NoiseBook

Rapport de projet de Base de Données

BECHERRA Mattéo – 22 00 61 56
DESMAZIERES Adèle – 21 97 90 92

Introduction

Nous avons modélisé et implémenté la base de donnée d'un réseau social de partage de musiques et concerts, appelé Noisebook. Dans ce rapport, nous expliquons les détails de notre implémentation et les choix que nous avons fait pour respecter au mieux les contraintes de l'énoncé.

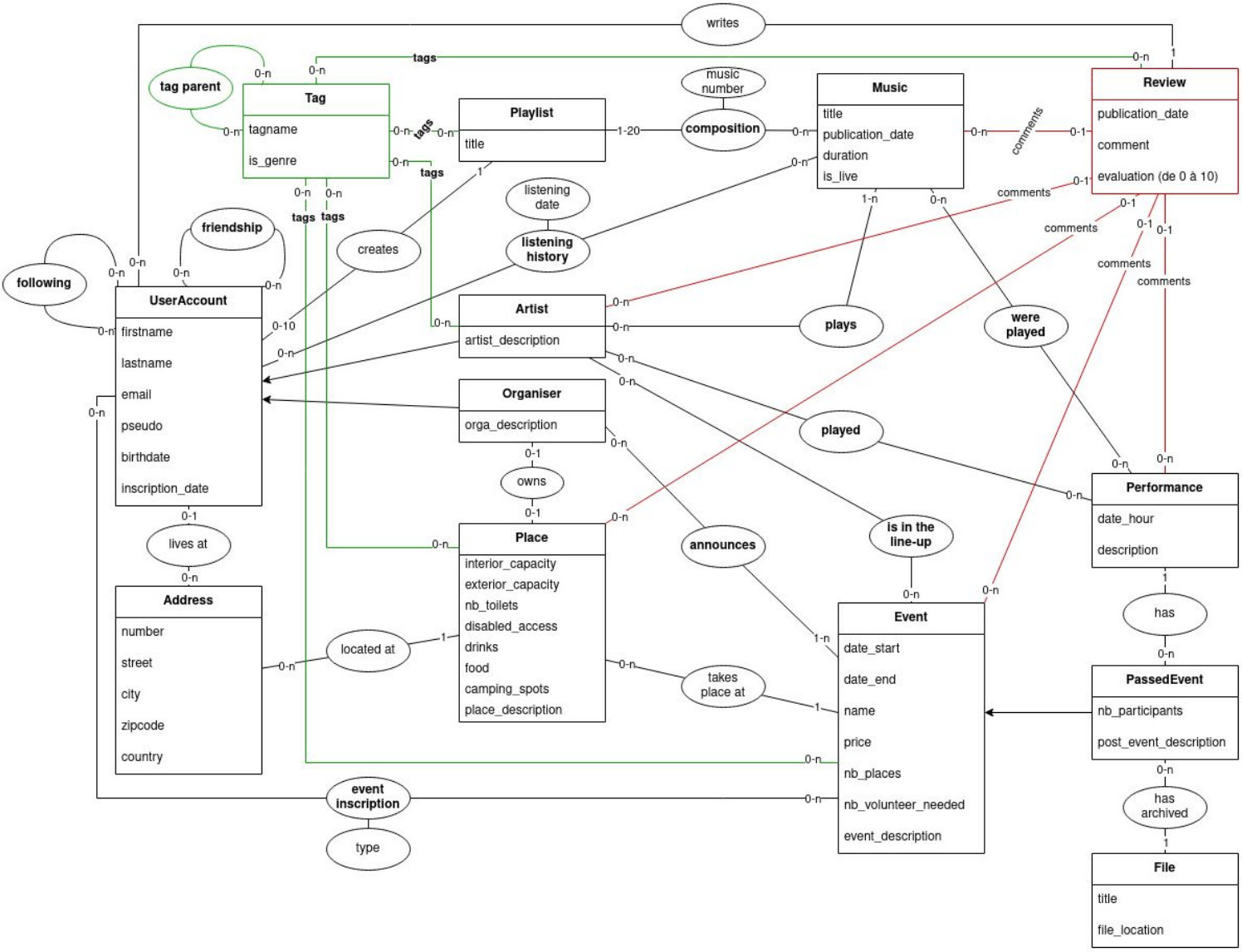
Pour générer et remplir les tables, dans psql tapper : `\i make-tables.sql`

Pour exécuter le script de requêtes, dans psql tapper : `\i requetes.sql`

Modèle relationnel

Address(address_id,house_number,street,city,zipcode,country)
Place(place_id,address_id, place_name, interior_capacity, exterior_capacity, nb_toilets, disabled_access, drinks, food, camping_spots, place_description)
UserAccount(user_id, firstname, lastname, email, pseudo, birthdate, inscription_date, address_id)
Artist(artist_id, artist_description)
Organiser(orga_id, place_id, orga_description)
Event(event_id,date_start, date_end, name, price, nb_places, nb_volunteer_needed, event_description)
PassedEvent(event_id, nb_participants, post_event_description)
File(file_id,event_id, title, file_location)
Performance(perf_id,event_id, date_hour, perf_description)
Music(music_id,title, duration, publication_date, is_live)
Playlist(playlist_id,user_id, title)
Review(review_id,publication_date, comment, evaluation, commented_id, commented_type)
Tag(tag_id, tagname, is_genre)
Following(follower, followed)
Friendship(friend1, friend2)
EventAnnouncement(orga_id, event_id)
EventInscription(user_id, event_id, type)
Lineup(artist_id, event_id)
PerfAuthor(artist_id, perf_id)
PerfMusic(music_id, perf_id)
MusicAuthor(music_id, artist_id)
PlaylistComposition(playlist_id,music_number, music_id)
ListeningHistory(music_id, user_id, listening_date, duration)
TagParent(tparent, tchild)
Tagging(tag_id, tagged_id, tagged_type)

Diagramme E/R



Précisions sur notre modélisation

UserAccount

C'est un compte associé à une unique personne. Si c'est un compte artiste ou organisateur, il doit tout de même y avoir une seule personne qui représente le compte, définie par son nom, prénom et mail, les autres membres pouvant être décrits dans la description. Quel que soit le type de compte, chaque utilisateur peut avoir un pseudo, qui sera affiché à la place de son nom et prénom. Un compte peut être User, ou User+Artist, ou User+Organiser, ou User+Artist+Organiser.

- following : Un utilisateur peut suivre N autres utilisateurs, et être suivi par M utilisateurs.
- friendship : Deux utilisateurs peuvent être amis entre eux. Un utilisateur peut être ami avec n'importe quel nombre d'utilisateurs.

Artist

C'est le compte d'une ou plusieurs personnes produisant de la musique (dont une qui représente le groupe d'artistes). La description permet de détailler la composition du groupe par exemple.

Organiser

C'est le compte d'organisateurs de concerts ou évènements musicaux en live. La description permet de préciser si c'est une association, ou autre, et quel est l'objectif de l'association par exemple. Il peut posséder une Place ou non.

Place

C'est un lieu où des évènements musicaux peuvent être organisés. Il peut être affilié à un compte Organiser ou non. Un Organiser peut être affilié à au plus une Place. Un lieu de concert peut avoir différentes caractéristiques si elle sont précisées.

Address

C'est une adresse postale, correspondant à l'adresse d'un lieu de concert ou l'adresse d'un utilisateur.

Event

C'est un évènement musical, un concert ou un festival par exemple. Il peut être organisé par un ou plusieurs organisateurs. Il peut avoir une date de début et une date de fin, un nom, un prix, un nombre de places disponibles, un nombre de volontaires recherchés et une description éventuellement. Il a lieu à une Place et a un lineup des artistes qui sont censés y jouer. Les utilisateurs peuvent s'inscrire à des évènements en tant que spectateur ou volontaire.

PassedEvent

Cela recense les informations d'un évènement musical passé, comme le nombre réel de participants qui sont venus, et les fichiers d'images ou de vidéos qui ont été filmés durant le concert. Ces fichiers sont représentés par l'entité File. Les évènements passés contiennent également la liste des performances des artistes.

Performance

Ce sont les musiques jouées par un ou plusieurs artistes à un moment précis du concert. Ainsi les utilisateurs peuvent commenter les performances de chaque artiste au sein des concerts.

Music

Ce sont les morceaux de musiques publiés par les artistes. Ils ont un titre, une durée, une date de publication et un attribut définissant si c'est un enregistrement live. Les musiques peuvent être écoutées par les utilisateurs, ce qui est enregistré dans leur historique d'écoute avec la date d'écoute.

Playlist

Une playlist peut contenir une à vingt musiques, et chaque utilisateur peut en créer zéro à dix. Elle doit avoir un titre. Les musiques d'une playlist sont numérotées pour être ordonnées. Elle peut contenir plusieurs fois la même musique.

Review

Une review contient éventuellement une note de 0 à 10, et/ou un commentaire rédigé, et une date de publication. Elle est écrite par un utilisateur. Il commente exactement un objet parmi une Performance, un Event, une Place, un Artist ou une Music. Une review peut également avoir un certain nombre de Tags associés.

Tag

Les Tags sont des mots-clefs qui labélisent des entités du logiciel. Un tag a un nom et un attribut définissant si c'est un genre musical. De plus chaque tag peut avoir des tags parents, et un tag parent peut avoir n'importe quel nombre de descendants. Les tags peuvent être appliqués aux Review, Playlist, Artist, Place et Event. Un tag appliqué directement à une entité non Review représente un tag "officiel" que l'entité s'est elle-même attribuée (par les Artists ou par les Organisateurs possédants des Places) ou que son auteur lui a attribué (par le créateur de la Playlist, ou par l'Organisateur de l'Event taggé). L'auteur d'une review peut lui associer des tags pour compléter des tags officiels qu'il juge incomplet par exemple.

Requêtes

Nous avons proposé plusieurs requêtes à faire sur nos données, qui sont détaillées dans le script "requetes.sql".

Indice de recommandation

Notre indice personnalisé de recommandation de concerts pourrait s'appuyer sur plusieurs critères, du plus important au moins important :

- recommander des concerts dans des pays en lien avec celui de l'utilisateur (cf. la requête qui renvoie les concerts qui ont lieu dans le même pays que n'importe quel autre concert que l'utilisateur a déjà vu)
- recommander des concerts dont le lineup contient des artistes écoutés et likés par l'utilisateur
- recommander des concerts dont les genres musicaux sont écoutés et likés par l'utilisateur.

Pour implémenter cela, nous pourrions utiliser un système de score associé à chaque évènement en fonction du niveau auquel il remplit chaque critère. Ensuite on ferait la somme pondérée des critères, et on recommanderait les évènements les mieux notés pour tel utilisateur à tel moment.

Problèmes et améliorations possibles

Liaisons 1-n to 0-n

Nous avons implémenté les contraintes de liaisons 1-n to 0-n en 0-n to 0-n. Mais cela pourrait être fait avec des triggers.

Conservation des données

Comme nous avons utilisé abondamment les contraintes ON DELETE CASCADE, si un Organiser supprime son compte cela effacerait les Event organisés uniquement par lui, ainsi que les informations archivés dans PassedEvent liées à ces Event, y compris les File qui deviennent inutilisés. Or ce genre de logiciel peut voir un besoin d'archiver toutes les données passées au lieu de les effacer, donc on pourrait créer des date d'effacement à chaque entité par exemple pour conserver le maximum de données.

Pondération des tags

On aurait pu pondérer les liens de parentés entre les tags pour représenter des proximités différentes entre tags.

Extension possibles

Nous aurions aimé implémenter une fonctionnalité de création automatique de playlists, regroupant des musiques d'un même concert par exemple, ou des musiques d'un même genre musical.

De plus nous aurions pu implémenter le principe de publication de posts par les utilisateurs, en rendant facultatif l'association des review avec une entité du réseau. Ainsi pour publier un post, un utilisateur pourrait publier une review sans entité liée. Cependant nous avons choisi de ne pas le faire, pour garder notre réseau social proche du style de Spotify.

Génération des données

Les données pour remplir les tables ont été générées avec le site Mockaroo, ainsi qu'avec des datasets du site Kaggle. Nous avons utilisé le site Musicmap pour avoir des genres et sous-genres musicaux cohérents.

La hiérarchie des genres et sous-genres