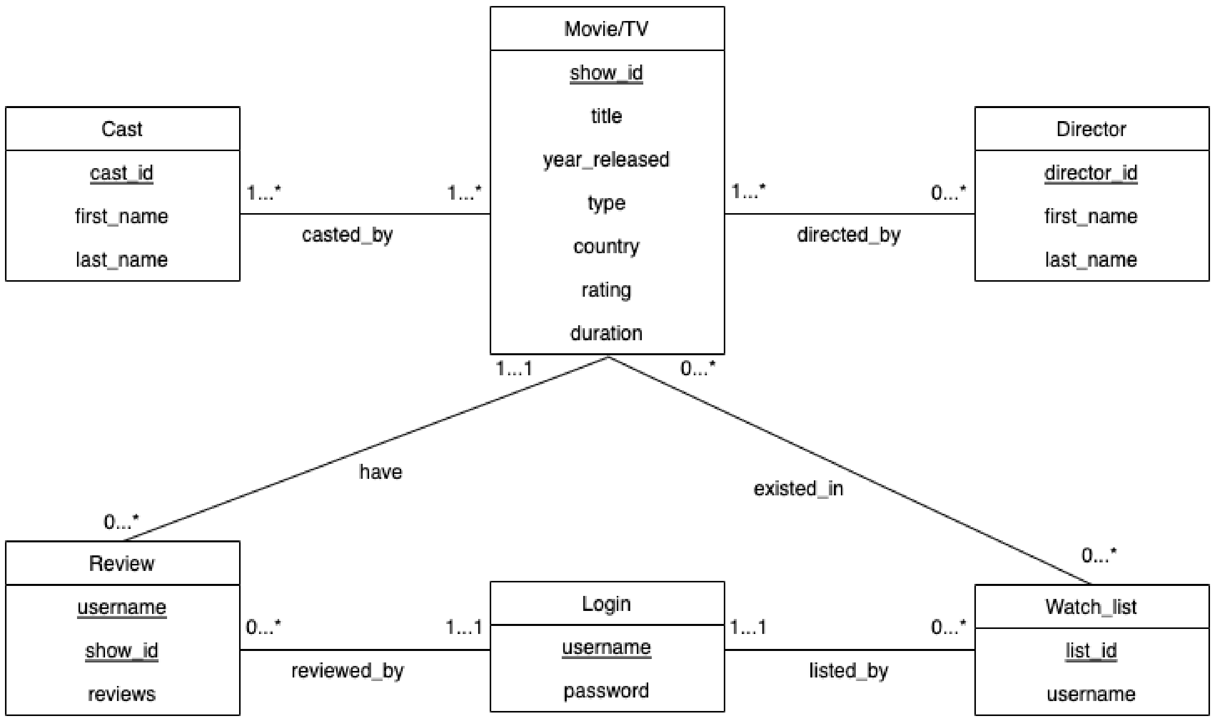
Conceptual Design

UML Diagram



Logical Design:

Main Entity:

Movie/TV (show_id: INT [PK], title: VARCHAR(255), year_released: INT, type: VARCHAR(255), country: VARCHAR(255), rating: VARCHAR(20), duration: VARCHAR(255))

Cast (cast_id: VARCHAR(255) [PK], first_name: VARCHAR(255), last_name: VARCHAR(255))

Director (director_id: VARCHAR(255) [PK], first_name: VARCHAR(255), last_name: VARCHAR(255))

Review (username: VARCHAR(255) [FK to Login.username] [PK] , show_id: INT [FK to Movie/TV.show_id][PK], reviews: VARCHAR(4095))

Login (username: VARCHAR(255) [PK], password: VARCHAR(255))

Watch_list (list_id: INT[PK],username: VARCHAR(255)[FK to Login.username])

Relational Table:

```
Existed_in (PK: listed_id: INT,
                show_id: INT
            FK: listed_id References Login,
                show_id References Movie/TV
              )

Directed_by (PK: director_id: VARCHAR(255),
                 show_id: INT (PK)
             FK: director_id References Director,
                 show_id References Movie/TV
              )

Casted_by (PK:cast_id: VARCHAR(255) ,
              show_id: INT
           FK: cast_id References Cast,
              show_id References Movie/TV
              )
```

Diagram Description:

Diagram Assumption:

Movie/TV:
Show_id is the primary key for this table. A movie/TV show also has its title, release year(year_released), type, country, rating (PG, R, etc.), and duration.

Cast:
Cast_id is the primary key for this table. Each unique cast_id represents an actor. An actor has a first name and last name which are both strings.

Director:
Director_id is the primary key for this table. A director has a first name and last name which are both strings.

Watch_list:
For each watchlist, we will have a unique list_id as its primary key, and there will be a username as a foreign key referencing who creates the watchlist. The show_id will be listed in the relational table between the movie entity, in order to reduce redundancy.

Review:
This table does not have a unique id since we do not need it to uniquely identify each review. So the primary key of this table is composed of both username and show_id. Each review should be posted by one user(username), commented under one movie/TV show(show_id), and it has the attribute ‘reviews’ which contains its content.

Login:
We will have a login system for each customer. This system will require a unique username as a primary key, and customers can also set their password.

Description of Each Relationship and Cardinality

Movie and Cast:
The relationship between movie/show and the cast is many to many, since one movie/show can have many actors/actresses and one cast can act in many movies/shows.

We assume that for every movie/show it will have at least one cast, and for every cast there will be at least one movie/show. (There won’t be NULL value for between the movie/show and cast)

Movie and Director:
The relationship from movie to director is many to many. A movie/TV can have 0 directors, one director, or more than 1 director. We set a show that can have 0 directors because the director information of some movies is missing in our dataset. One director can direct many movies or TV shows.

Login and watch list:
The relationship from user to watch list is 1 to many. Each user can create many watch lists, and each watch list can only be created by at most one user.

Login and Review:
The relationship from user to review is 1 to many. Each user can have several reviews or choose not to create a review, while each review can be created by at most one user.

Watch List and movie:
The relationship from movie/TV to watchlist is many to many, because each watch list can contain zero or more movie/TVs, and every movie/TV could be contained in zero or more watch lists.

Review and Movie:
The relationship from movie/TV to review is 1 to many because each movie/TV can have zero or more reviews, but every review can only exist for exactly one movie/TV.