PHASE-1
IMDb Dataset Exploration and Loading Using PostgreSQL

Karunawansa Yasiru	Mehta Akshat	Tidwell Kippy	Xu Shengyan
Yik4325	Am1717	Kct2548	Sx6553

Table of Contents

1. Introduction

This project focuses on creating a relational database structure for IMDb data using PostgreSQL. The IMDb dataset contains information about various media, such as movies, TV shows, documentaries, and related people. The project includes building an Entity-Relationship (ER) diagram to represent the data model, implementing a SQL schema, and developing a Python program to load data from TSV files into the PostgreSQL database.

2. Project Description

Dataset Overview

The dataset chosen for this project comes from Kaggle and contains multiple files with IMDb data. These files include information about titles, people involved, ratings, and other metadata. The data is stored in TSV format (Tab-Separated Values), which contains tabular data for easy import into the PostgreSQL database.

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

Data Description

The dataset is divided into several files, each providing specific information:

- People (name.basics): This file contains details about individuals associated with media, including their unique identifiers (nconst), birth year, death year, primary professions, and titles they are known for.

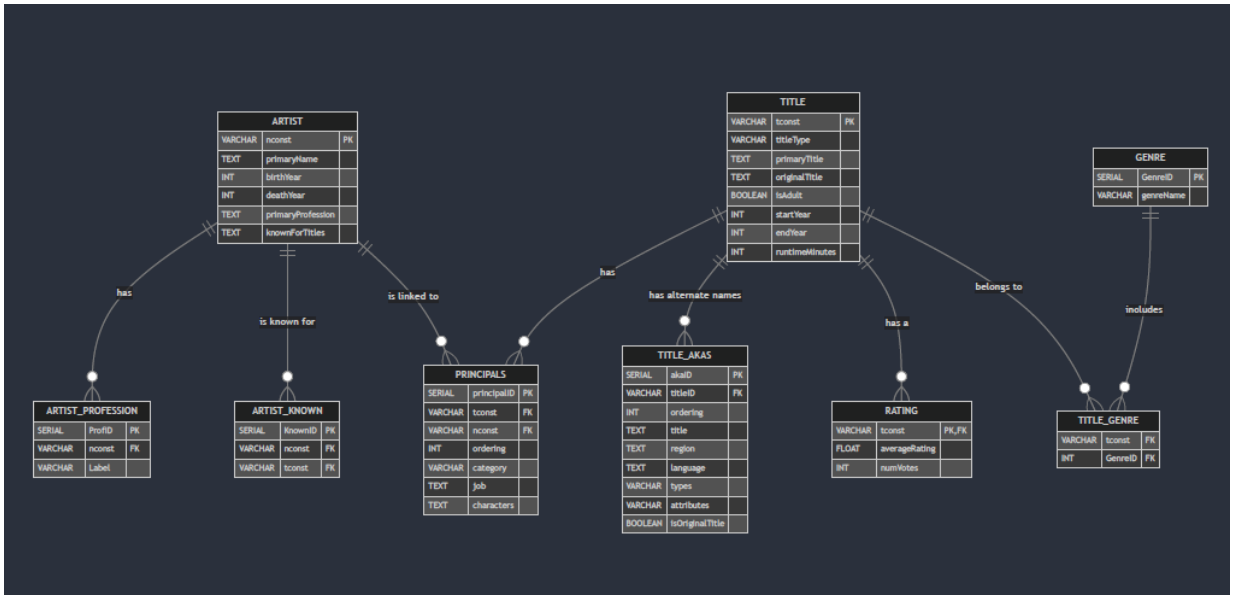
- Titles (title.basic): This file stores metadata about different media titles (e.g., movies, TV shows, shorts) such as their type, primary title, original title, whether it contains adult content, start and end years, runtime, and genres.

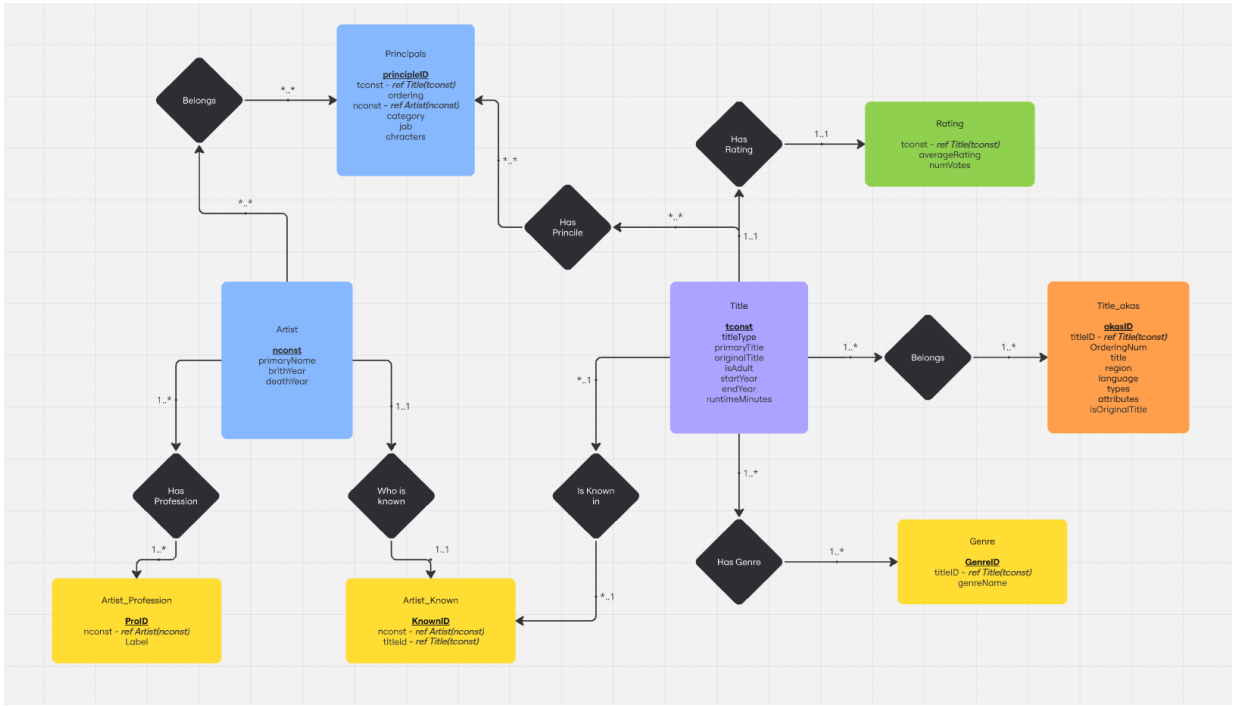
- Alternate Titles (title.akas): This file provides alternate names for titles across different regions and languages.

- Title Principals (title.principals): This file contains data about the people who worked on specific titles.

- Title Ratings (title.ratings): This file stores ratings for titles, capturing the average user rating and the number of votes the title has received.

Relational Model & ER Diagram Explanation and Relationships





- **Artist (nconst):** Represents people associated with media titles. It stores each artist's unique identifier, name, birth year, and death year.
- **Title (tconst):** Refers to various types of media (movies, TV shows, etc.). This entity stores details like title type, primary and original titles, runtime, and whether the title contains adult content.
- **Rating:** Each title has a rating, which consists of an average rating (on a scale of 1 to 10) and the number of votes it received.
- **Genre**: Titles can be associated with one or more genres, such as drama, action, or comedy.
- **Principals**: An associative entity linking artists to titles, storing details about their role in the production.
- **Artist Profession:** Stores professions associated with each artist, allowing flexible classification of their work.

Relationships

1. **Artist and Artist_Profession (Has Profession):** One-to-Many relationship. Each artist can have multiple professions.
2. **Artist and Artist_Known (Who is Known):** One-to-Many relationship. Each artist is known for one or more titles.
3. **Artist and Principals (Has Principle):** One-to-Many relationship. Each artist can be associated with multiple titles.
4. **Title and Rating (Has Rating):** One-to-One relationship. Each title has one rating.
5. **Title and Genre (Has Genre):** Many-to-Many relationship. Each title can have multiple genres.
6. **Title and Title_Akas (Belongs):** One-to-Many relationship. Each title can have multiple alternate names.
7. **Title and Principals (Belongs):** One-to-Many relationship. Each title can have multiple artists working on it.

3. Implementation

SQL Tables Explanation

The relational structure is defined through several SQL tables. Each table represents a key entity, with relationships and constraints enforced to maintain data integrity.

- Artist Table: Stores the basic information about artists, such as their unique identifier (nconst), primary name, birth and death years, professions, and the titles they are best known for.

- Artist_Profession Table: Links each artist to their professions. It stores a reference to the artist's unique identifier (nconst) along with the profession label (e.g., actor, director).

- Title Table: Stores information about various media titles. The tconst field serves as the primary key, holding details like the title type, primary and original titles, and the title's runtime.

- Title_Akas Table: Stores alternate names for titles, linking them to regions or languages.

- Rating Table: Stores information about the average user rating and the number of votes a title has received.

- Genre Table: Stores genre information, with each genre having a unique identifier.

- Principals Table: Links artists to titles, specifying their role or job in the production.

Explanation of CREATE TABLE SQL Code

The CREATE TABLE SQL queries define the structure of each table and enforce necessary constraints.

- Primary Keys: Each main entity table has a primary key (e.g., nconst in the Artist table, tconst in the Title table).

- Foreign Keys: Used to enforce relationships between tables, such as linking titles and artists in the Principals table.

- Data Types: Fields are assigned appropriate data types (e.g., VARCHAR, BOOLEAN, INT) based on their contents.

- Cascading Deletes: Some foreign keys use ON DELETE CASCADE to automatically delete related records when a referenced record is deleted.

- Constraints: In some tables, constraints like CHECK are applied to enforce valid data ranges, such as in the Rating table where averageRating is constrained between 1.0 and 10.0.

Data Loading Program Explanation

A Python program was created to automate the process of loading IMDb data from TSV files into the PostgreSQL database. This program connects to the PostgreSQL database using the psycopg2 library, reads the TSV files, processes each row, and inserts the data into the relevant SQL tables. The program handles missing or inconsistent data and ensures that invalid records are not inserted into the database. The data is committed in batches for performance efficiency.

4. Conclusion

This project successfully demonstrates the process of building a relational database for IMDb data using PostgreSQL. The ER diagram outlines the relationships between entities such as artists, titles, ratings, and genres. The SQL schema enforces data integrity, and the Python data-loading program automates the import of large datasets into the database.

5. Appendix

Instructions on Running the Program:

1. Create a new database in PostgreSQL and run the provided SQL queries to create the relevant tables.

2. Update the connection parameters in the Python program to match your PostgreSQL configuration.

3. Run the Python program (load_data.py) to load the IMDb data from TSV files into the database.

4. Ensure that the TSV data files are located in the same directory as the Python script.

5. After the program completes, the data will be available in the PostgreSQL database for querying.