Test Technique - Backend / Data Engineering

Réalisé par :

Wail DEBZ

Étape 1 : Modélisation de la Donnée

1. Analyse de la Donnée Brute

Source de données : Le point de départ est un fichier CSV contenant des données brutes sur le catalogue et les performances de chansons, fournies par le partenaire de Base For Music Soundcharts.

Structure initiale : Le fichier contient plusieurs colonnes, dont la plus importante est timeSeries. Cette colonne se présente sous la forme d'une chaîne de caractères au format JSON, encapsulant les données de performance journalières.

Qualité des données : Une première analyse révèle plusieurs problèmes de qualité qui nécessitent un prétraitement rigoureux avant toute insertion en base de données :

- **Formatage des noms de colonnes :** Le nom de la première colonne est mal formaté (' 1\tsong_id') et doit être corrigé pour garantir la fiabilité des accès.
- **Données composites :** La colonne artist peut contenir plusieurs noms d'artistes dans une seule chaîne de caractères, ce qui viole le principe de la première forme normale et nécessite une séparation.
- **Données redondantes :** La colonne summaries contient des agrégats (comme le total des streams) qui sont calculables à partir des données détaillées de la colonne timeSeries. Pour éviter la redondance et assurer l'intégrité, cette colonne sera écartée du modèle final.
- **Corruption de fichier :** L'analyse a également révélé que le fichier source pouvait être physiquement corrompu (ex: ligne finale tronquée). Un pipeline de données robuste doit pouvoir détecter ou gérer de tels cas.

2. Stratégie de Nettoyage et de Normalisation

Les actions suivantes sont proposées pour préparer et structurer la donnée avant son intégration.

Renommage des champs pour plus de clarté :

- ' 1\tsong_id' est renommé en song_id de manière programmatique.
- image_url est renommé en cover_url pour mieux refléter son contenu.

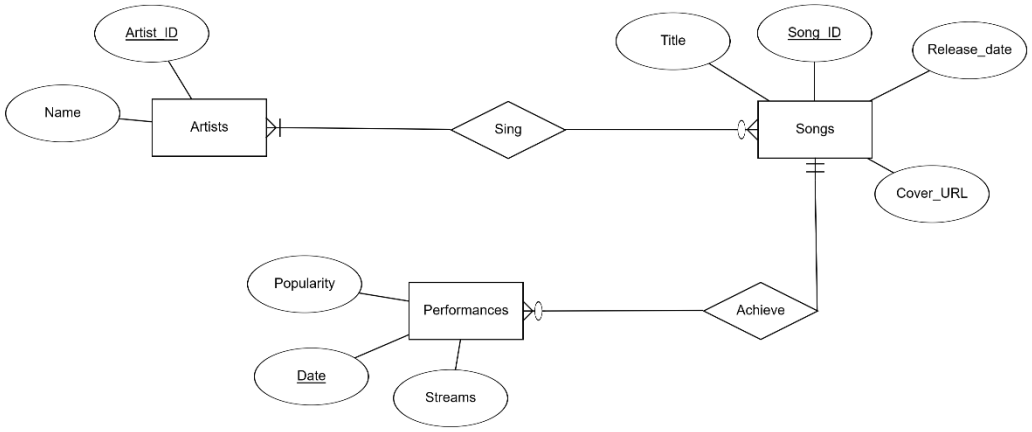
Transformation des données :

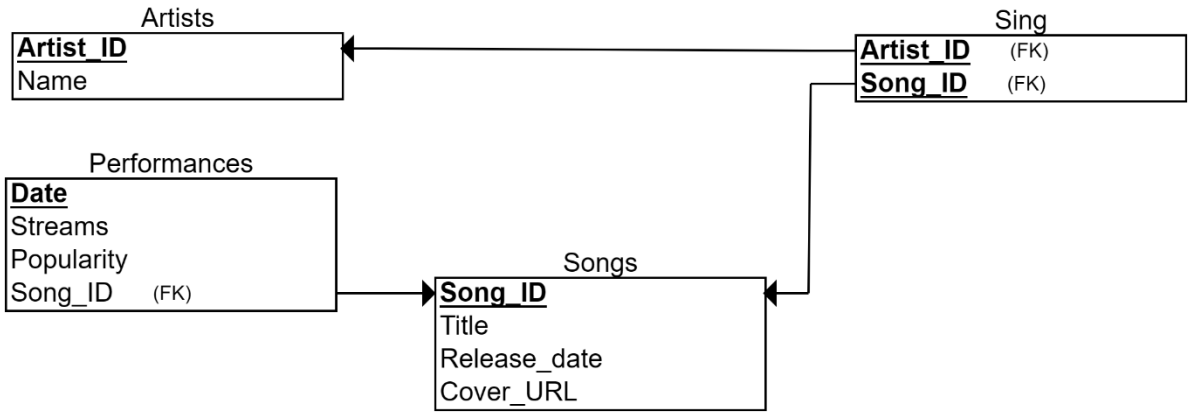
- **Artistes (Relation Plusieurs-à-Plusieurs) :** La chaîne artist est systématiquement parsée pour extraire chaque artiste. Ceci permet de peupler une table Artists dédiée (sans doublons) et une table de jonction Sing qui modélise la relation plusieurs-à-plusieurs.
- **Gestion des dates invalides :** Les valeurs de release_date non conformes (par ex: 0000-01-01 ou chaînes vides '') sont converties en NULL ou une date logique pour être compatibles avec le type DATE de PostgreSQL.

- **Gestion des données manquantes dans le JSON :** Les objets JSON au sein de timeSeries n'ont pas toujours les clés spotify-streams et spotify-popularity. Le script d'intégration gère ces cas en insérant une valeur NULL lorsque la clé correspondante est absente, assurant ainsi l'exhaustivité de l'enregistrement des performances.
- **Gestion des ID des artistes et chansons :** une colonne artist_ID a été ajoutée de type Integer, et le song_id a été modifié d'une chaine des caractères à un type INTEGER, la raison pour cette modification est le faite que l'identifiant original est présent dans l'URL de l'image, ainsi que l'utilisation d'un identifiant numérique est plus pratique.

3. Schéma Relationnel Proposé (ERD)

Le schéma ci-dessous est proposé pour une implémentation sous PostgreSQL, comme demandé dans le test. Il est le résultat de l'analyse et de la stratégie de normalisation décrites précédemment.





Justification du modèle : Le schéma est conçu pour être normalisé (proche de la 3FN) et performant.

- **Table Artists (Artist_ID [PK], Name)** : Stocke chaque artiste une seule fois pour garantir l'unicité et éviter la redondance. Artist_ID est un entier auto-généré.
- **Table Songs (Song_ID [PK], Title, Release_Date, Cover_URL)** : Contient les métadonnées uniques de chaque chanson. Release_Date est de type DATE et accepte les valeurs NULL.
- **Table Sing (Artist_ID [FK], Song_ID [FK])** : Table de jonction qui modélise la relation plusieurs-à-plusieurs entre Artists et Songs. Sa clé primaire est composite.

- **Table Performances (Song_ID [FK], Date [PK], Streams, Popularity)** : Stocke les données de séries temporelles. La clé primaire composite (Song_ID, Date) garantit une seule entrée par chanson et par jour.
- **Point crucial - Choix du type de données :** Les colonnes Streams est définie avec le type **BIGINT**. Ce choix est justifié par la découverte, lors de l'analyse, que les valeurs de streams peuvent provoquer une erreur "entier en dehors des limites". L'utilisation de BIGINT garantit la capacité à stocker des chiffres de grande magnitude, ce qui est essentiel pour des données de streaming à grande échelle.