Design Netflix Schema

Design Database Schema for a system like Netflix with the following Use Cases. You can draw it on pen and paper and upload the image of your solution.

Use Case

- Netflix has users
- Every user has an email and a password
- Users can create profiles to have separate independent environments.
- Each profile has a name and a type. Type can be KID or ADULT.
- There are multiple videos on Netflix.
- For each video, there will be a title, description and a cast.
- A cast is a list of actors who were a part of the video. For each actor, we need to know their name and list of videos they were a part of.
- For every video, for any profile who watched that video, we need to know the status (COMPLETED/ IN PROGRESS).
- For every profile for whom a video is in progress, we want to know their last watch timestamp.

Solution :

Netflix Schema Design

Main Tables:

1. **Users**
    - **User_id (PK) - INT**
    - Email - VARCHAR(255)
    - Password - VARCHAR(150)
2. **Profiles**
    - **Profile_id (PK) - INT**
    - Name - VARCHAR(150)
    - type - ENUM('Kid', 'Adult')
    - user_id (FK) - INT
3. **Videos**
    - **Video_id (PK) - INT**
    - Title - VARCHAR(150)
    - Description - TEXT
4. **Actors**
    - **Actor_id (PK) - INT**
    - Actor_Name - VARCHAR(150)

Lookup Tables:

5. **Videos_Actors (m:m)**
    - **Composite Primary Key: (Video_id, Actor_id)**
    - **Video_id (FK) - INT**
    - **Actor_id (FK) - INT**
6. **Profile_Video_Progress (m:m)**
    - **Composite Primary Key: (Profile_id, Video_id)**
    - **Status - ENUM('Completed', 'In Progress')**
    - **Last_Watch - DATETIME**
    - **Profile_id (FK) - INT**
    - **Video_id (FK) - INT**

Primary Keys:

1. **User_id in Users**
2. **Profile_id in Profiles**
3. **Video_id in Videos**
4. **Actor_id in Actors**
5. **(Video_id, Actor_id) Composite PK in Videos_Actors**
6. **(Profile_id, Video_id) Composite PK in Profile_Video_Progress**

Foreign Keys:

1. **user_id in Profiles refers to User_id in Users**
2. **Video_id in Videos_Actors refers to Video_id in Videos**
3. **Actor_id in Videos_Actors refers to Actor_id in Actors**
4. **Profile_id in Profile_Video_Progress refers to Profile_id in Profiles**
5. **Video_id in Profile_Video_Progress refers to Video_id in Videos**

Indexes:

1. **User_id in Users (PK)**
2. **Profile_id in Profiles (PK), user_id in Profiles (FK)**
3. **Video_id in Videos (PK)**
4. **Actor_id in Actors (PK)**
5. **(Video_id, Actor_id) Composite PK in Videos_Actors**
6. **(Profile_id, Video_id) Composite PK in Profile_Video_Progress**

Cardinality and Relationships:

1. **Users to Profiles: One-to-Many (1:m)**
    - Each user can have multiple profiles, but each profile belongs to only one user.
2. **Profiles to Videos: Many-to-Many (m:m)**
    - Each profile can have multiple videos
    - Each video can be watched by multiple profiles.
    - Managed by Profile_Video_Progress.
3. **Videos to Actors: Many-to-Many (m)**
    - Each video can have multiple actors
    - Each actor can act in multiple videos.
    - Managed by Videos_Actors.