

Entity Relationships

Exercise - How to

Outline	2
How to	2
Recapping the Data Model	2
Movie Genre Relationship	3
People in Movies	4
User Ratings	5
Entity Diagram	6

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.

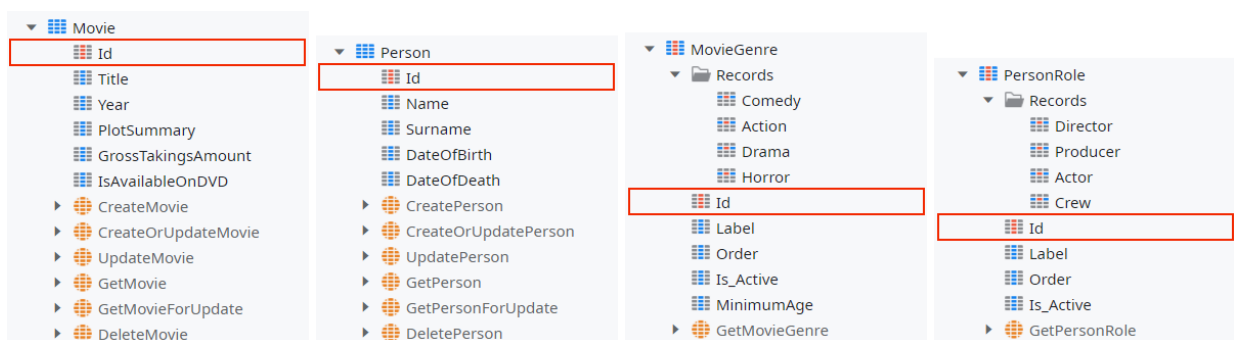
How to

In this section, we'll describe exercise *5.1 – Entity Relationships*, step-by-step.

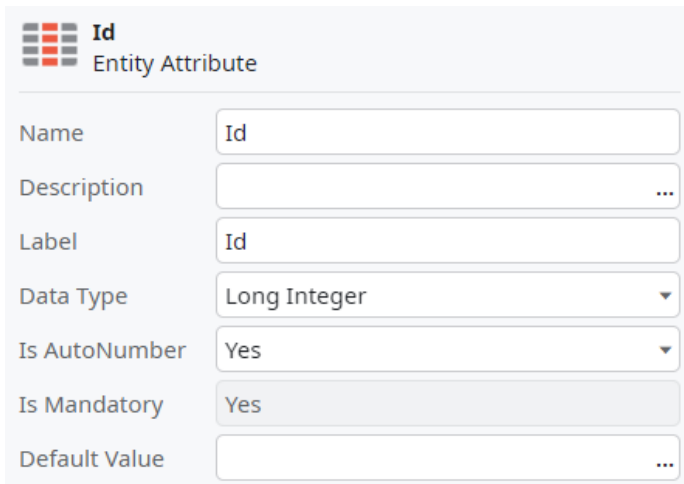
Recapping the Data Model

Let's start by recapping the data model we created two exercises ago.

1. In the OSMDb app, open the **Data** tab.
2. Expand the Movie and Person Entities as well as the MovieGenre and PersonRole Static Entities (if they are currently collapsed). Notice that all the Entities have a default attribute called **Id** that is colored red. This is what is called the Entity's Identifier.



3. Select the **Id** attribute of the Movie Entity and notice that its **Data Type** is *Long Integer* and the **Is AutoNumber** property is set to *Yes*.



Id
Entity Attribute

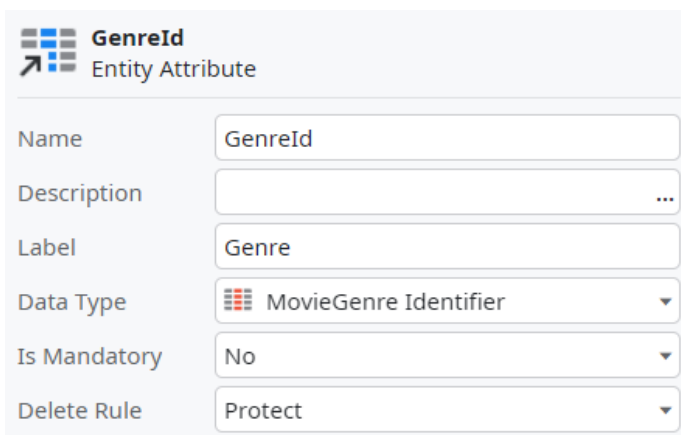
Name	Id
Description	
Label	Id
Data Type	Long Integer
Is AutoNumber	Yes
Is Mandatory	Yes
Default Value	

This means that every new record in the database will have a unique integer identifier that is automatically set by the platform. With all of this in mind, let's start extending the data model.

Movie Genre Relationship

In this section, we will prepare the data model to represent a one-to-many relationship between the MovieGenre and the Movie.

1. Right-click on the Movie Entity and select **Add Entity Attribute**.
2. Set the **Name** of the attribute to *GenreId*.
3. Change the **Data Type** of the GenreId to *MovieGenre Identifier*.



GenreId
Entity Attribute

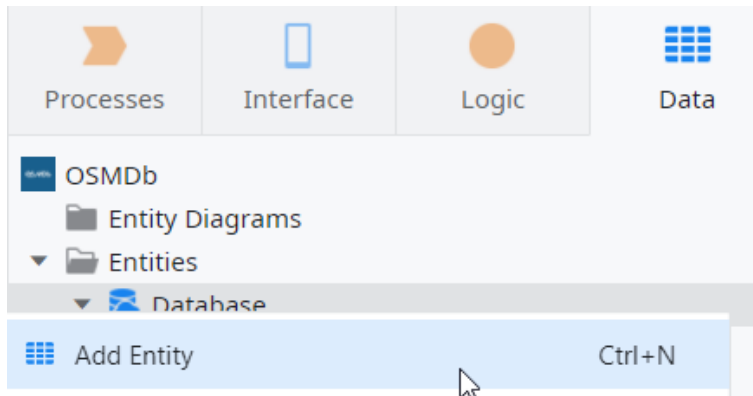
Name	GenreId
Description	
Label	Genre
Data Type	MovieGenre Identifier
Is Mandatory	No
Delete Rule	Protect

This new attribute creates a new column in the Movie database for the genre. This means that now each movie can have a genre.

People in Movies

In this section, we will create a many-to-many relationship between the Person and the Movie Entities. This will be done using a new Entity that, besides the information regarding the movie and the person, will also have the role of the person in that movie.

1. In the **Data** tab, right-click on the Entities folder and select **Add Entity to Database**.



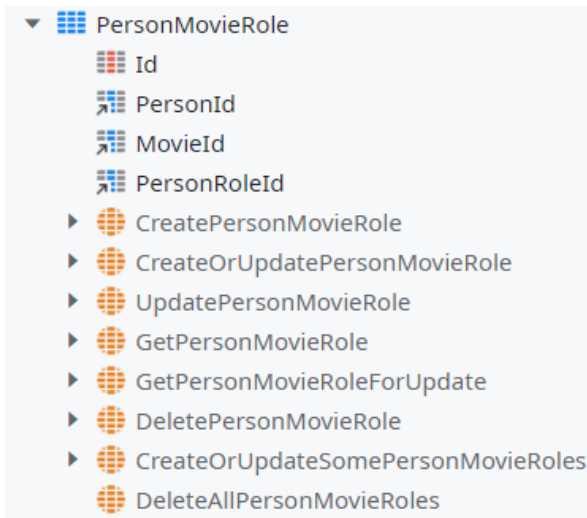
2. Name the new Entity as *PersonMovieRole*.
3. Right-click on the *PersonMovieRole* Entity and select **Add Entity Attribute**. Set its Name to *PersonId* and its **Mandatory** property to *Yes*.
4. Create two new **Mandatory** attributes: *MovieId* and *PersonRoleId*.
5. Verify that the types for these attributes were inferred correctly:

PersonId: *Person Identifier*

MovieId: *Movie Identifier*

PersonRoleId: *PersonRole Identifier*

6. The **PersonMovieRole** Entity should look like this:

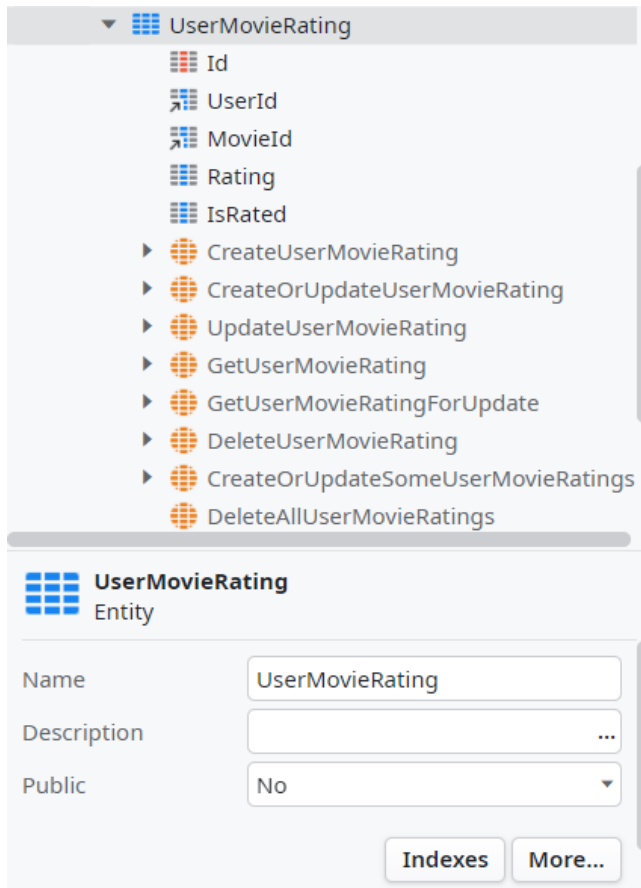


This Entity now has the identifier of the person, the movie, and the role of the person in that movie, creating a relationship between the Entities.

User Ratings

In this section, we will create a many-to-many relationship between a User and the Ratings they can give to a movie. This will also require a new Entity. This Entity needs to save the user information, the movie being rated, the actual rating (thumbs up or thumbs down), and if the movie was already rated. The steps are very similar to the ones in the previous section.

1. Right-click on the **Entities** folder and select **Add Entity to Database**. Enter *UserMovieRating* for the name of the Entity.
2. Add a new *UserId* attribute to the UserMovieRating Entity. Leave its **Mandatory** property set to No. Verify if the **Data Type** is set to *User Identifier*.
3. Add a new *MovieId* attribute to the UserMovieRating Entity. Set its **Mandatory** property as Yes. Verify if the **Data Type** is set to *Movie Identifier*.
4. Add a new *Rating* attribute to the UserMovieRating Entity. Set its **Mandatory** property to Yes and set its **Data Type** to *Boolean*.
5. Add a new *IsRated* attribute to the UserMovieRating Entity. Set its **Mandatory** property to Yes and set its **Data Type** to *Boolean*.
6. Your Entity should look like this:

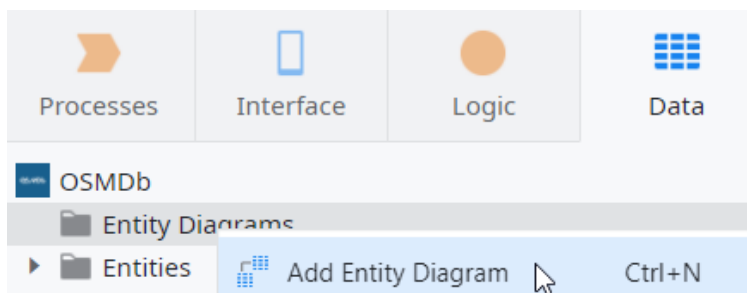


7. Publish the app to save the new version of the app.

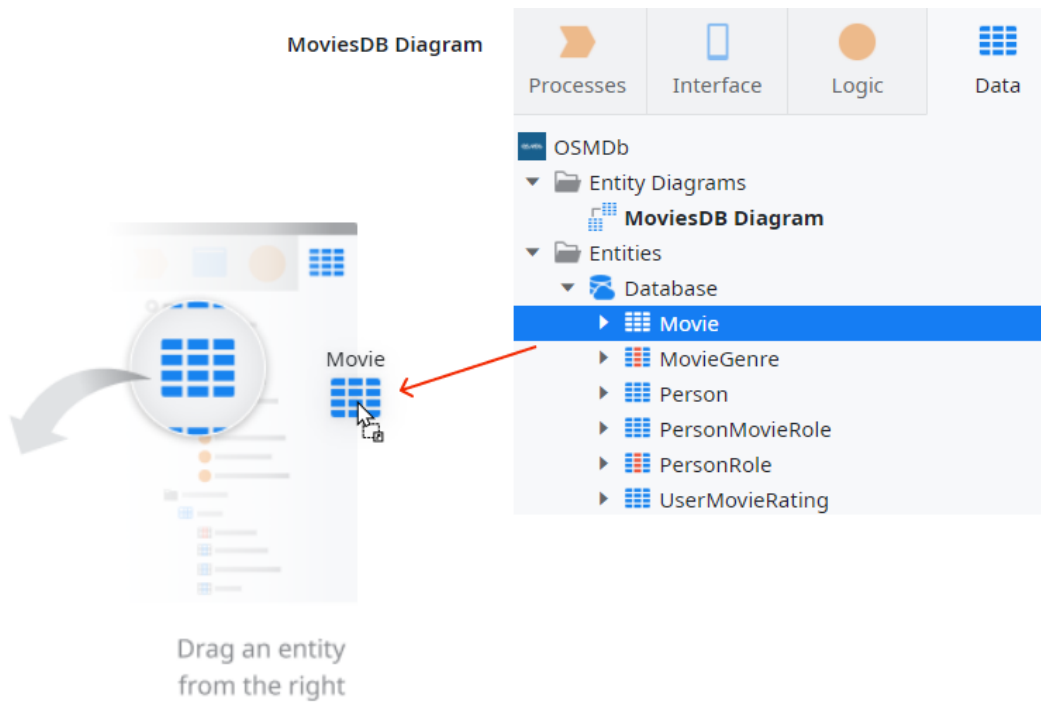
Entity Diagram

The data model is done, now let's take a look at a very useful feature, the entity diagram, that allows us to have a visual representation of the data model. So, let's create an Entity Diagram with all the Entities in it.

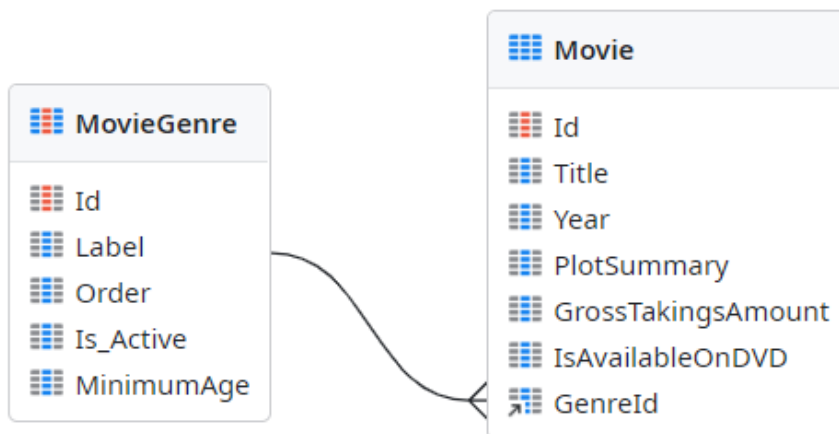
1. In the **Data** tab, right click the **Entity Diagrams** folder and select **Add Entity Diagram**.



2. Name this diagram *MoviesDB Diagram*.
3. Drag and drop the **Movie** Entity into the *MoviesDB Diagram* canvas.

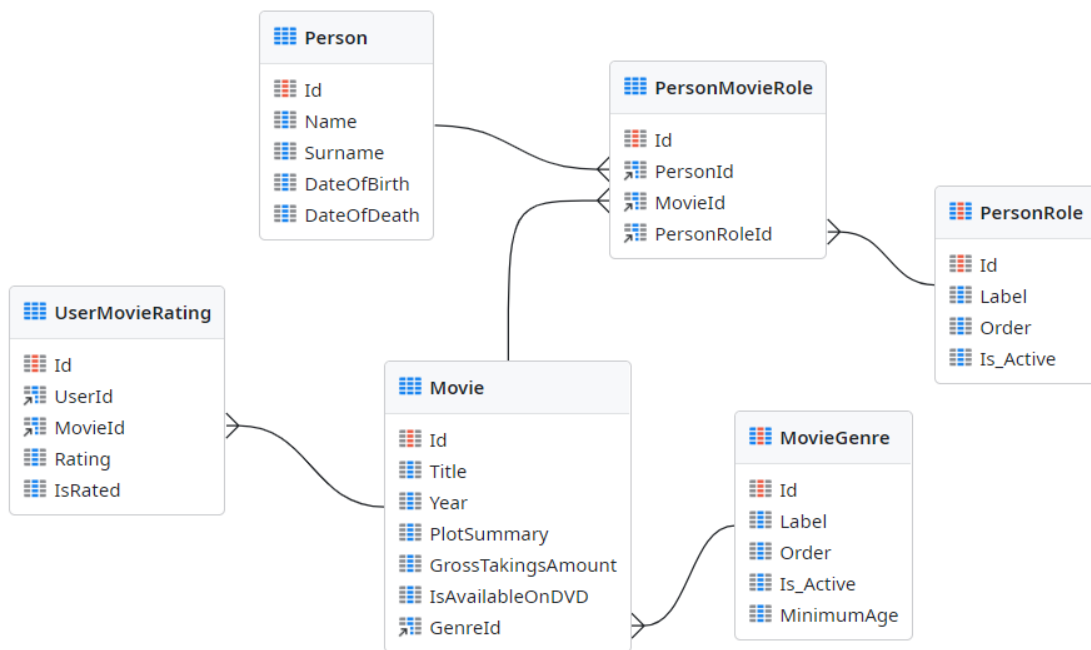


4. Drag and drop the **MovieGenre** Static Entity to the diagram.



Notice there's a connector linking the **GenreId** attribute to the MovieGenre Static Entity, highlighting the one-to-many relationship.

5. Add the **Person**, **PersonMovieRole**, and **UserMovieRating** Entities and **PersonRole** Static Entity to the Entity Diagram. The Diagram should look somewhat like this:



6. Publish your app to save the latest changes! At this time, there are no visible changes in the app, so in the next exercise, we will start using these new Entities.

