Projet NOISE DB

Rapport Rendu

Binome :

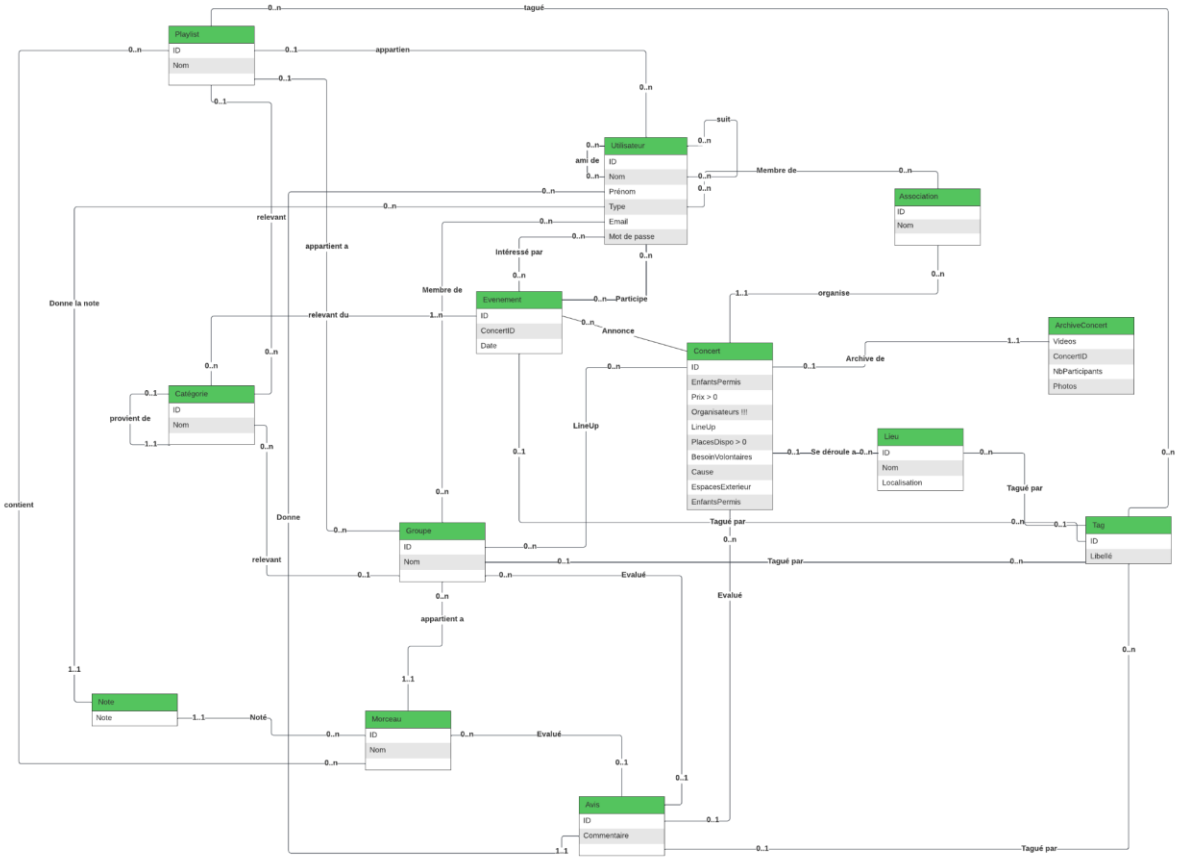
OTMANE CHERIF Mohammed Islem

SAADI Abdrezzak

Date: 20/05/2023

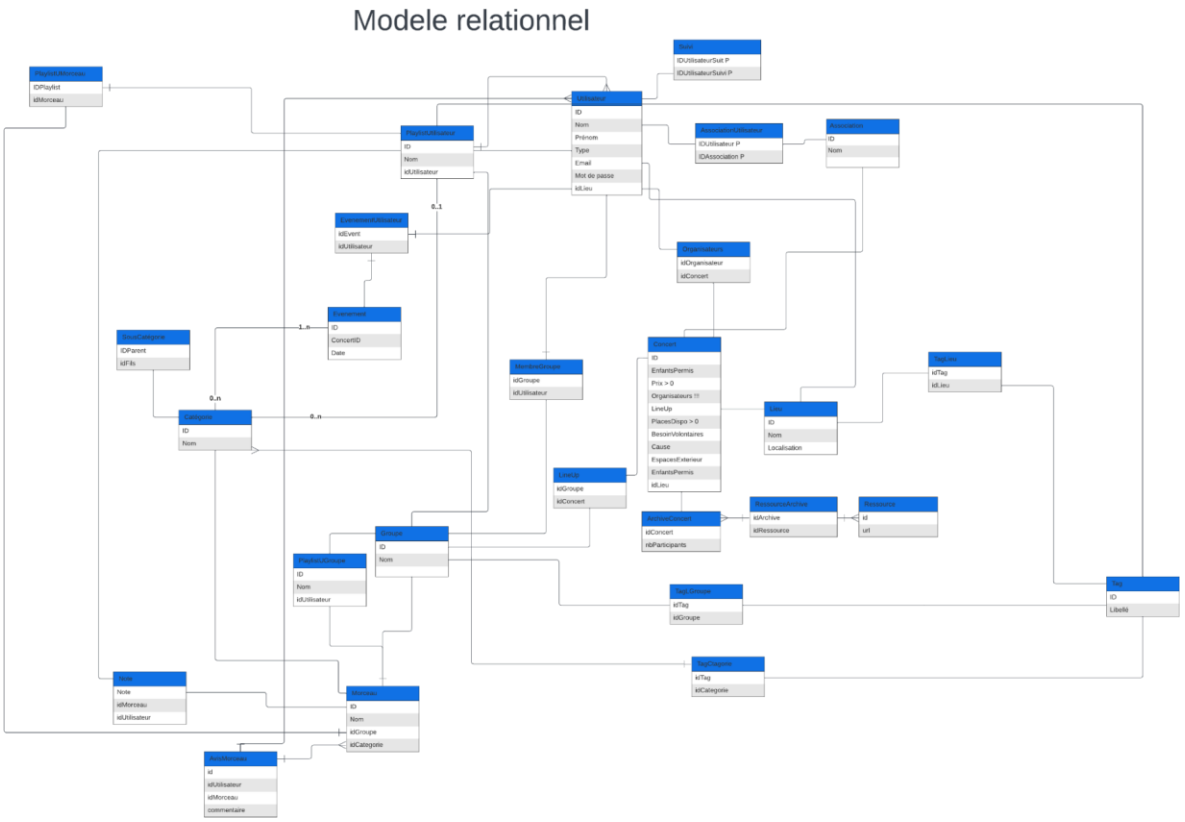
1. Conception de la base de données:

On a commencé par comprendre les besoins du système noiseBook et identifier ses besoins, pour identifier les entités principales impliquées, les relations entre eux, pour réaliser le schéma entité association .



Ensuite on a procédé à la création du schéma relationnel du schéma entité association en suivant les principes suivant:

1. Convertir les entités en tables
2. Identifier les clés primaires
3. Gérer les relations un-à-un
4. Gérer les relations un-à-plusieurs
5. Gérer les relations plusieurs-à-plusieurs
6. Définir les contraintes d'intégrité référentielle
7. Vérifier la cohérence



et finalement on a créé le script sql pour la création de notre base de données en postgresql.

2. Limitation de la modélisation:

Voici quelques limitations de la modélisation présenté:

- Regroupement des tables:

  Par exemple, les tables TagLieu, TagGroupe peut être représenté en une seule table et la colonne id peut référencer un id groupe ou id lieu.

- Performance:

  la gestion des archives de concerts avec la table "ArchiveConcert". Cette table conserve les informations archivées sur les concerts, y compris le nombre de participants.

  Un exemple de limitation de performance dans le modèle de base de données que vous avez fourni est la gestion des archives de concerts avec la table "ArchiveConcert". Cette table conserve les informations archivées sur les concerts, y compris le nombre de participants.

  Si notre application traite un grand volume de concerts et que le nombre de participants pour chaque concert est également élevé, cela peut entraîner

des performances médiocres lors de l'accès aux données de la table "ArchiveConcert".

- Extensibilité :

  Actuellement, on a des tables distinctes pour les playlists des utilisateurs et les playlists des groupes, avec les tables

  "PlaylistUtilisateur" et "PlaylistGroupe", Cette conception peut poser des problèmes d'extensibilité si vous souhaitez prendre en charge d'autres types de playlists, tels que des playlists pour des événements ou des playlists pour des lieux.

  On peut corriger ça en regroupant toutes les table dans une table playlist et ajouter une colonne type.

3. Description des requêtes :

Pour les requêtes présenté on a essayé de prendre un exemple pour chaque type de requête :

Type de requête	Description
1- sur au moins trois tables	donner la liste des organisateurs par concert
2-auto-jointure	donner la liste des utilisateurs qui ont le meme prenom
3-une sous-requête corrélée	donner la liste des utilisateurs qui sont des organisateurs
4-une sous-requête dans le FROM	selectionner la liste des groupe qui ont un nombre de concert >= 2
5- sous requête dans le where	la liste des concert ou la localisation est a paris
6- agregats avec groupeBy et having	la liste des prix moyens par groupe ou le prix >= 50
7-agregats avec groupeBy et	le nombre de concert organisé par association ou le

having :	nombre de concert > 1:
8- requête impliquant le calcul de deux agrégats	sélectionner la moyenne des maximum et minimum des prix de concert par association
9-LEFT JOIN	afficher le nombre de concert par lieu
10- RIGHT JOIN	toutes les associations, avec le nom du concert associé
11- FULL JOIN	tous les utilisateurs et leurs commentaires associés, que ce soit sur des morceaux, des groupes ou des concerts.
12- en utilisant Agrégat MAX	sélectionner le concert avec le plus de places disponibles
13- en utilisant AVG	le morceau qui a la meilleure note moyenne
14- en utilisant COunt, AVG et LIMIT	les cinq morceaux ayant reçu le plus grand nombre d'avis
15- une requête exprimant une condition de totalité avec des sous requêtes corrélées	récupérer les concerts qui n'ont aucune places disponibles
16- une requête exprimant une condition de totalité avec de l'agrégation	sélectionner uniquement les lieux qui ont au moins 2 concert associé:
17 et 18 -Les deux requêtes qui renverrait le même résultat si les tables ne contient pas de nulls et des résultats différent sinon	sélectionner les lieux qui ont un concert correspondant
19- requête récursive	sélectionner la liste des utilisateurs lié à l'utilisateur d'id = 1

20- requete avec fenêtrage:	le nombre total de concerts auxquels chaque utilisateur a participé

4. Indice de recommandation:

Pour mettre en place un système de recommandation de concerts pour les utilisateurs en doit suivre ces étapes:

- Analyse des données: Examinez les tables pertinentes telles que "Utilisateurs", "Concert", "Association", "Groupe", "Tag", etc., : pour définir les critères de recommandation et attribuer un poids à chaque attribut, on va choisir les informations sur le concert,l'association qui l'organise, le lieu où il se déroule, le line up qui vont performer et le tag du concert et l'avis de l'utilisateur sur le concert.
- Collecter les préférence de l'utilisateur:
  on va collecter les préférences des utilisateurs, telles que les genres musicaux préférés, en plus des informations précédentes exemple de requête pour collecter ces informations:

```
SELECT a.Nom AS Association, g.Nom AS Groupe, l.Nom AS Lieu, t.nom
AS Tag
FROM Utilisateurs u
LEFT JOIN AssociationUtilisateur au ON u.id = au.IdUtilisateur
LEFT JOIN Association a ON au.IdAssociation = a.id
LEFT JOIN MembreGroupe mg ON u.id = mg.idUtilisateur
LEFT JOIN Groupe g ON mg.idGroupe = g.id
LEFT JOIN Lieu l ON u.idLieu = l.id
LEFT JOIN TagGroupe          tu ON g.id = tu.idGroupe
LEFT JOIN Tag t ON tu.idTag = t.id
WHERE u.id = 1;
```

- Évaluation des concerts existants selon le critère de recommandation suivant les critère qu'on a choisi
- Filtrage des concerts de ce jour
- Classement des concerts et présentation des recommandations ex de la requete:

```
-- Évaluation des concerts existants
SELECT
    c.id,
```

```
    c.Nom,
    c.Date,
    c.PlacesDispo,
    -- Appliquer des critères et calculer un score pour chaque concert
    (CASE
        WHEN c.Genre = 'Rock' THEN 5
        WHEN c.Genre = 'Pop' THEN 3 ELSE 1
    END) AS genre_score,

FROM
    Concert c
WHERE
    c.Date >= CURRENT_DATE -- Filtrer les concerts futurs
;

-- Filtrage des concerts
SELECT
    *
FROM
    (
        -- Utilisez la requête précédente comme sous-requête
        SELECT
            c.id,
            c.Nom,
            c.Date,
            c.PlacesDispo, genre_score,
            -- Autres scores de critères
            -- ...
        FROM
            Concert c
        WHERE
            c.Date >= CURRENT_DATE -- Filtrer les concerts futurs )
    filtered_concerts
WHERE
    filtered_concerts.genre_score >= 4 -- Filtrer par genre_score
    -- Appliquer d'autres filtres en fonction des préférences de l'utilisateur
    -- ...

-- Classement des concerts
SELECT *
```

```
FROM (

        SELECT
            c.id,
            c.Nom,
            c.Date,
            c.PlacesDispo,
            genre_score FROM
            Concert c
        WHERE
            c.Date >= CURRENT_DATE -- Filtrer les concerts futurs )
    filtered_concerts
ORDER BY genre_score DESC -- Classement par genre_score


 --       Présentation des recommandations
SELECT
    *
FROM
    (
        SELECT
            c.id,
            c.Nom,
            c.Date,
            c.PlacesDispo,
            genre_score FROM
            Concert c
        WHERE
            c.Date >= CURRENT_DATE -- Filtrer les concerts futurs )
    ranked_concerts
LIMIT 10; – limiter le nombre de concerts a afficher
```