Entity Relationships - Exercise

Table of Contents

Outline

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

- Every movie should have a genre and multiple movies can have the same genre.
- A Person can have multiple roles in the same 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 application in the future will support a new functionality that will allow users to rate a movie.

To support that, we need to add an extra Entity, **UserMovieRating**, which relates the users of the application with 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 OSMDb_Core module and update the data model with some relationships between the existing Entities and also add additional Entities.

Movie Genres

Let's start with the movie genres. In our application, we want to make sure that a movie can have a genre. Also, we don't want to limit our database to just have 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

Now, 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 :)

Ratings

Now, we want to bring the users of our application (end-users) to the equation. We want to make sure that in the foreseeable future the application will support movie ratings. So, we need to prepare our data model to save that information.

When designing this new extension to the data model, we need to keep in mind the following:

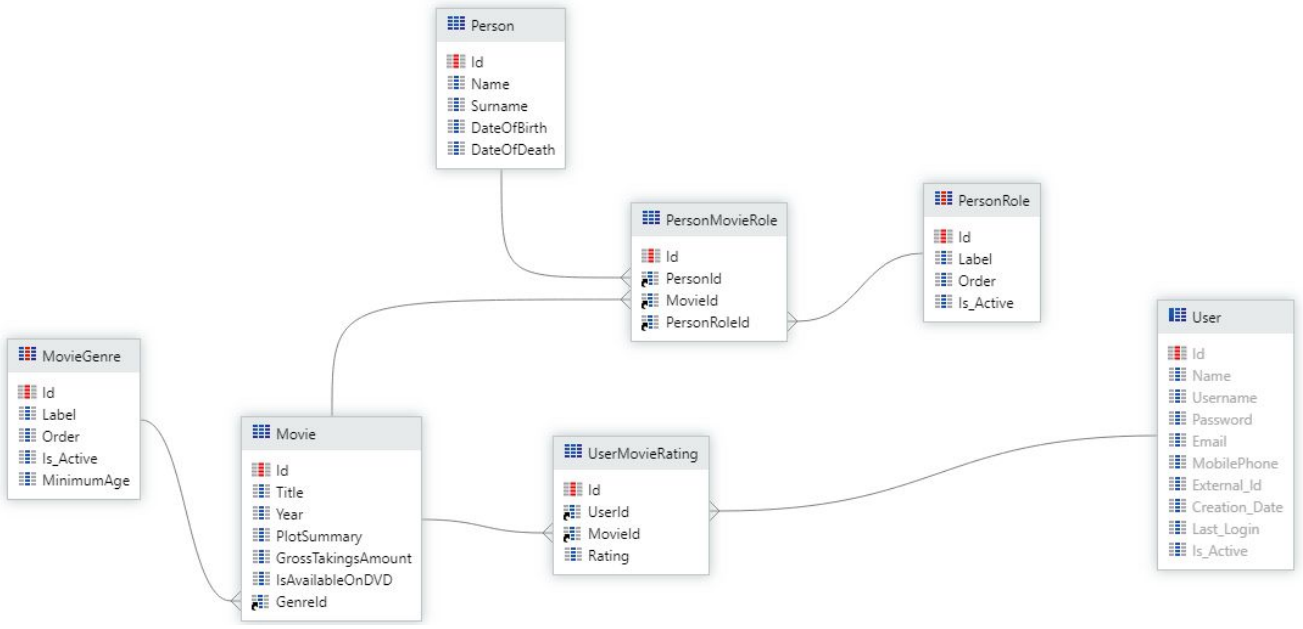
- A user to rate multiple movies.
- A movie to be rated by multiple users.
- To know which user rated a particular movie.

Hint: The (System) User table holds the information of all registered end-users.

Conclusion

At the end of the exercise, don’t forget to verify that all the attribute data types are correct and also the mandatory property of each attribute. Also, the Entities should be public and exposed as read-only.

To have a visualization of the data model, we can create an Entity Diagram and drag all the Entities there. The data model should look like this:



NOTE: When using many-to-many relationships it is often common that we 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.