Design and Development of a Media Streaming Service Database

Introduction

This Media Streaming Service database is designed in a way to manage and organize information for a streaming platform.
It handles user data, subscriptions, payments, entities e.g. movies, series, and episodes, ...
In the rest of this report we outline the requirements, conceptual design, and technical specifications of the database, ensuring scalability, efficiency, and flexibility to support key platform functionalities like user interactions, media curation, and recommendations.

Step 1 : Requirements Analysis

Primary Entities and Attributes

1. **User**
    - Attributes: UserID (PK), Username, Email, Password, RegistrationDate, ProfilePicture, Address.
2. **Subscription**
    - Attributes: SubscriptionID (PK), UserID (FK), TierName, StartDate, EndDate.
3. **Payment**
    - Attributes: PaymentID (PK), SubscriptionID (FK), Amount, PaymentDate, TransactionStatus.
4. **Media** *(Superclass for Movies and Series)*
    - Attributes: MediaID (PK), Title, Genre, ProductionYear, AverageRating, DirectorID (FK), ProductionCompanyID (FK).
5. **Movie** *(Subclass of Media)*
    - Additional Attributes: Duration.
6. **Series** *(Subclass of Media)*
    - Additional Attributes: TotalSeasons.
7. **Episode**
    - Attributes: EpisodeID (PK), SeriesID (FK), SeasonNumber, EpisodeNumber, Title, Duration, StorageServer, StoragePath.
8. **StorageLocation**

- Attributes: LocationID (PK), ServerName, FilePath.

9. **ProductionCompany**
   - Attributes: CompanyID (PK), Name, EstablishmentYear, ContactInfo.
10. **Person** *(Superclass for Director, Actors, Producers)*
    - Attributes: PersonID (PK), Name, BirthDate.
11. **Comment**
    - Attributes: CommentID (PK), UserID (FK), MediaID (FK), CommentText, CommentDate.
12. **Rating**
    - Attributes: RatingID (PK), UserID (FK), MediaID (FK), RatingValue.
13. **WatchLaterList**
    - Attributes: ListID (PK), UserID (FK), MediaID (FK).

Relationships and Constraints

- **User ↔ Subscription: One-to-Many (1:N)**
  - A user can have multiple subscriptions over time.
  - Foreign Key: UserID in Subscription.
- **Subscription ↔ Payment: One-to-Many (1:N)**
  - A subscription can have multiple payments.
  - Foreign Key: SubscriptionID in Payment.
- **Media ↔ ProductionCompany: Many-to-One (N:1)**
  - Many media items belong to one production company.
  - Foreign Key: CompanyID in Media.
- **Media ↔ StorageLocation: One-to-One (1:1)**
  - Each media item has one storage location.
  - Foreign Key: LocationID in Media.
- **Media ↔ Comment: One-to-Many (1:N)**
  - Each media item can have multiple comments.
  - Foreign Key: MediaID in Comment.
- **Media ↔ Rating: One-to-Many (1:N)**
  - Each media item can have multiple ratings.
  - Foreign Key: MediaID in Rating.
- **Series ↔ Episode: One-to-Many (1:N)**
  - A series can have multiple episodes.
  - Foreign Key: SeriesID in Episode.
- **User ↔ Comment: One-to-Many (1:N)**
  - A user can leave multiple comments.
  - Foreign Key: UserID in Comment.
- **User ↔ Rating: One-to-Many (1:N)**
  - A user can rate multiple media items, but only once per item.

    - Foreign Key: UserID in Rating.
- **User ↔ WatchLaterList: One-to-Many (1:N)**
    - A user can add multiple media items to their watch list.
    - Foreign Key: UserID in WatchLaterList.

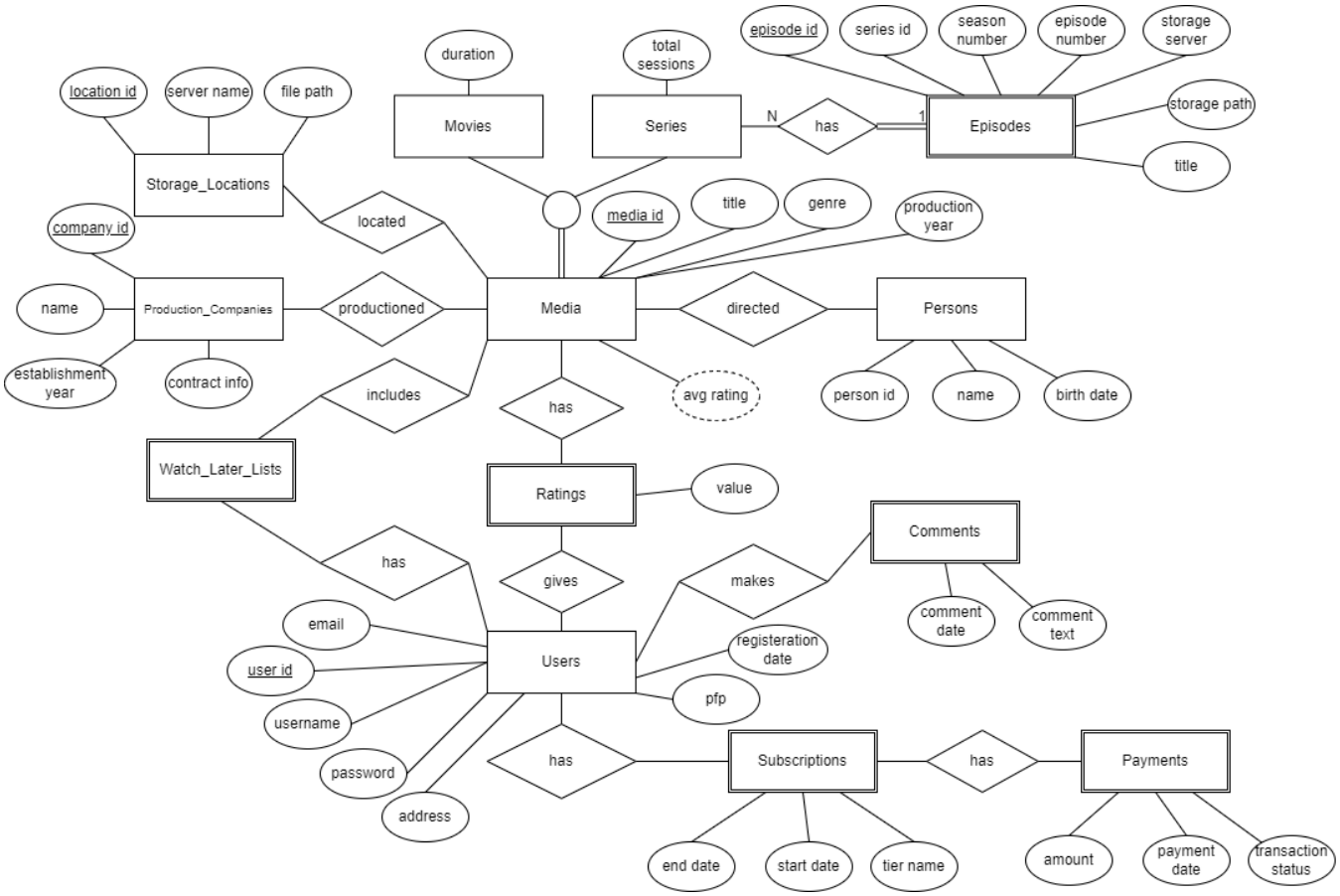
Constraints

- **Unique Constraints:**
    - UserID, Email in User.
    - Title in Media (within the same production year).
- **Validations:**
    - RatingValue must be between 0-5.
    - Subscription dates must be valid (StartDate < EndDate).
    - Prevent duplicate ratings by the same user for the same media.
- **Composite Keys:**
    - UserID + MediaID in Rating to enforce one rating per user-media pair.
- **Primary Keys**:
    - Unique identifiers for all entities (e.g., user_id, media_id).
- **Foreign Keys**:
    - Ensure referential integrity (e.g., user_id in subscriptions, media_id in comments).

Step 2 : Conceptual Database Design

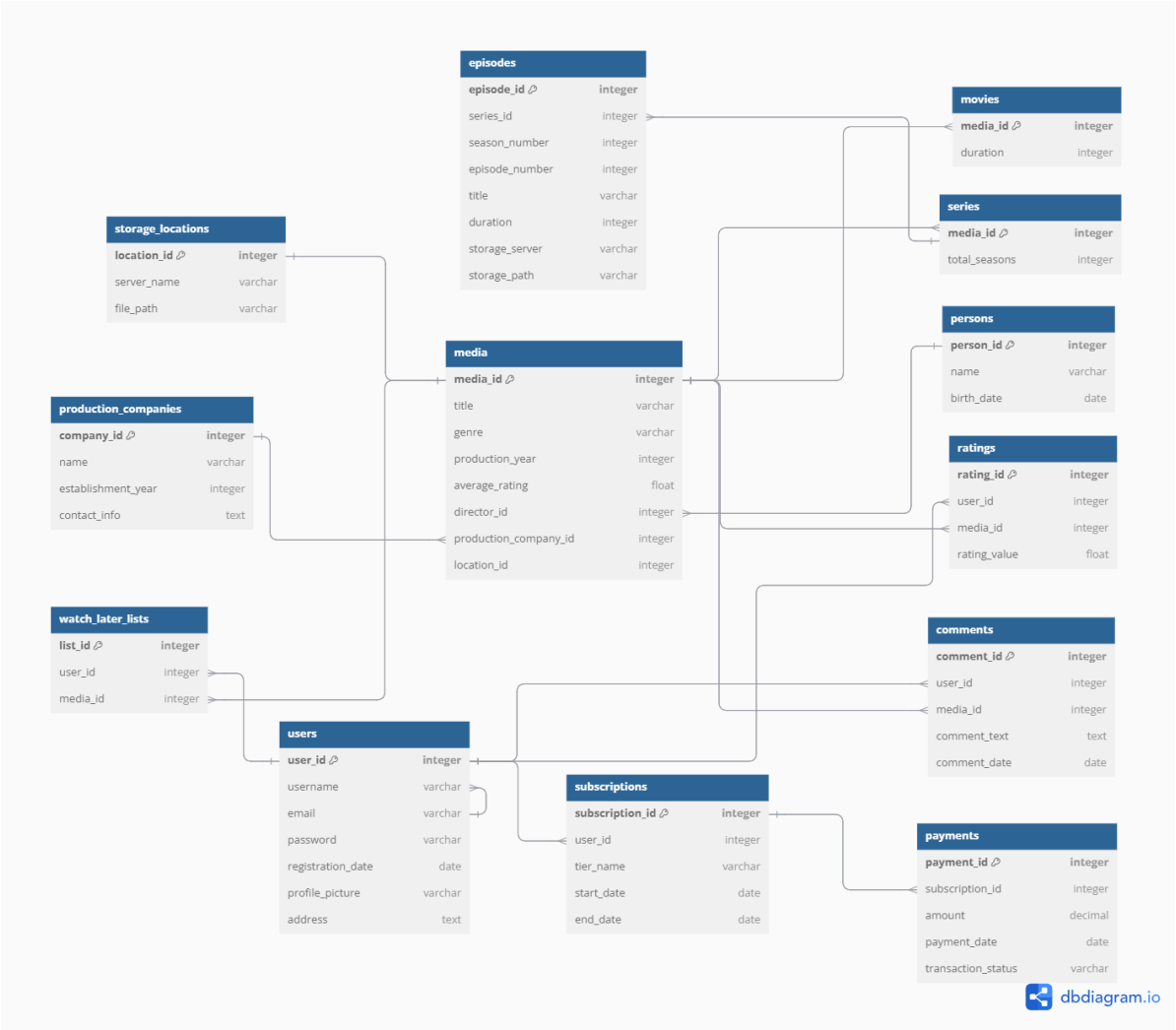
Enhanced Entity-Relationship Diagram (EER)

- **Hierarchical structure:**
  - Media as a superclass for Movies and Series.
  - Person as a superclass for Director, Actors, and Producers.



UML Diagram

- Class diagrams show the object-oriented design for the database.
- Important note: the DBML language that prof suggested we use does not support inheritance like the broader UML and I will implement it further in the future phases of project.
- Relationships (e.g., One-to-Many or One-to-One) between entities are defined in the link below the image.



[The link to dbdiagram.io to check the cardinalities]