Ma (première) DB

Récap

Etape 1 (Inspiration) :

Nous avons choisi de créer une base de données pour gérer les informations sur les films, les acteurs, les réalisateurs, les genres, les pays, les utilisateurs, les commentaires et les notes. Nous avons déterminé les relations entre les tables, ainsi que les cardinalités.

Voir filminforating_diagram.pdf

Etape 2 (Création) :

Ensuite, nous avons utilisé Flyway pour gérer les migrations de la base de données. Nous avons créé un fichier de migration SQL pour créer la base de données et les tables.

1. Installation de Java (si pas déjà fait)
2. Installation de Flyway

Décompressez l'archive téléchargée dans un répertoire de votre choix. Ajoutez ensuite le répertoire flyway-x.x.x/bin (où x.x.x est la version de Flyway pour nous 9.16.0) à la variable d'environnement PATH de votre système, redémarrer le pc pour appliquer la variable d'environnement.

3. Configuration de Flyway :

Voir Flyway/flyway.conf

4. Création de migration :

Voir dans dossier sql

5. Exécution des migrations :

cmd à exécuter dans le dossier Flyway -> flyway migrate

Etape 3 (Alimentation) :

Nous avons utilisé Faker.js pour générer des données aléatoires et les insérer dans les tables de la base de données. Nous avons créé un script Node.js pour insérer au moins 400 000 enregistrements dans les tables de la base de données.

Voir le Dossier Faker.js

cmd à exécuter dans le dossier Faker.js -> node seed.js

Etape 4 (Exploitation) :

Nous avons créé deux vues en SQL :

- Top 10 des films :

```
CREATE VIEW top_10_films AS
SELECT f.id, f.titre, AVG(n.valeur) AS moyenne_notes
FROM Film f
JOIN Note n ON f.id = n.film_id
GROUP BY f.id, f.titre
ORDER BY moyenne_notes DESC
LIMIT 10;
```

- Afficher le détails des films : réalisateurs, acteurs et les genres :

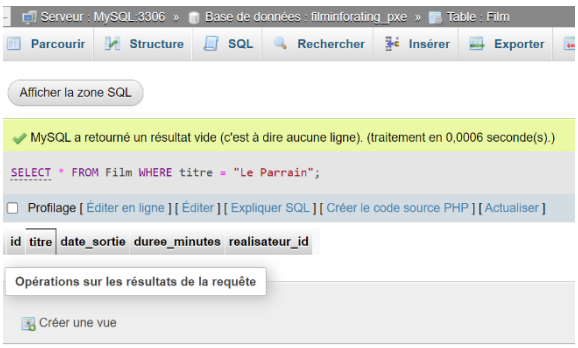
```
CREATE VIEW film_details AS
SELECT f.id, f.titre, f.date_sortie, f.duree_minutes,
    r.nom AS realisateur_nom, r.prenom AS realisateur_prenom,
    GROUP_CONCAT(g.nom) AS genres
FROM Film f
JOIN Realisateur r ON f.realisateur_id = r.id
JOIN Film_Genre fg ON f.id = fg.film_id
JOIN Genre g ON fg.genre_id = g.id
GROUP BY f.id, f.titre, f.date_sortie, f.duree_minutes, r.nom, r.prenom;
```

Nous avons créé un index avec une requête avant et après la création de l'index :

- Avant d'ajouter un index :
  SELECT * FROM Film WHERE titre = "Le Parrain";

- Ajouter l'index :
  ALTER TABLE Film ADD INDEX (titre);

- Après avoir ajouté l'index :
  SELECT * FROM Film WHERE titre = "Le Parrain";



Etape 5 (Procédure) :

Nous avons créé une procédure stockée en SQL pour calculer le nombre total d'acteurs et de réalisateurs dans la base de données. La procédure prend deux paramètres de sortie et renvoie le nombre total d'enregistrements dans les tables Acteur et Réalisateur.

Création :

```
DELIMITER //

CREATE PROCEDURE total_acteurs_realisateurs (OUT total_acteurs INT, OUT total_realisateurs INT)

BEGIN

  SELECT COUNT(*) INTO total_acteurs FROM Acteur;

  SELECT COUNT(*) INTO total_realisateurs FROM Realisateur;

END //

DELIMITER ;
```

Cette procédure stockée prend deux paramètres de sortie, total_acteurs et total_realisateurs, qui stockent le nombre total d'acteurs et de réalisateurs respectivement. Nous avons réalisé un décompte des enregistrements dans les tables Acteur et Realisateur et les avons stockés dans les paramètres de sortie. Pour utiliser cette procédure stockée, nous pouvons exécuter la requête suivante pour appeler la procédure et récupérer les résultats :

```
CALL total_acteurs_realisateurs(@total_acteurs, @total_realisateurs);

SELECT @total_acteurs AS Total_Acteurs, @total_realisateurs AS Total_Realisateurs;
```