Rapport de projet

Alexandre Guerassimov, Christian Qian

27 mai 2020

Résumé

Des bases de données relationnelles complexes sont utilisées pour le stockage et la gestion de données dans de nombreuses applications. Le but de ce projet est de concevoir et de mettre en place un modèle relationnel à partir de données réelles du domaine du cinéma.

1 Présentation

Pour ce projet, nous avons choisi pour contexte la gestion des données des utilisateurs et des films pour un site web de vidéo à la demande (VOD). Ce site devra donc gérer des utilisateurs avec leurs droits et des films avec leurs évaluations. Les droits dépendront de la souscription ou non d'un abonnement, et du type d'abonnement souscrit. Il faudra également gérer les différentes connexions dont le nombre simultané peut varier selon les droits accordés.

2 Modèle relationnel

2.1 Schéma relationnel

FILMS(id_film, titre, date, adulte, nb_note, moy_note)

TAGS(id_util*, id_film*, tag, horodatage)

GENRES(genre)

APPARTIENT(id_film*, genre*)

UTILISATEURS(id_util, nom, prenom, date_nais, pseudo, mdp)

NOTE(id_film*, id_util*, note, horodatage)

VISIONNER(id_film*, clef*)

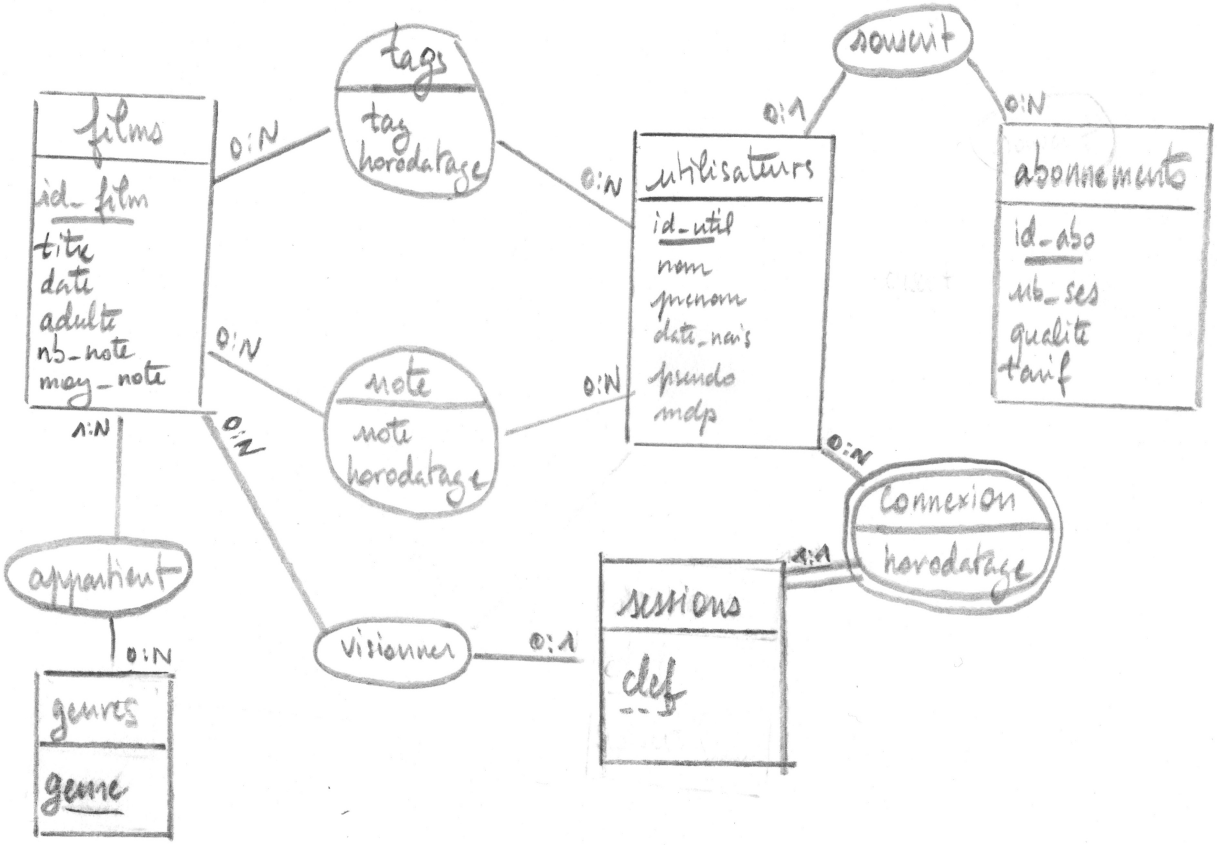
ABONNEMENTS(id_abo, qualite, nb_ses, tarif)

SOUSCRIT(id_util*, id_abo*)

SESSIONS(clef)

CONNEXION(id_util*, clef*, horodatage)

2.2 Diagramme entité-association



3 Contraintes et gestion

Chaque identifiant (id) doit être unique — ce qui va de soi pour une clef primaire — et 'NOT NULL' ; pour toute nouvelle ligne, l'identifiant est généré automatiquement par auto-incrémentation à l'aide de 'SERIAL'.

Pour la suppression d'une ligne de la table FILMS ou de la table UTILISATEURS, il faut s'assurer que tout ce qui concerne cette ligne dans les autres tables soit supprimé également : un 'ON DELETE CASCADE' pour les clefs étrangères gère ce cas.

3.1 Contraintes par table

FILMS Tout film doit avoir un titre, un nombre de notes, une moyenne des notes et une catégorie (adulte ou non) : ces attributs doivent donc être 'NOT NULL' également. Il ne peut y avoir ni de moyenne négative, ni de nombre de notes négatif et la moyenne ne doit pas dépasser 10 : un 'CHECK' assure le respect de cette règle.

TAGS L'attribut tag doit être 'NOT NULL' pour que la ligne ait un intérêt ; l'horodatage est dispensable par contre.

GENRES Le genre est comme un identifiant, mais étant une chaîne de caractères, il n'y a pas d'incrémentations, simplement des éléments uniques.

UTILISATEURS Les noms, prénoms, pseudos et mots de passe doivent être présents et seront donc 'NOT NULL'. La date de naissance doit être présente également pour savoir si l'utilisateur peut visionner des films adultes ou non, et cette date doit aussi être réaliste — nous pouvons supposer que l'utilisateur ne peut pas être né il y a deux siècles : l'attribut est donc 'NOT NULL' avec un 'CHECK' pour nous assurer de la vraisemblance de la donnée. Le pseudo doit être 'UNIQUE' pour pouvoir différencier les utilisateurs. Le pseudo et le mot de passe doivent avoir une taille minimum : un 'CHECK' gérera cela.

NOTE La présence d'une ligne sur cette table sans la note n'aurait pas de sens : un 'NOT NULL' s'impose donc pour cet attribut. La note doit aussi être strictement positive, sans dépasser 10, ce qu'un 'CHECK' assure. L'horodatage est par contre dispensable.

ABONNEMENTS Pour cette table, tous les attributs sont indispensables pour le bon fonctionnement de l'application : tous les attributs sont donc 'NOT NULL'. De plus, la valeur de la qualité doit être parmi un ensemble fini de valeurs ; le nombre de sessions et le tarif doivent être positifs : toutes ces contraintes seront gérées par des 'CHECK'.

SOUSCRIT Puisqu'un utilisateur ne doit pouvoir s'abonner qu'à un seul abonnement, un 'UNIQUE' s'impose pour l'identifiant de l'utilisateur.

LOGS Toutes les tables servant d'archives sont celles dont le nom commence par LOG_. Les contraintes sont souvent les mêmes pour chacune d'entre elles : un 'NOT NULL' pour tous les horodatages avec un 'DEFAULT CURRENT_TIMESTAMP'. À cela, nous avons ajouté une singularité chacune pour 'LOG_FILMS' et 'LOG_SOUSCRIT' : un 'NOT NULL' pour l'attribut titre du premier et un 'UNIQUE'

3.2 Gestion des données

3.2.1 Triggers

Notes Pour chaque ajout ou modification d'une note pour un film, il faut que le nombre de notes de ce film, avec la moyenne, changent. Les triggers 'ajout_maj_note_films' et 'effacer_note_films' s'occupent de cela.

Sessions Avant d'ajouter une session, il faut vérifier que l'abonnement souscrit le permet et donc vérifier le nombre de sessions déjà existantes. Le trigger 'nb_sessions' s'occupe de ça.

Archivage Après qu'une ligne de la table VISIONNER se fait effacer, elle devra être archivée dans une table à part dédiée. Il en va de même pour les tables FILMS, SOUSCRIT et CONNEXION. Ce sont les triggers 'maj_log_connexion', 'maj_log_souscrit', 'maj_log_films' et 'maj_log_visionner' qui s'occupent de ça.

3.2.2 Fonctions

Créer son compte : ajout_util() Prend en paramètres le nom, le prénom, la date de naissance, le pseudo et le mot de passe pour créer un nouveau compte.

Supprimer son compte : eff_util() Prend l'identifiant de l'utilisateur en paramètre et retrouve les sessions correspondantes à un utilisateur, leur suppression entraîne l'effacement des lignes de la table VISIONNER et CONNEXION (par cascade). On supprime ensuite l'utilisateur qui efface les lignes des tables TAGS, NOTE, SOUSCRIT (par cascade).

Souscrire à un abonnement : ajout_abo() Prend l'identifiant de l'utilisateur et de l'abonnement en paramètres, pour permettre à un utilisateur de souscrite à un nouvel abonnement.

Supprimer un abonnement : eff_abo() Prend l'identifiant de l'utilisateur en paramètre pour pouvoir supprimer sa souscription.

Connexion : ajout_connexion() Prend en paramètre un pseudo et un mot de passe pour permettre à un utilisateur de se connecter sur différentes machines dans la limite de son abonnement.

Déconnexion : eff_connexion() Prend en paramètre la clef de connexion pour pouvoir supprimer les lignes des tables SESSIONS et CONNEXION correspondantes.

Ajouter un tag : ajout_tag() Prend en paramètre les identifiants de l'utilisateur et du film avec le tag à ajouter au film. Ajoute un tag à un film pour faciliter la recherche d'un film par un mot-clé.

Noter un film : ajout_note() Prend en paramètre l'identifiant de l'utilisateur et du film avec la note à attribuer.

Démarrer un visionnage : ajout_visionner() Prend en paramètre l'identifiant d'un film et une clef de connexion et fait une insertion dans la table VISIONNER.

Ajouter un film : ajout_film() Prend en paramètre un titre, une année, un booléen indiquant si c'est un film adulte ou non et fait une insertion dans la table FILMS.

Générer une clef de connexion : clef_generator Renvoie une clef.

Arrêter un visionnage : eff_visionner() Prend en paramètre une clef de connexion pour effacer la ligne correspondante dans la table VISIONNER.

Mettre à jour sa souscription : maj_abo() Prend en paramètre l'identifiant d'un utilisateur et d'un abonnement pour remplacer l'identifiant de l'abonnement qui lui était déjà associé par un nouveau dans la table SOUSCRIT.

Mettre à jour une note : maj_note() Prend en paramètre l'identifiant d'un film et d'un utilisateur avec une note pour remplacer la note qu'avait mis l'utilisateur par une nouvelle dans NOTE.

Rechercher un film par genre ou note : recherche_films_genre_note() Prend en paramètre au moins un attribut parmi un genre, une note ou une année pour renvoyer une liste pertinente de films.

Obtenir l'historique d'un utilisateur : get_historique() Prend en paramètre le pseudo d'un utilisateur pour renvoyer la liste des films qu'il a visionné dans l'ordre décroissant par rapport à la date.

Rechercher un film par tag : recherche_films_tag() Prend en paramètre une chaîne de caractères pouvant contenir une partie d'un tag ou le tag entier avec laquelle faire une recherche parmi les films possédant au moins un tag.

Rechercher les films les mieux notés : meilleurs_films() Ne prend rien en paramètre et renvoie une liste des films en ordre décroissant par rapport aux moyennes et aux nombres de notes.

3.2.3 Index

Clefs Les clefs primaires — comme les clefs étrangères — se voient attribuées un index chacune automatiquement par PostgreSQL. Par défaut, ceux-ci sont des 'BTree' et nous les avons laissés tels quels.

Titre Les titres se trouvant dans la table 'FILMS' sont des valeurs que l'on pourrait souvent utiliser pour des requêtes avec des comparaisons ponctuelles sur l'attribut. Nous avons donc décidé d'ajouter un 'Hash' à l'attribut ; titre étant une chaîne de caractères, nous ne pouvons pas faire de comparaisons autres que l'égalité. Nous espérons ainsi gagner en performances pour les recherches de films se faisant avec le titre.

Tags De même que pour les titres, les tags peuvent bénéficier d'un hash pour les mêmes raisons.

4 Sources des données

https ://grouplens.org/datasets/movielens/

http ://webia.lip6.fr/~guigue/film_v2.pkl

http ://webia.lip6.fr/~guigue/act_v2.pkl

http ://webia.lip6.fr/~guigue/crew_v2.pkl

https ://www.kaggle.com/ashirwadsangwan/imdb-dataset