

Entity Relationships

Exercise

Outline	2
Hands-on	2
Movie Genres	2
Person Role in a Movie	2
User Ratings	3
Conclusion	3

Outline

In this exercise lab, we will extend the app's current data model by adding relationships between the Entities. By the end of the exercise, we want to make sure that:

- A movie can have a genre and multiple movies can have the same genre.
- A person can have a role (or different roles) in a movie, or in different movies. The database can also have multiple people with the same role, for instance, several Directors and several Actors.

Also, at this point, we know that the app will support a new functionality in the future that will allow users to rate a movie. To support that, we need to add an extra Entity named **UserMovieRating**, which relates the users of the app to the movie they just rated. A user can rate multiple movies and a movie can be rated by multiple users.

Hands-on

In this exercise, we need to go back to the Data tab and update the data model. We need to create relationships between the existing Entities, and also add additional Entities.

Movie Genres

Let's start with the movie genres. We want to make sure that a movie can have a genre. Also, we don't want to limit our database to having a single movie of a single genre. For instance, just one comedy or one horror movie. We want our database to have multiple movies of the same genre.

Person Role in a Movie

We want to relate a person with a movie, using its role. For that, we want our data model to follow these rules:

- A Person can participate in multiple movies with the same or different roles.
- A Person can participate in the same movie, with different roles.
- A Movie can have multiple people participating in it, with the same or different roles.

Hint: Creating a new Entity will help :)

User Ratings

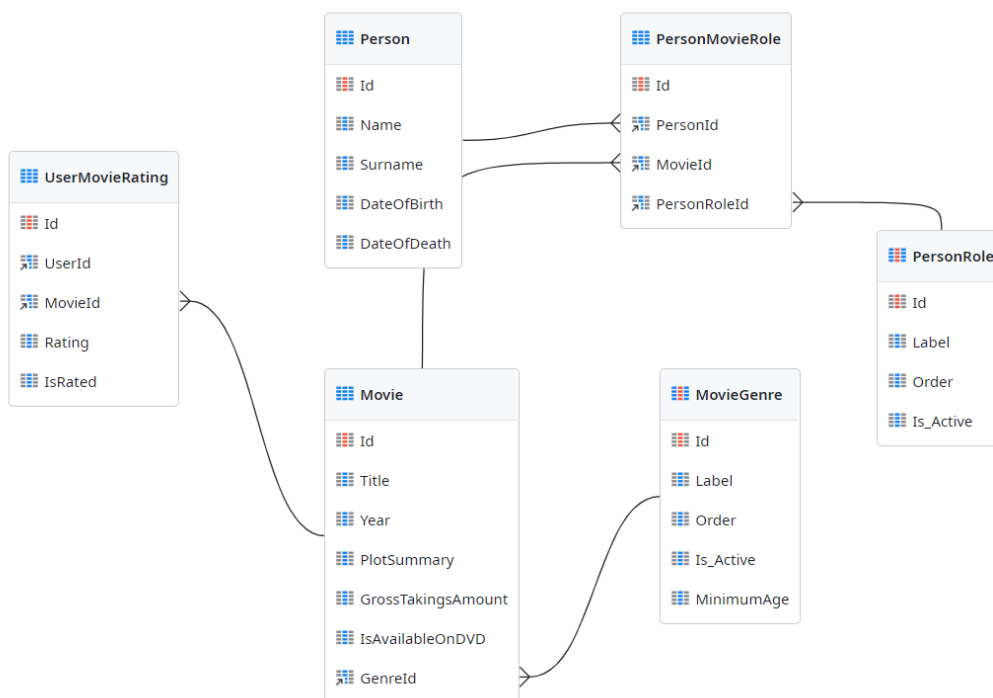
We also want to allow a user to be able to rate the movies, and for that, our data model also needs to follow some rules:

- A User must be able to rate one or more movies.
- This rating can be a thumbs up or thumbs down.
- It's also important to save if the movie is already rated or not.

Conclusion

At the end of the exercise, don't forget to verify that all the attribute's data types are correct, as well as the mandatory property of each attribute.

We can create an Entity Diagram and drag all the Entities there to have a visual representation of the data model. The data model should look like this:



Note: When using many-to-many relationships, we usually want to avoid data repetition. For instance, it is not ok to have the same person, with the same role in the same movie. To avoid that, we often use Unique Indexes in the Entities. In the example of the

PersonMovieRole, we would not want two different PersonMovieRole records in the database with the same PersonId, MovieId, and PersonRoleId. We can find more info about Indexes [here](#).