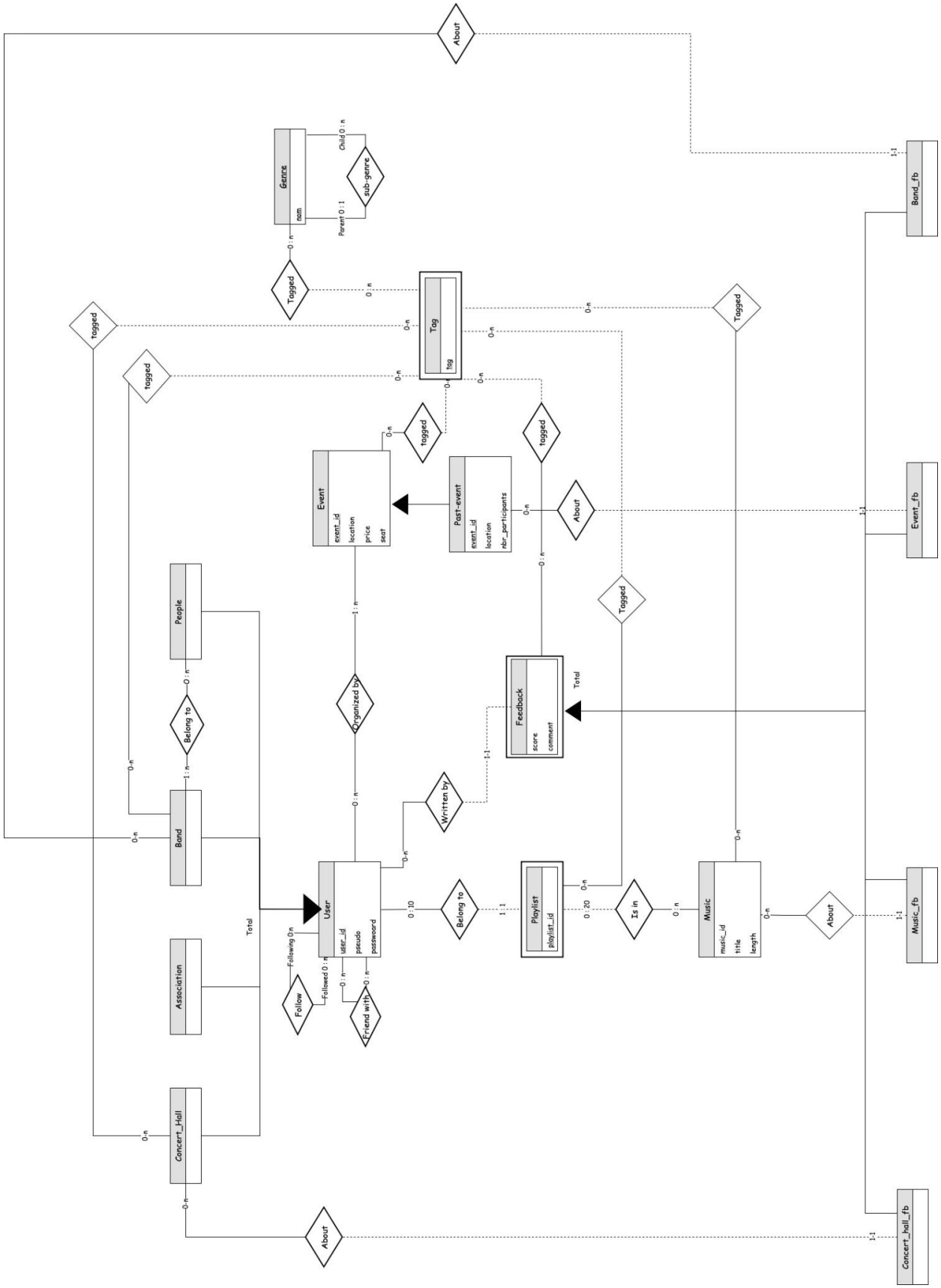
Groupe 44
GILARDONI Daniel
LIM Victoria

RAPPORT BD6 2022/2023

I. Modélisation E/R

A. Schéma



B. Contraintes non-présentées sur le schémas

- La relation *'friend with'* doit être symétrique.
- Un utilisateur ne peut pas être ami avec lui-même.
- Un utilisateur ne peut pas se suivre lui-même.
- L'attribut *'score'* de *'feedback'* doit être compris entre 0 et 10.
- On ne peut pas mettre plusieurs fois la même musique dans une même playlist.
- L'attribut *'nbr_participants'* de *'Past-event'* doit être inférieur ou égale à *'seat'* de *'Event'*.
- Un genre ne peut pas être son propre sous-genre.

C. Explications

- On a besoin de pouvoir distinguer les différents types d'utilisateurs.
- On a besoin de savoir sur quoi un utilisateur partage son avis.
- Un avis se distingue d'un autre avis par son auteur et l'objet qu'il concerne.
- Une playlist se distingue d'une autre playlist par l'utilisateur auquel il appartient et son identifiant.
- Une personne peut appartenir ou non à un ou plusieurs groupes et un groupe est composé d'au moins une personne.
- Un utilisateur peut suivre autant d'utilisateurs qu'il veut et peut être suivi ou non par plusieurs utilisateurs.
- Un utilisateur peut être ami avec autant d'utilisateurs qu'il veut et peut être l'ami ou non de plusieurs utilisateurs.
- Un utilisateur peut donner autant d'avis qu'il veut, et un avis n'est écrit que par un seul utilisateur.
- Un utilisateur peut organiser autant d'événements qu'il souhaite, et un événement doit être organisé par au moins une utilisateur.
- Un utilisateur possède au plus dix playlists et une playlist n'appartient qu'à un seul utilisateur.
- Un playlist peut contenir au plus vingt musiques et une musique peut appartenir à plusieurs playlist.
- Chaque avis ne porte que sur un seul objet, mais chaque objet peut avoir plusieurs avis.
- Un tag peut être utilisé sur plusieurs objets et un objet peut être tagué par plusieurs tags.
- Un genre peut avoir plusieurs sous-genre, mais il n'a pas de genre parent. Tandis qu'un sous-genre peut avoir un sous-genre et à au moins un genre parent.

II. Modélisation relationnelle

A. Schéma

type_user(<u>type_user</u>)
evenement(<u>id_event</u>, nom, localisation, prix, places, passé, nbr_participants)
utilisateur(<u>id_user</u>, pseudo, mdp, type_user)
genre(<u>id_genre</u>, nom)
musique(<u>id_musique</u>, id_user, titre, duree, id_genre)
playlist(<u>id_playlist</u>, <u>id_user, id_musique</u>)
tag(<u>tag_name</u>)
organisateur(<u>id_user</u>, <u>id_event</u>)
abonnement(<u>suivi</u>, <u>abonne</u>)
ami(<u>user1</u>, <u>user2</u>)
membre(<u>id_groupe</u>, <u>id_personne</u>)
sub_genre(<u>id_parent</u>, <u>id_enfant</u>)
avis_musique(<u>id_auteur</u>, <u>id_objet</u>, note, commentaire)

avis_evenement(id_auteur, id_objet, note, commentaire)
tagged_user(tag, id_objet)
tagged_musique(tag, id_objet)
tagged_genre(tag, id_objet)
tagged_evenement(tag, id_objet)

B. Contraintes

1. Contraintes externes

- L'attribut '*type_user*' de la table '*utilisateur*' fait référence à l'attribut '*type_user*' de la table 'type_user'.
- Les attributs *'id_user'* et *'id_genre'* de la table *'musique'* font respectivement références à l'attribut *'id_user'* de *'utilisateur'* et *'id_genre'* de *'genre'*.
- Les attributs *'id_user'* et *'id_musique'* de la table *'playlist'* font respectivement références à l'attribut *'id_user'* de *'utilisateur'* et *'id_musique'* de *'musique'*.
- Les attributs *'id_user'* et *'id_event'* de la table *'organisateur'* font respectivement références à l'attribut *'id_user'* de *'utilisateur'* et *'id_evenent'* de *'evenement'*.
- Les attributs *'suivi'* et *'abonne'* de la table *'abonnement'* font tous les deux références à l'attribut *'id_user'* de *'utilisateur'*.
- Les attributs *'user1'* et *'user2'* de la table *'ami'* font tous les deux références à l'attribut *'id_user'* de *'utilisateur'*.
- Les attributs *'id_groupe'* et *'id_personne* de la table *'membre'* font tous les deux références à l'attribut *'id_user'* de *'genre'*.
- Les attributs *'id_parent'* et *'id_enfant* de la table *'sub_genre'* font tous les deux références à l'attribut *'id_genre'* de *'utilisateur'*.
- Les attributs *'id_auteur'* et *'id_objet'* de la table *'avis_musique'* font respectivement références à l'attribut *'id_user'* de *'utilisateur'* et *'id_musique* de *'musique'*.
- Les attributs *'id_auteur'* et *'id_objet'* de la table *'avis_evenement'* font respectivement références à l'attribut *'id_user'* de *'utilisateur'* et *'id_event* de *'evenement'*.
- Les attributs *'tag'* et *'id_objet'* de la table *'tagged_user'* font respectivement références à l'attribut *'tag'* de *'tag'* et *'id_user'* de *'utilisateur'*.
- Les attributs *'tag'* et *'id_objet'* de la table *'tagged_musique'* font respectivement références à l'attribut *'tag'* de *'tag' et 'id_musique'* de *'musique'*.
- Les attributs *'tag'* et *'id_objet'* de la table *'tagged_genre'* font respectivement références à l'attribut *'tag'* de *'tag' et 'id_genre'* de *'genre'*.
- Les attributs *'tag'* et *'id_objet'* de la table *'tagged_evenement'* font respectivement références à l'attribut *'tag'* de *'tag''* et *'id_event* de *'evenement'*.

2. Autres contraintes implémentées

- Toutes suppressions d'une donnée se répercutent sur l'ensemble de la base de données.
- Dans la table *'evenement'*, le nombre de participants à un événement passé ne peut pas être supérieur au nombre de places disponibles.
- Dans la table *'evenement'*, si l'événement n'a pas encore eu lieu, alors *'nbr_partcipants'* vaut NULL.
- On ne peut pas avoir la même musique plusieurs fois dans une même playlist.
- Un genre ne peut pas être son propre sous-genre.
- Un utilisateur ne peut pas se suivre lui-même ou être son propre ami.
- La note d'un avis est comprise entre 0 et 10.

3. Contraintes non-implémentées

- Vérifier que l'attribut *'passe'* de *'evenement'* soit vrai pour pouvoir partager son avis sur cet événement.
- Vérifier la symétrie de la relation *'ami'*.
- Pour la table *'membre'*, vérifier que l'utilisateur ayant pour identifiant *'id_groupe'* soit bien un utilisateur de type *'groupe'* et celui ayant pour identifiant *'id_personne'* soit bien de type *'personne'*.
- Limiter le nombre de playlists pour chaque utilisateur à 10.
- Limiter le nombre de musiques présentes dans une playlist à 20.

C. Explications

- Puisque les différents types d'utilisateurs ne contiennent pas d'attribut supplémentaire, on élimine les entités enfants et on ne garde qu'une seule entité *'utilisateur'*, qui contiendra un attribut *'type_user'* pour différencier les différents types d'utilisateur.
- Puisque *'Past-event'* ne contient pas beaucoup d'attribut en plus par rapport à *'Event'*, on supprime *'Past-event'* et on déplace les associations auxquelles elle est reliée et ses attributs dans *'evenement'* avec un attribut 'passe' pour déterminer s'il a déjà eu lieu ou non.
- Puisqu'il est nécessaire de savoir sur quoi l'utilisateur partage son avis et qu'un avis ne parle que d'un seul sujet, on supprime l'entité mère *'Feedback'* et on reporte tous ses attributs et toutes ses associations à chacune de ses entités filles.
- Toutes les associations plusieurs à plusieurs deviennent une nouvelle table, plus particulièrement, *'Belong to'*, *'Follow'*, *'Friend'*, 'Sub-genre', ainsi que chaque association *'Tagged'*.

III. Limites

- Il faut ajouter une table pour chaque entité pouvant être taguée.
- Il faut ajouter une table pour chaque entité sur laquelle un utilisateur peut partager son avis.
- On suppose qu'on ne modifiera jamais un tag. (Il faudrait, dans le cas contraire, rajouter un ON UPDATE CASCADE pour toutes les clés étrangères référençant l'attribut *'tag_name'* de *'tag'*.)
- On suppose que seul l'auteur original de la musique peut la poster.

IV. Requêtes

A. Listes des requêtes

- Requête 1 : Liste des utilisateurs abonnés à 'daft_punk'.
- Requête 2 : Nombre d'avis moyen par musique.
- Requête 3 : Les auteurs des musiques d'une playlist.
- Requête 4 : Calculer la moyenne des notes d'une musique et son nombre total d'avis.
- Requête 5 : Récupérer les utilisateurs ayant créé des musiques bien notées du genre 'Rock'.
- Requête 6 : Les événements ayant un nombre de participants supérieur à la moyenne.
- Requête 7 : Les utilisateurs qui ont le plus d'amis en commun avec un utilisateur spécifique.
- Requête 8 : Obtenir la moyenne des notes données par un utilisateur à chaque genre musical.
- Requête 9 : Les genres de musique avec plus de 5 musiques associées et une note moyenne supérieure à 7.

- Requête 10 : Liste des utilisateurs et de leur nombre d'amis.
- Requête 11 : Les playlists qui durent le plus longtemps.
- Requête 12 : Le nombre de tags existants.
- Requête 13 : La playlist qui dure le moins longtemps.
- Requête 14 : Les playlists durent plus de 10 minutes.
- Requête 15 : Les utilisateurs ayant donné leur avis sur toutes les musiques.
- Requête 16 : Les utilisateurs les plus actifs sur NoiseBook (la plus de musique postée et d'événement organisé.).
- Requête 17 : Les musiques les plus longues de chaque playlist
- Requête 18 : Les genres parents du genre 'Dub'.
- Requête 19 : Le nombre moyen de participations de tous les événements ayant eu lieu au 'Parc Central'.
- Requête 20 : La moyenne des notes maximales attribuées à chaque musique par genre.

B. Conditions

- Requêtes portant sur au moins 3 tables : 3, 5, 8
- Requêtes avec une jointure réflexive : 1 et 9
- Requêtes avec une sous-requête corrélée : 6 et 7
- Requêtes avec une sous-requête dans le FROM : 20
- Requêtes avec une sous-requête dans le WHERE : 6, 7, 11, 13 et 17
- Requêtes avec des agrégats nécessitant GROUP BY et HAVING : 5, 9, 14 et 15
- Requêtes avec un calcul de deux agrégats : 4, 8 et 20
- Requêtes avec une jointure externe : 2, 9 et 16
- Requêtes équivalentes exprimant une condition de totalité : 15
- Requêtes qui n'envoient pas le même résultat si la base de données contient des NULL : 19
- Requêtes récursives : 18
- Requêtes avec fenêtrage : 10

V. Difficultés rencontrées

- Comment gérer les tags et les avis ?
- Trouver des requêtes qui correspondent aux conditions.