séance de cinéma
18. Afficher, pour chaque personnage, le nombre de fois qu'il est joué et son titre
19. Afficher le personnage et le nombre de fois qu'il est joué si ce nombre est au moins égal à deux
20. Afficher, pour chaque prénom, le nombre d'acteurs portant ce prénom
21. Afficher l'heure de fin de la séance la plus tardive
22. Afficher l'heure de début et de fin des séances commençant le plus tôt
23. Afficher les horaires des séances qui projettent le film Impitoyable
24. Afficher le titre des films programmés
25. Afficher le titre des films programmés avec les horaires des séances prévues
26. Afficher le titre des films programmés (sans doublons) par ordre alphabétique sur le titre
27. Afficher le titre des films non programmés (opérateur minus)
28. Afficher le nom des cinémas et les séances qu'ils proposent
29. Afficher pour chaque prénom présent au moins deux fois chez les acteurs, le nom et le prénom de ces acteurs
30. Afficher le nom des cinémas, les horaires des séances ainsi que le titre du film proposé à cette séance
31. Afficher le nom des cinémas, les horaires des séances sous le bon format, où passe le film intitulé 'Impitoyable', quelle que soit la casse du titre du film
32. Même chose que précédemment en ordonnant la sélection sur la date de début des séances.
33. Afficher les titres des films dont une séance programmée dure moins de 2h00 et afficher aussi cette durée en minutes
34. Afficher le nom et prénom des acteurs jouant dans chaque film ainsi que le titre du film

3°) Modification de tables et données.

Sauvegarder les commandes à venir dans le fichier tp3-2.sql.

- Modifier la table Seance pour lui rajouter une contrainte de type check (contrainte pertinente que vous proposez)
- Modifier la table Film en lui ajoutant l'attribut duree. Mettre à jour cet attribut pour les données existantes, en faisant un calcul simple.

4°) Introduction aux transactions

Une transaction est une séquence de commandes SQL considérée comme unitaire et indivisible.

Une transaction commence avec la première commande exécutable qui suit un COMMIT, un ROLLBACK, ou la connexion à la base de données.

Une transaction se termine avec la commande COMMIT, ROLLBACK ou lors d'une déconnexion.

Quelle commande permet de valider une transaction ?

Ex Oracle : Utiliser la commande COMMIT WORK permet de valider la transaction courante. Attention, on ne peut plus revenir en arrière.

Utiliser la commande SAVEPOINT <nom> à 'intérieur d'une transaction permet de créer un point de retour éventuel. Le retour à ce point s'effectue avec la commande ROLLBACK WORK TO SAVEPOINT <nom>.

Utiliser la commande ROLLBACK sans autre paramètre termine la transaction et annule toutes les modifications ayant été faites durant la transaction.

Réaliser l'ensemble des questions suivantes directement sous l'éditeur PostgreSQL. L'objectif étant de visualiser l'effet d'une transaction.

- Créer la table Categorie2 (idcat, agemini, intitule)
- Créer un point de sauvegarde : SAVEPOINT tuplecat;
- Créer deux tuples de la table Categorie2 puis vérifier que ces tuples existent.
- Taper ROLLBACK WORK TO SAVEPOINT tuplecat;
- Selectionner tous les tuples de la table Categorie2. Quelles sont vos conclusions sur le résultat obtenu ?