

# Modeling Data Relationships Exercise

## Table of Contents

<b>Outline.....</b>	<b>2</b>
Resources	2
Scenario	2
1-to-Many Relationship	3
1-to-1 Relationship	3
Many-to-Many Relationship	3
<b>How-To.....</b>	<b>5</b>
Getting Started	5
1-to-Many Relationship	9
1-to-1 Relationship	10
Many-to-Many Relationship	15

# Outline

In this exercise, we will focus on data relationships and how we can define them while building our data model in OutSystems. There are three different types of data relationships in OutSystems, which we will have the chance to create during the exercise:

- 1-to-1
- 1-to-Many
- Many-to-Many

Upon completion, we will end up with a data model with five Entities, with all of them having relationships with other Entities, and with all types of relationships represented.

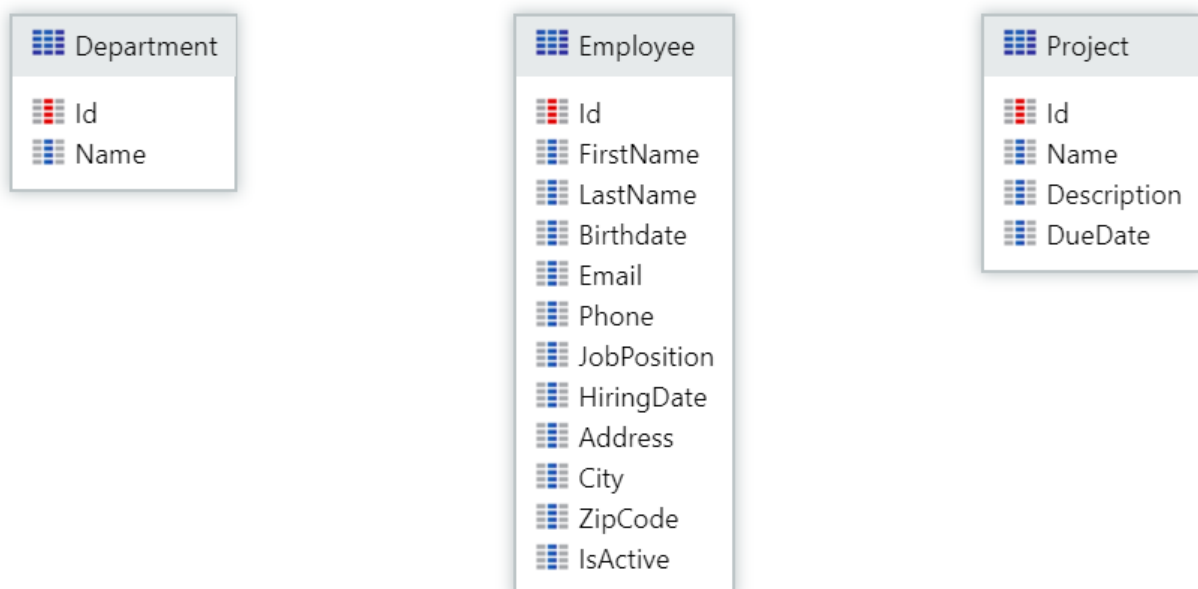
## Resources

This exercise has a Quickstart application already created. This application has everything needed to start the exercise. This quickstart application can be found in the Resources folder of this exercise, with the name **Data Relationships Exercise.oap**.

## Scenario

The data model that we're going to work on supports an Employee Services application, representing the Employees of an Organization, the Departments, and the Projects.

The starting point for our data model are the Employee, Department, and Project Entities.



These Entities are already created, with all attributes being mandatory, and the data was bootstrapped from an Excel file.

We can now start creating a richer data model. Let's focus on Employees!

## 1-to-Many Relationship

An Employee works in one of the organization's Departments. On the other hand, a Department can have multiple Employees, making this a scenario where a one-to-many relationship makes sense.

In the first part of this exercise, we need to create the 1-to-many relationship between the Employee and the Department Entity, with the Department being the Master Entity and the Employee the Detail Entity of the relationship.

## 1-to-1 Relationship

A second requirement for this data model is to have a picture associated with every Employee. We want to make sure that, in the future, the application will allow other employees to search for co-workers, which makes the picture a valuable visual asset. An Employee will only have one picture, while a picture can only be relative to an Employee.

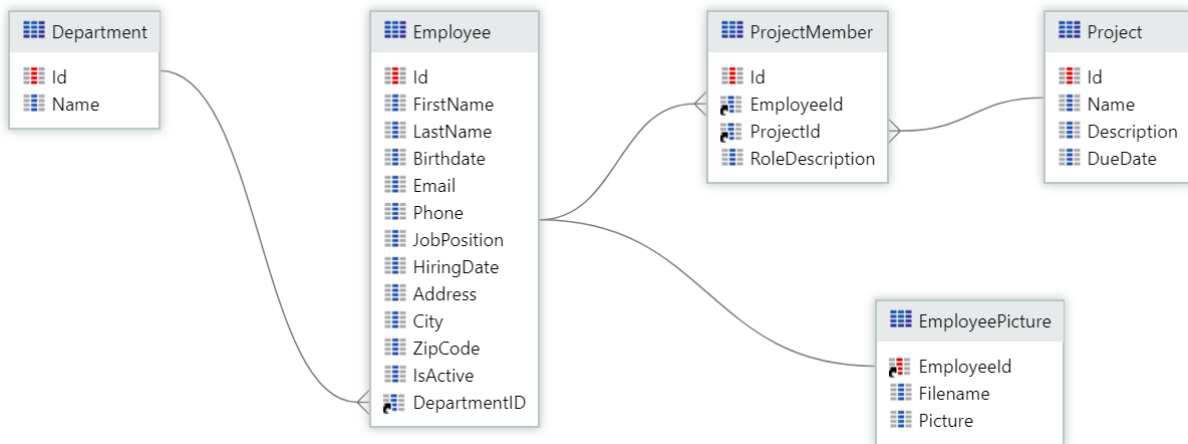
In the second part of the exercise we need to create an Extension Entity, EmployeePicture, with a 1-to-1 relationship to the Employee Entity. Besides its identifier, the Employee Picture Entity will have a Filename attribute (Data Type: Text, mandatory) and a Picture attribute (Data Type: Binary Data, mandatory).

## Many-to-Many Relationship

The final requirement for this exercise is to represent Project Members. Our scenario will require that several Employees will be associated to a Project. Also, an Employee can be part of different projects. This requires a many-to-many relationship between the Employee and the Project Entities.

In the final part of the exercise, we need to create the relationship between these two Entities. For that, we need a Junction Entity, ProjectMember, which will require at least the RoleDescription of the Employee on the Project (Data Type: Text, mandatory).

The final data model should look like this:

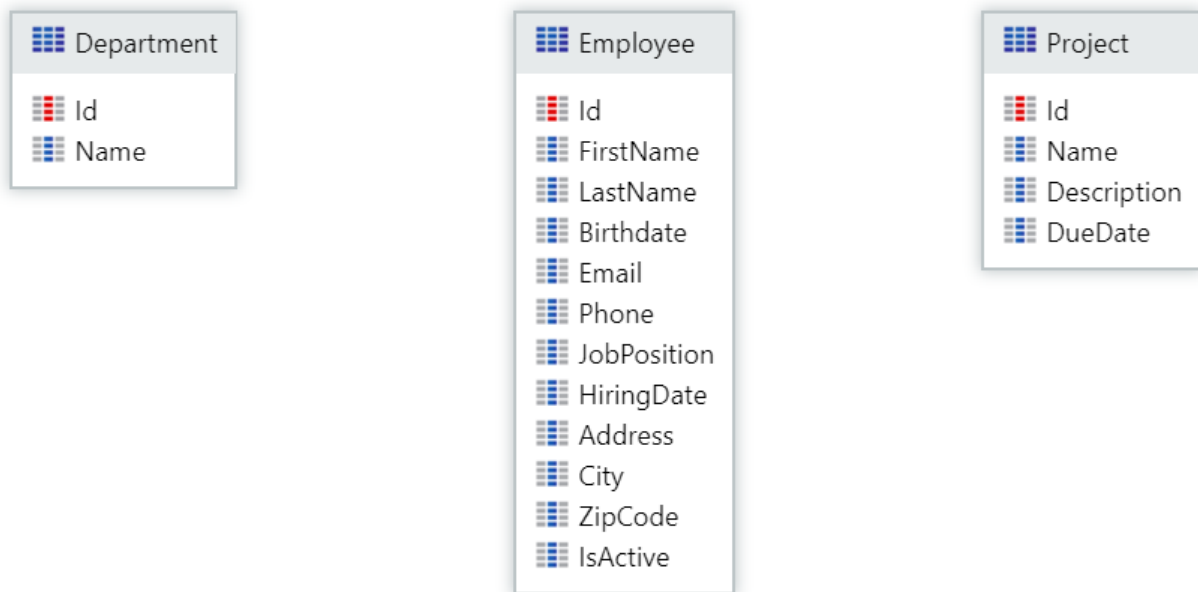


## How-To

In this section, we will show you how to do this exercise, with a thorough step-by-step description. **If you already finished the exercise on your own, great! You don't need to do it again.** If you didn't finish the exercise, that's fine! We are here to help you.

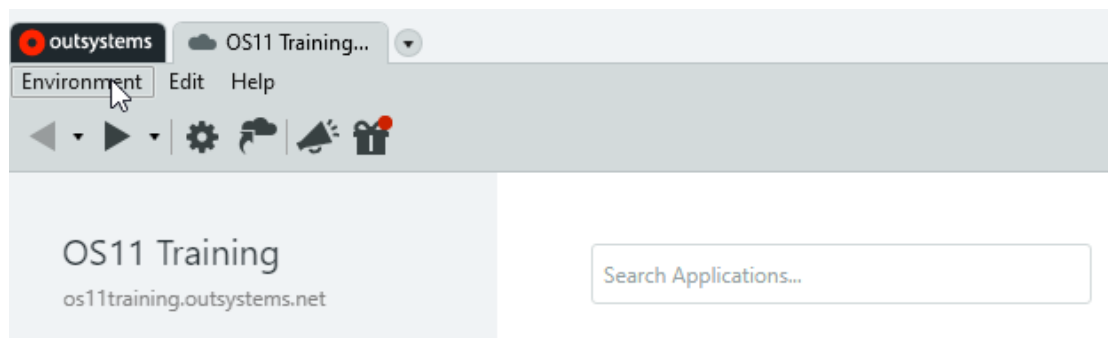
### Getting Started

To get started with this exercise, we need to install the Quickstart file, which has the application already created with the three Entities: Employee, Department, and Project.

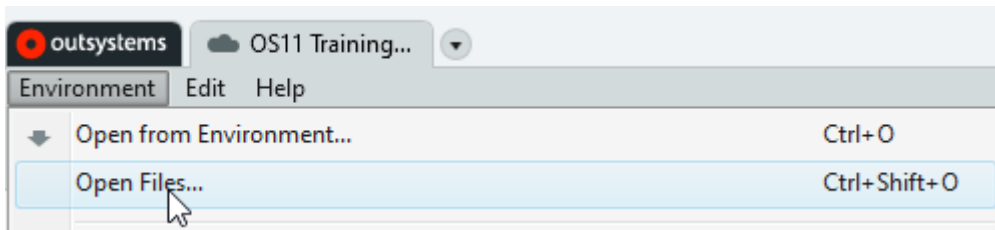


The first step that we need to take is to install the Quickstart application in our development environment. Before proceeding, you must have Service Studio open and connected to an OutSystems Environment (e.g. Personal Environment).

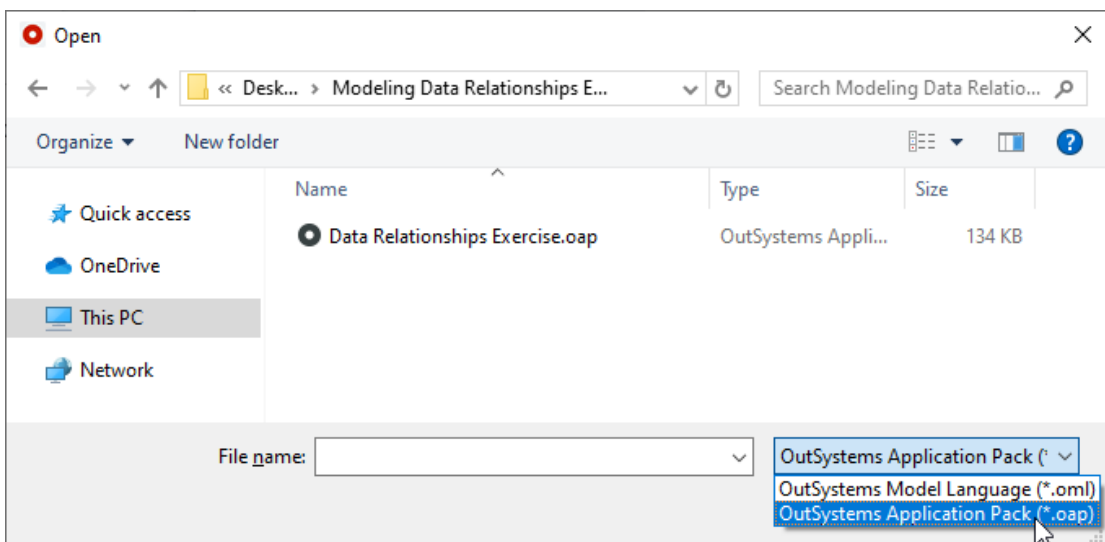
- 1) In Service Studio's main window, select the **Environment** menu on the top left.



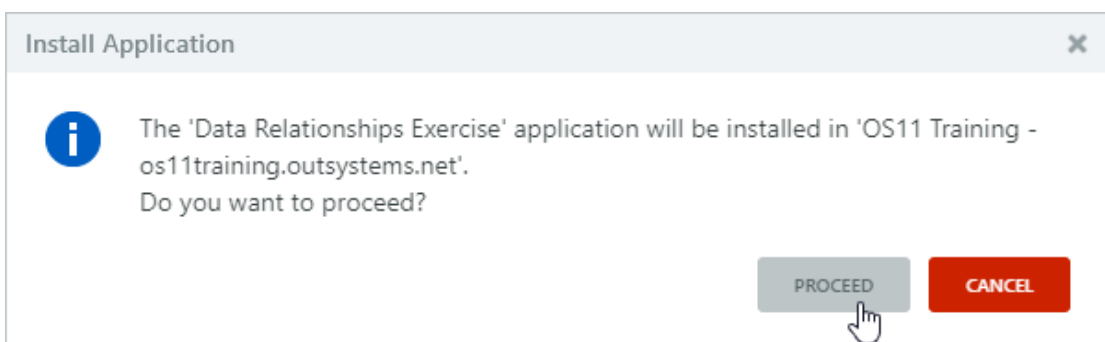
- 2) Select **Open Files...**



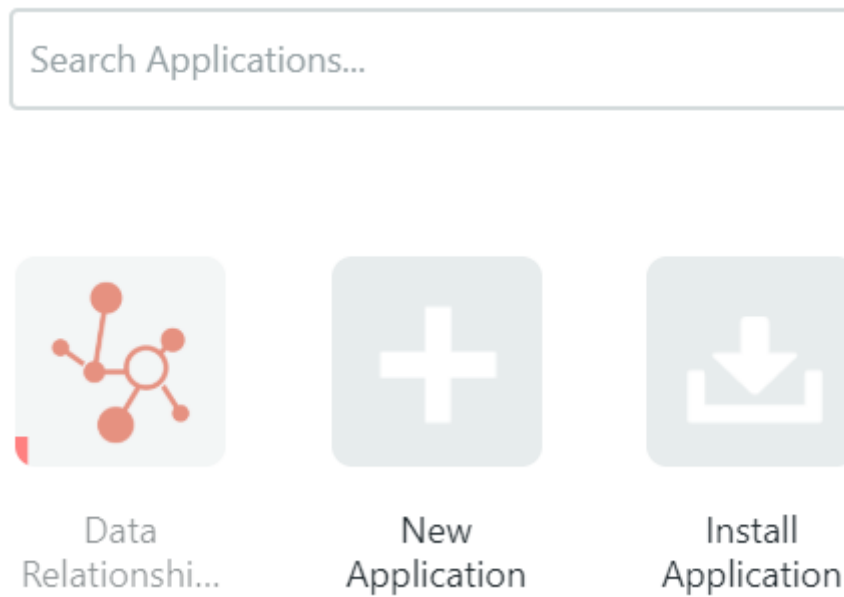
- 3) In the following dialog, change the file type to OutSystems Application Pack (.oap), find the location of the Quickstart file and open the file named **Data Relationships Exercise.oap**.



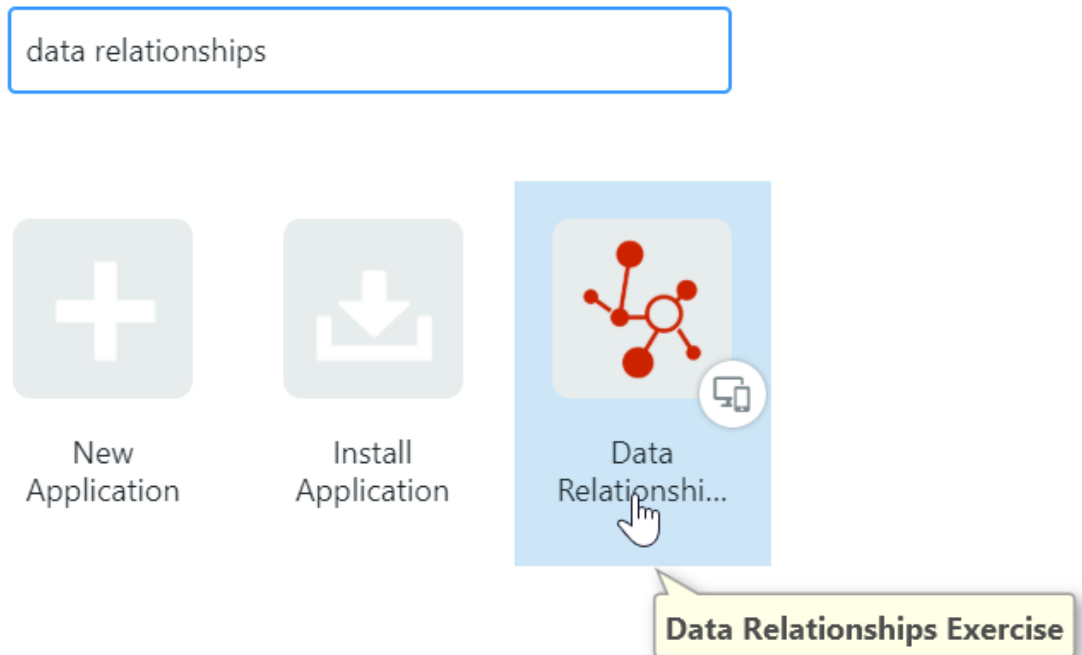
- 4) In the confirmation dialog, select **Proceed**.



- 5) The application will begin installing automatically. When it's finished, we're ready to start!




- 6) Open the application in Service Studio.



7) The application has only one module. Let's open it!

Applications in OS11 Training > Data Relationships Exercise

## Data Relationships Exercise



EDIT DELETE DOWNLOAD OPEN IN BROWSER

Develop

Modules

Modules allow you to structure your application into several pieces, each piece implementing a specific purpose.

DataRelationshipsExercise Changed yesterday 4:13 PM by OutSystems Developer

ADD MODULE

8) Under the module's Data, we can see the three Entities already created.

Processes Interface Logic Data

Diagram 1

Department

- Id
- Name

Employee

- Id
- FirstName
- LastName
- Birthdate
- Email
- Phone
- JobPosition
- HiringDate
- Address
- City
- ZipCode
- IsActive

Project

- Id
- Name
- Description
- DueDate

DataRelationshipsExercise

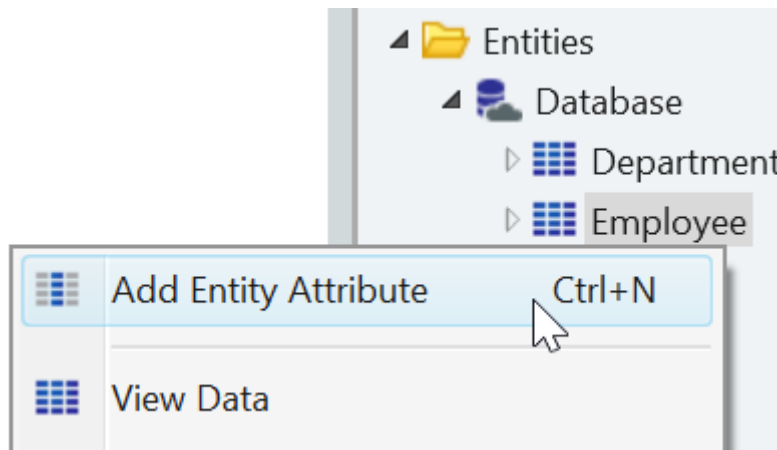
- Entity Diagrams
  - Diagram 1
- Entities
  - Database
    - Department
    - Employee
    - Project
    - (System)
  - Structures
  - Client Variables
  - Site Properties
  - Resources



## 1-to-Many Relationship

In this section, we will expand our data model by creating a 1-to-Many relationship between the Employee and the Department Entities. Let's go back to Service Studio and create the relationship.

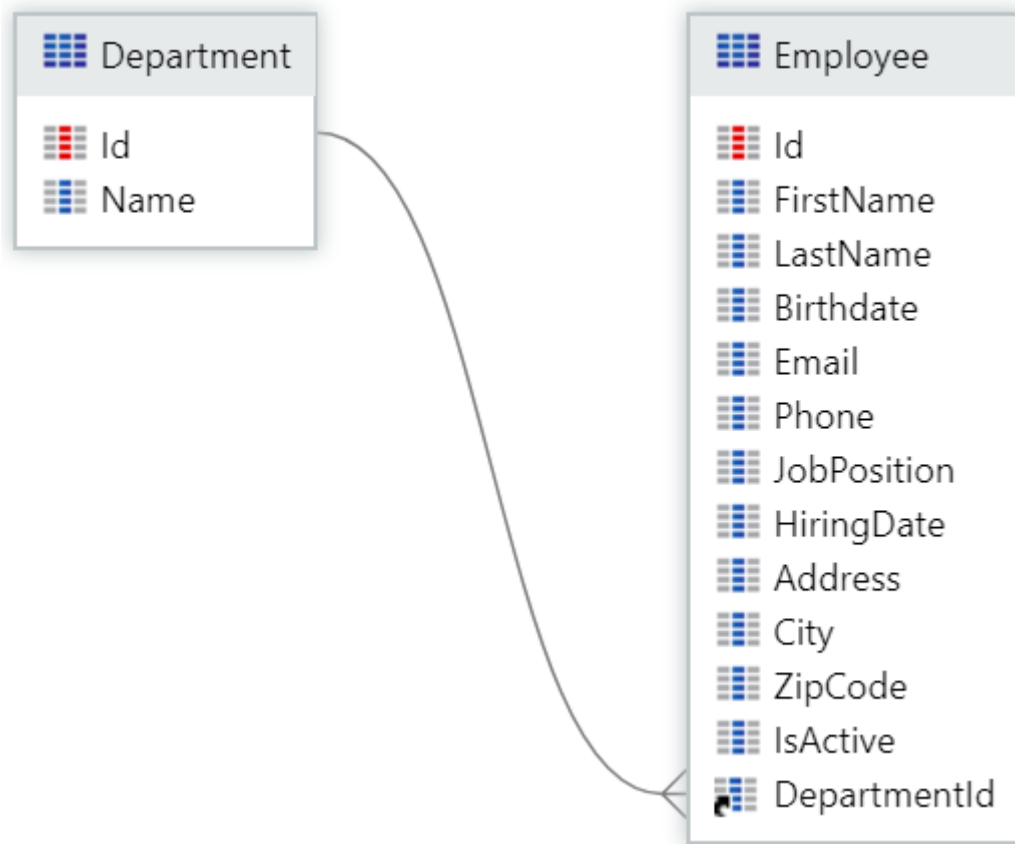
- 1) Under the Entities folder on the right, right-click the **Employee** Entity and select **Add Entity Attribute**.



- 2) Select the newly created attribute (called Attribute1) and set its name to *DepartmentId*. Make sure its **Data Type** is set to *Department Identifier*.

DepartmentId Entity Attribute	
Name	DepartmentId
Description	...
Label	Department
Data Type	Department Identifier ▼
Is Mandatory	No ▼
Delete Rule	Protect ▼

- 3) Notice in this Entity Diagram that a visual representation of the relationship was created. This means that the relationship between the two Entities is defined.



- 4) Publish the module to save the changes to the server

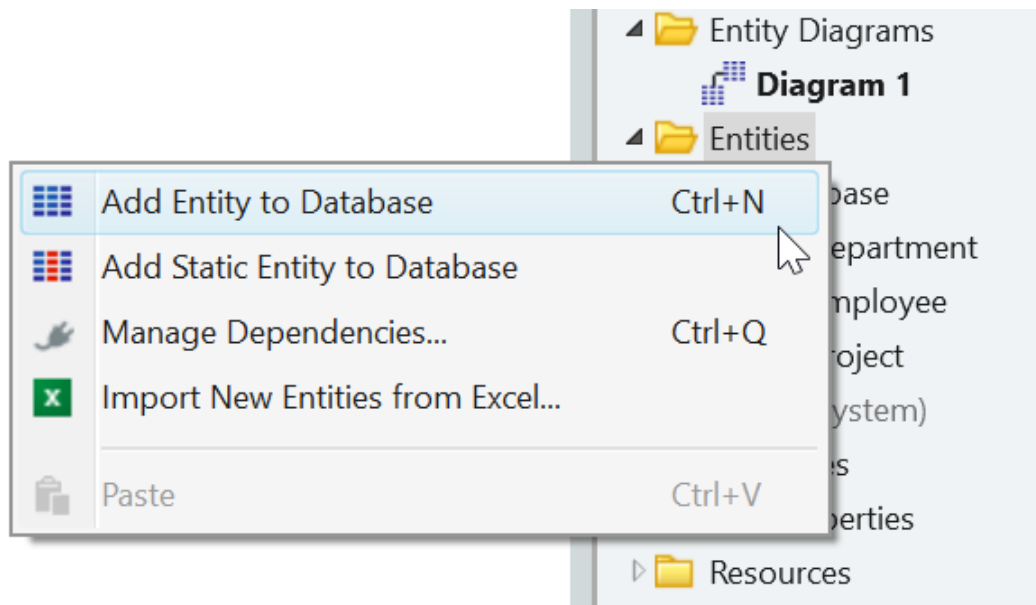


## 1-to-1 Relationship

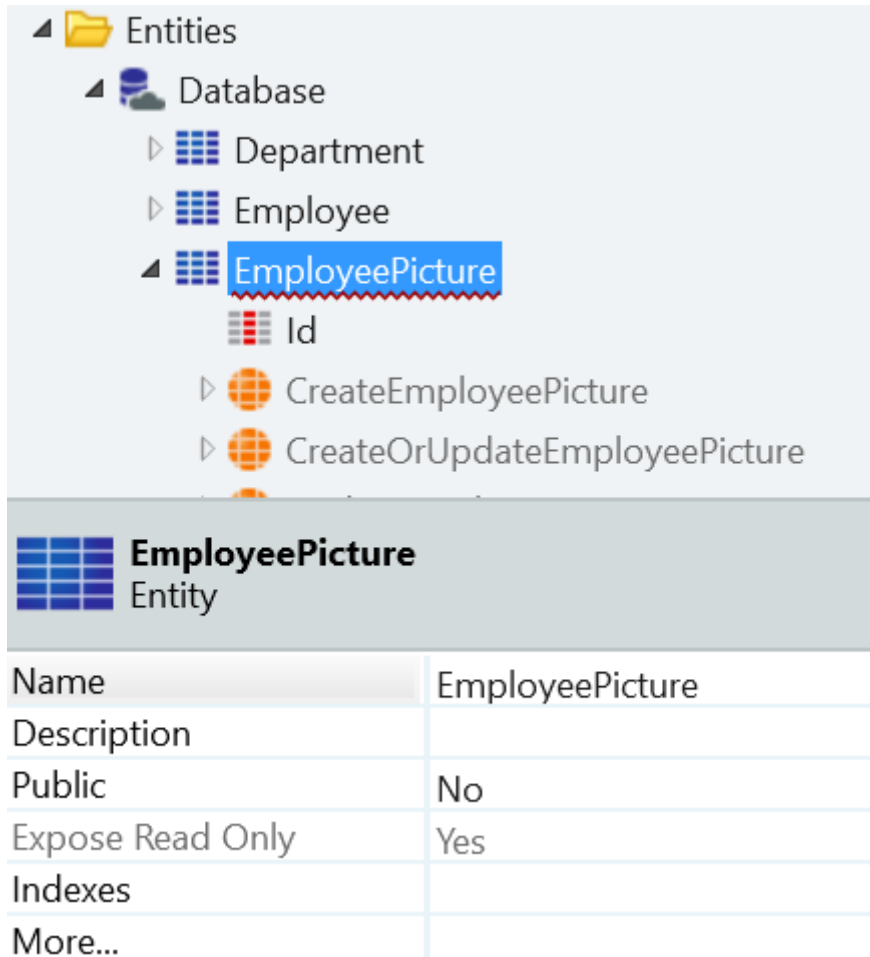
Now, we want to expand our data model by adding the Employee picture. We need to create a new Entity, **EmployeePicture**, with three attributes: **EmployeeId** (Employee

Identifier, mandatory), Filename (Text, mandatory), and Picture (Binary Data, mandatory).

- 1) In Service Studio, right-click the Entities folder on the right, and select **Add Entity to Database**.



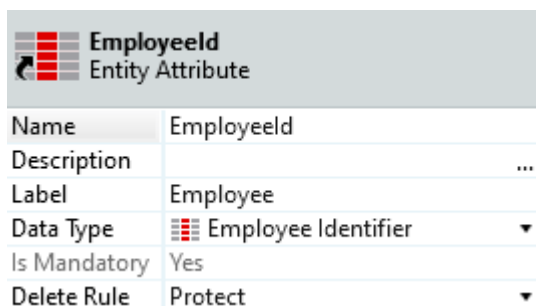
- 2) Set the name of the new Entity to *EmployeePicture*. There's an error indicating that the Entity can't be exclusively composed of an Auto-Number attribute (**Id**). We will fix that in the next steps.



The screenshot shows the OutSystems IDE interface. On the left, a tree view displays the project structure: **Entities** (folder) contains **Database** (folder), which contains **Department**, **Employee**, and **EmployeePicture** (Entity). The **EmployeePicture** Entity has an attribute **Id** (Auto-Number) and two actions: **CreateEmployeePicture** and **CreateOrUpdateEmployeePicture**. Below the tree, the details for the **EmployeePicture** Entity are shown in a table.

EmployeePicture Entity	
Name	EmployeePicture
Description	
Public	No
Expose Read Only	Yes
Indexes	
More...	

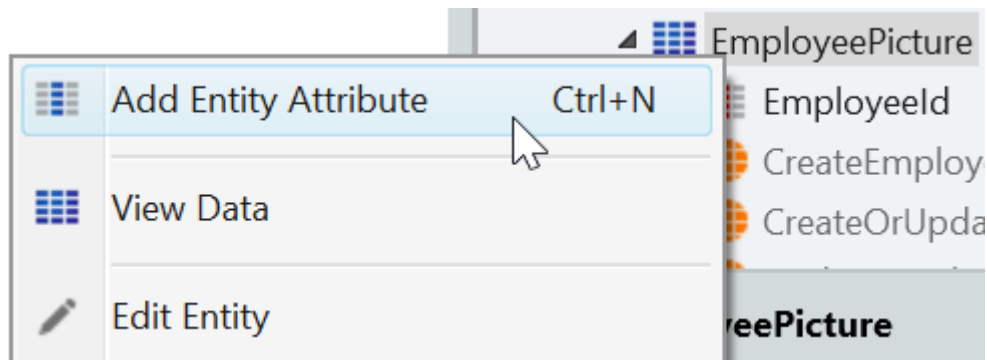
- 3) Select the **Id** attribute of the EmployeePicture Entity and change its name to *EmployeeId*. Notice that the data type automatically changed to *Employee Identifier*.



The screenshot shows the details for the **EmployeeId** Entity Attribute. The attribute is named **EmployeeId** and is of type **Employee Identifier**. The data type has been automatically changed to *Employee Identifier*.

EmployeeId Entity Attribute	
Name	EmployeeId
Description	...
Label	Employee
Data Type	Employee Identifier
Is Mandatory	Yes
Delete Rule	Protect

- 4) Right-click the **EmployeePicture** entity and select **Add Entity Attribute**.



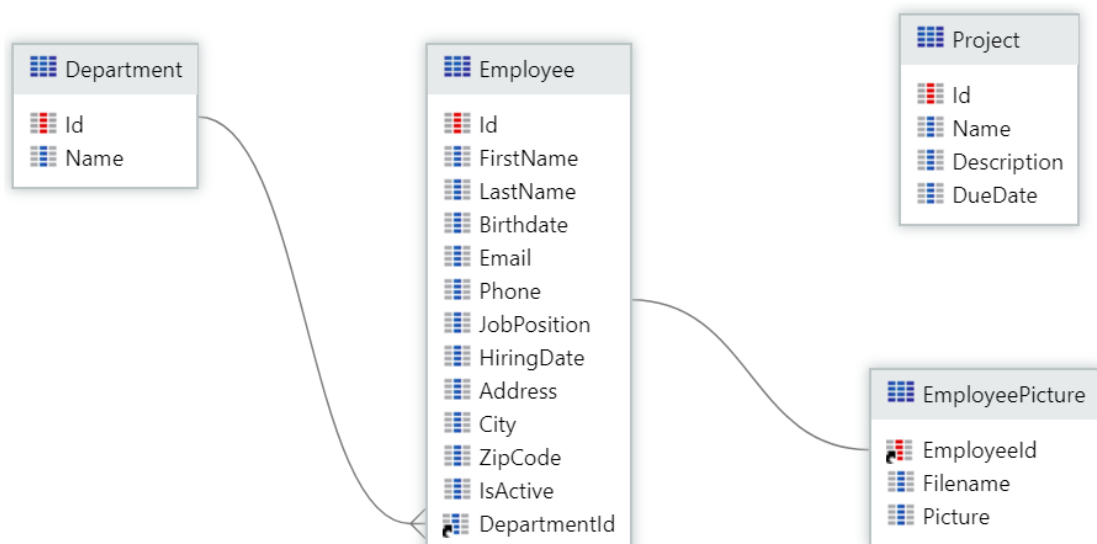
- 5) Set the name of the new attribute to *Filename* and make sure its **Data Type** is set to *Text*. Set the **Is Mandatory** property to *Yes*.

Filename Entity Attribute	
Name	Filename
Description	
Label	Filename
Data Type	Text
Length	50
Is Mandatory	Yes

- 6) Following the same process, add a new attribute to the EmployeePicture Entity and set its **Name** to *Picture*. Make sure its **Data Type** is set to *Binary Data* and the **Is Mandatory** property is set to *Yes*.

EmployeePicture Entity Attributes	
Name	Picture
Description	
Label	Picture
Data Type	Binary Data
Is Mandatory	Yes

- 7) Drag and drop the EmployeePicture Entity to the Entity Diagram on the left and make sure that there's a relationship visible between the Employee Entity and the EmployeePicture.



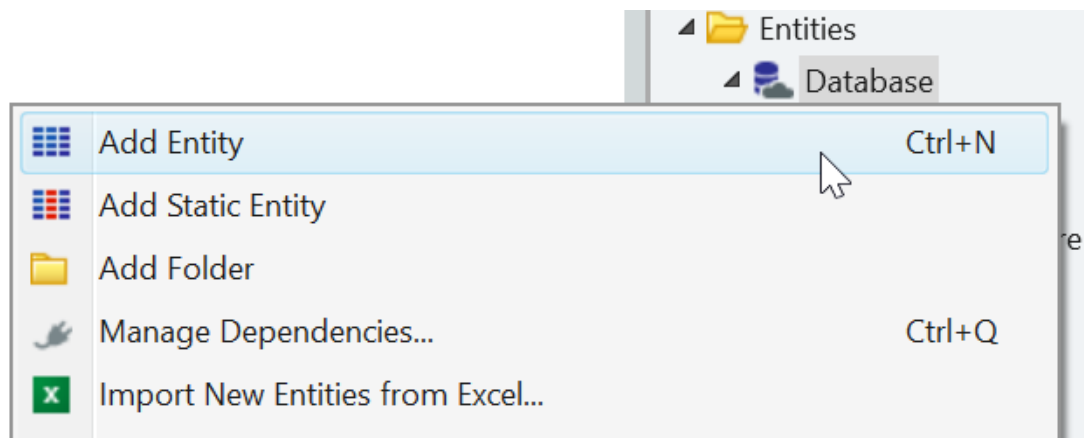
- 8) Publish the module to save the changes to the server



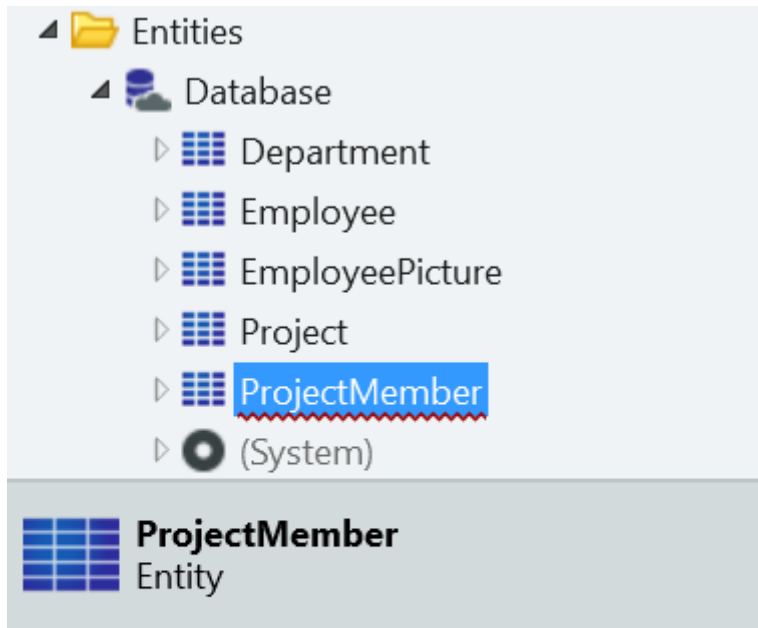
## Many-to-Many Relationship

Let's create a new Entity, ProjectMember, to support the many-to-many relationship between the Employee and Project Entities. This new Entity will have three new attributes: EmployeeId (Employee Identifier, mandatory), ProjectId (Project Identifier, mandatory) and RoleDescription (Text, mandatory).

- 1) Right-click the **Database** element, under the Entities folder, and select **Add Entity**.



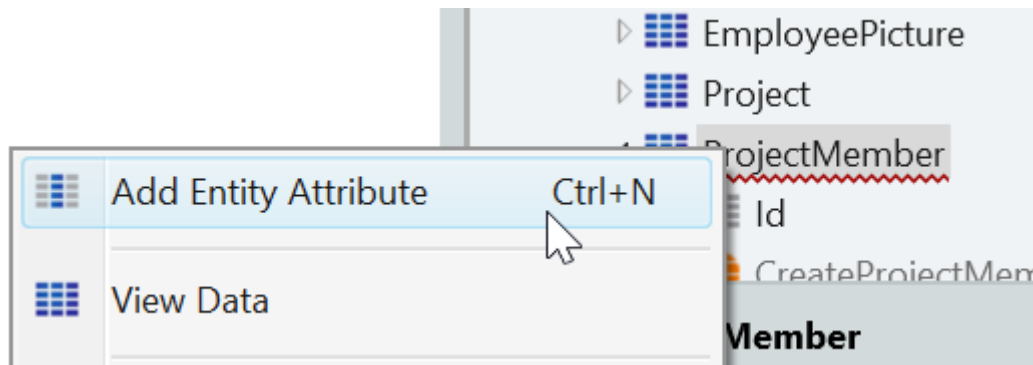
- 2) Set the **Name** of the new Entity to *ProjectMember*.



The screenshot shows the OutSystems Entities console. Under the 'Database' folder, the 'ProjectMember' entity is selected and highlighted with a blue border and a red wavy underline. Below the console, the 'ProjectMember Entity' properties are displayed in a table.

Name	ProjectMember
Description	
Public	No
Expose Read Only	Yes

- 3) Right-click the **ProjectMember** Entity and select **Add Entity Attribute**.



The screenshot shows a right-click context menu for the 'ProjectMember' entity. The 'Add Entity Attribute' option is highlighted, and the keyboard shortcut 'Ctrl+N' is displayed next to it. Other options visible in the menu include 'View Data'.



- 4) Set the **Name** of the new attribute to *EmployeeId*. Make sure its **Data Type** was automatically set to *Employee Identifier* and set its **Is Mandatory** property to *Yes*.

ProjectMember

- Id
- EmployeeId

**EmployeeId**  
Entity Attribute

Name	EmployeeId
Description	
Label	Employee
Data Type	Employee Identifier
Is Mandatory	Yes
Delete Rule	Protect

- 5) Repeat the process to add the *ProjectId* attribute. Make sure its **Data Type** was set to *Project Identifier* and make it mandatory.

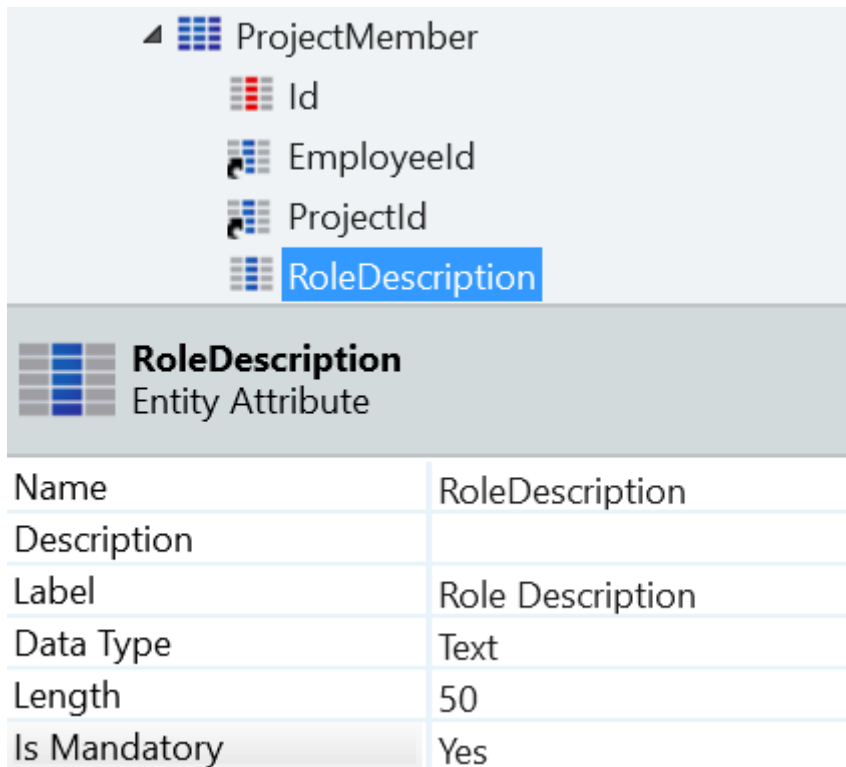
ProjectMember

- Id
- EmployeeId
- ProjectId

**ProjectId**  
Entity Attribute

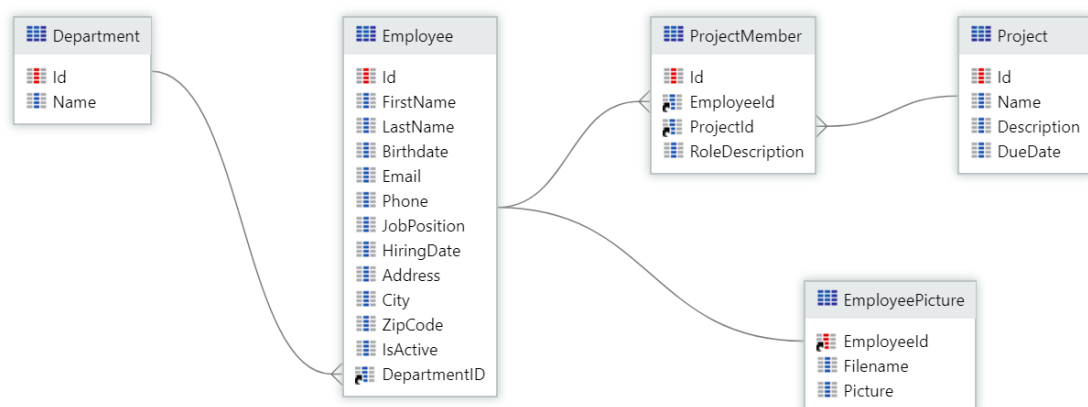
Name	ProjectId
Description	
Label	Project
Data Type	Project Identifier
Is Mandatory	Yes
Delete Rule	Protect

- 6) Add a final attribute named *RoleDescription*. Set its **Data Type** to *Text* and make the attribute mandatory.



Name	RoleDescription
Description	
Label	Role Description
Data Type	Text
Length	50
Is Mandatory	Yes

- 7) Drag and drop the ProjectMember Entity to the EntityDiagram, between the Employee and the Project Entities. Make sure the relationships between the Entities are visible.



- 8) Publish the module to save the changes to the server



Note that, at this point, there is no way to test if these relationships work. The new Entities and attributes do not have any values yet, since they were just recently created. We would need to create some logic and UI that would enable adding members to projects, relate employees with departments, or adding employee pictures. However, that is not part of the scope of this exercise.