Movie Streaming Platform Database

Project Documentation

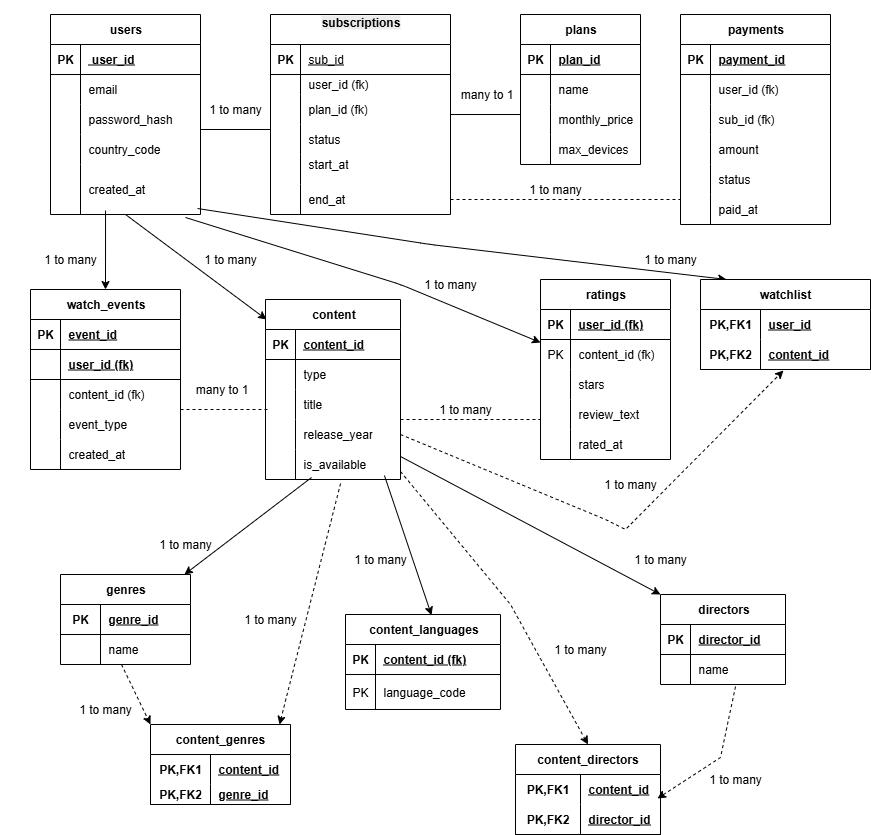
**1. Project Motivation**

* The project aims to design and implement a database system for a movie streaming platform (like Netflix or Amazon Prime) that can:
* Manage millions of users, thousands of movies/shows.
* Support subscriptions, payments, and content security.
* Provide personalized recommendations.
* Enable smooth streaming and detailed analytics.

**2. Objectives**

* Store user information and their subscription details.
* Maintain a catalog of movies, genres, and actors.
* Track user activity, including watch history and ratings/reviews.
* Provide reliable data for analytics and recommendations.

**3. ER Diagram**

****

**3. Database Entities**

* **Users → Stores user information.**
* **Plans → Subscription plans offered (Basic, Premium, etc.).**
* **Subscriptions → Connects users with plans (start date, end date, status).**
* **Payments → Stores payment transactions for subscriptions.**
* **Content → Stores movies/shows information.**
* **Directors, genres, content\_languages → Additional details of content.**
* **content\_directors, content\_genres → Linking tables for many-to-many relations.**
* **8. Ratings → Users’ ratings and reviews for content.**
* **9. Watch\_events → what users actually watched (date, progress).**
* **10. Watchlist → Movies/shows users saved to watch later.**

**4. Main Operations (CRUD)**

* **Create:** Insert new movies, users, actors, and subscriptions.
* **Read:** Retrieve movie catalogs, user watch history, and ratings.
* **Update:** Modify user plans, update actor details, and correct ratings.
* **Delete:** Remove outdated subscriptions or old watch history.

**5. Relationships**

* Users → Subscriptions → Plans (1-to-many relationship).
* Subscriptions → Payments (1-to-many).
* Users → Ratings → Content (many-to-many).
* Users → WatchEvents → Content (many-to-many).
* Users → Watchlist → Content (many-to-many).
* Content → Genres / Directors / Languages (many-to-many).

**6. Example SQL Operations**

* **DDL (Create Tables):**
* CREATE TABLE Users (
* user\_id INT PRIMARY KEY,
* name VARCHAR(100),
* email VARCHAR(100) UNIQUE,
* subscription\_id INT,
* join\_date DATE,
* FOREIGN KEY (subscription\_id) REFERENCES Subscription\_Plans(subscription\_id)
* );
* **DML (Insert Data):**
* INSERT INTO Users (user\_id, name, email, subscription\_id, join\_date)
* VALUES (1, 'John Doe', 'john@example.com', 1, '2024-01-10');
* **DCL (Grant Permissions):**
* GRANT SELECT, INSERT ON MovieStreamingDB.\* TO 'app\_user'@'localhost';
* **TCL (Transaction Control):**
* START TRANSACTION;
* UPDATE Users SET subscription\_id = 2 WHERE user\_id = 1;
* COMMIT;

**7. Events & Triggers**

* **Automatic Rating Update:** Trigger to update average movie rating after a new review is added.
* **Watch History Clean-up:** Event to remove history older than 1 year.

**8. Use Cases**

* Show trending movies based on user reviews.
* Generate a report of most-watched genres.
* View user watch history and suggestions.