Using Custom SQL Queries Exercise

Table of Contents

knowledge@outsystems.com

www.outsystems.com

Outline

In this exercise, we will create a series of custom SQL Queries that will allow us to retrieve, update, and delete data in existing entities. This exercise requires some work in Service Studio and also some testing in the browser.

In the application, we have a set of employees, and throughout this exercise we want to leverage the SQL Tool to:

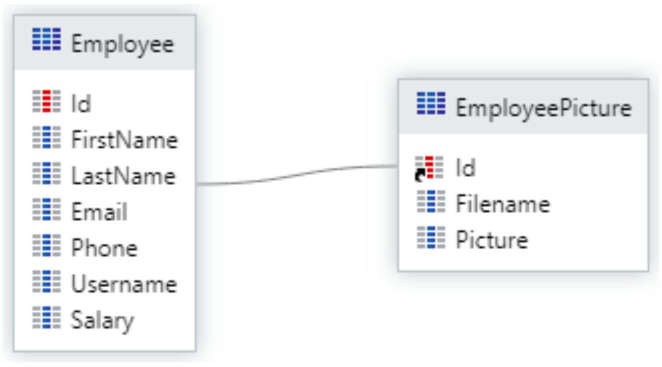
- Retrieve data using a SELECT SQL Query (instead of an Aggregate) with a filtering criteria
- Delete data using a DELETE SQL Query
- Update data in an Entity based on an input parameter

Resources

This exercise has a Quickstart application already created. This application has everything needed to start the exercise. The Quickstart application can be found in the Resources folder of this exercise, with the name **Using Custom SQL Queries Exercise.oap**.

Scenario

In this exercise, we'll start from an existing application with one module. Inside that module, we have a data model already defined with two Entities. The module has the logic to import data from Excel to bootstrap the existing Entities (Employee and EmployeePicture) with data. So, when the application (and its module) is installed, the data will be automatically bootstrapped.



The module also has a screen named **Employees**. This screen shows a list of all the employees, and it also contains a text input to filter the list of employees. The screen also contains three buttons at the bottom:

- A button to update the salary of the employees

- A button to delete the employees data
- A button to reload the data into the entities

Although the skeleton of the update and delete buttons is already built (Client Actions and Server Actions), they are not fully implemented. Both the update and the delete buttons' behavior will be implemented in this exercise.

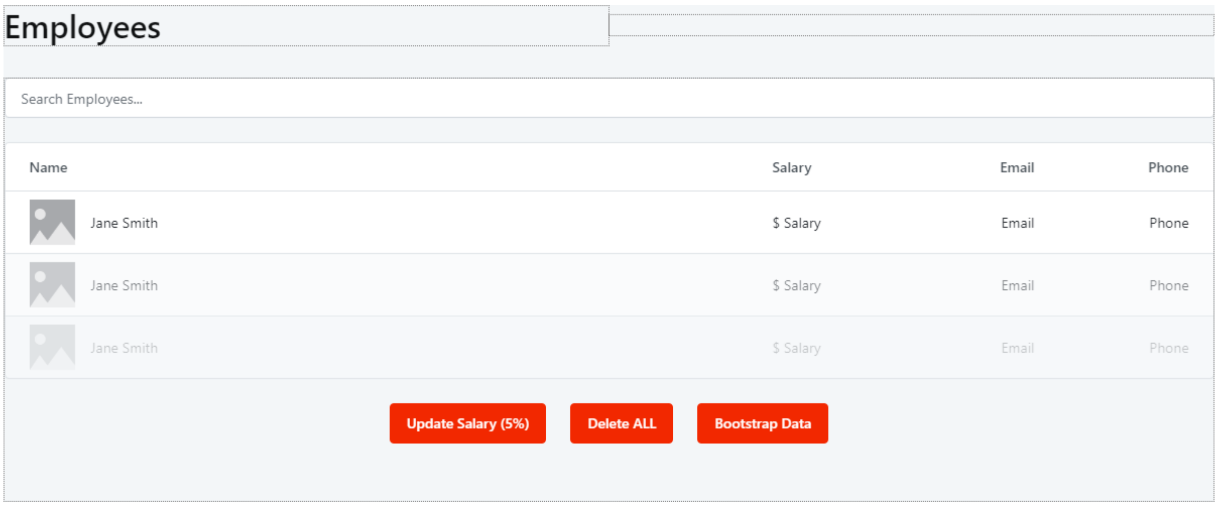
In this exercise, we will start by replacing the existing *GetEmployees* Aggregate by a SELECT SQL Query that will provide the same behavior. Later on, we will implement a DELETE SQL Query to delete all employees. Finally, we will create an UPDATE SQL Query to increase the salary of all employees by a specific percentage.

How-To

In this section, we'll 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.

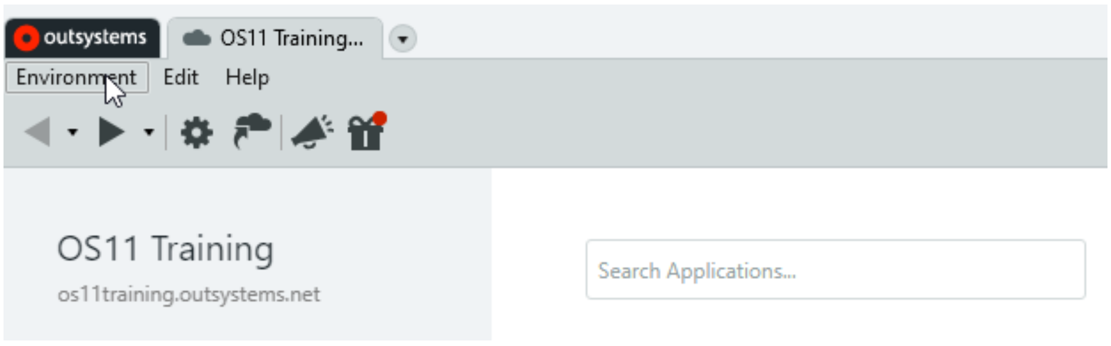
Getting Started

To start this exercise, we need to install the Quickstart file, **Using Custom SQL Queries Exercise.oap**. This file has an application with one Screen, named Employees. The Screen contains a Table to display the existing employees that are retrieved from the existing entities. The list of employees is obtained using an Aggregate, which will be replaced by a SELECT SQL Query. The Screen also contains set of buttons that you will use to execute other SQL Queries that you will create.

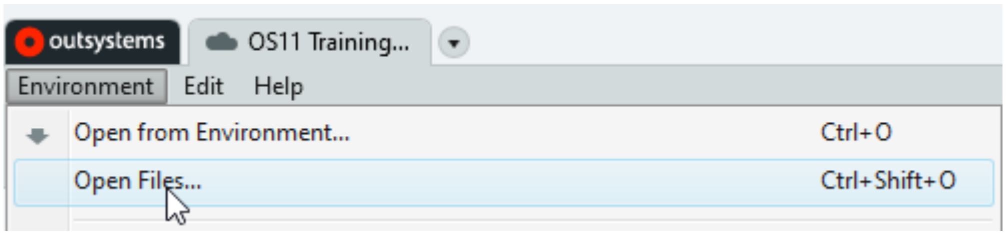


The first step we need to take is to install the Quickstart application in our development environment. Before proceeding, you must have Service Studio opened and connected to an OutSystems Environment (e.g. Personal Environment).

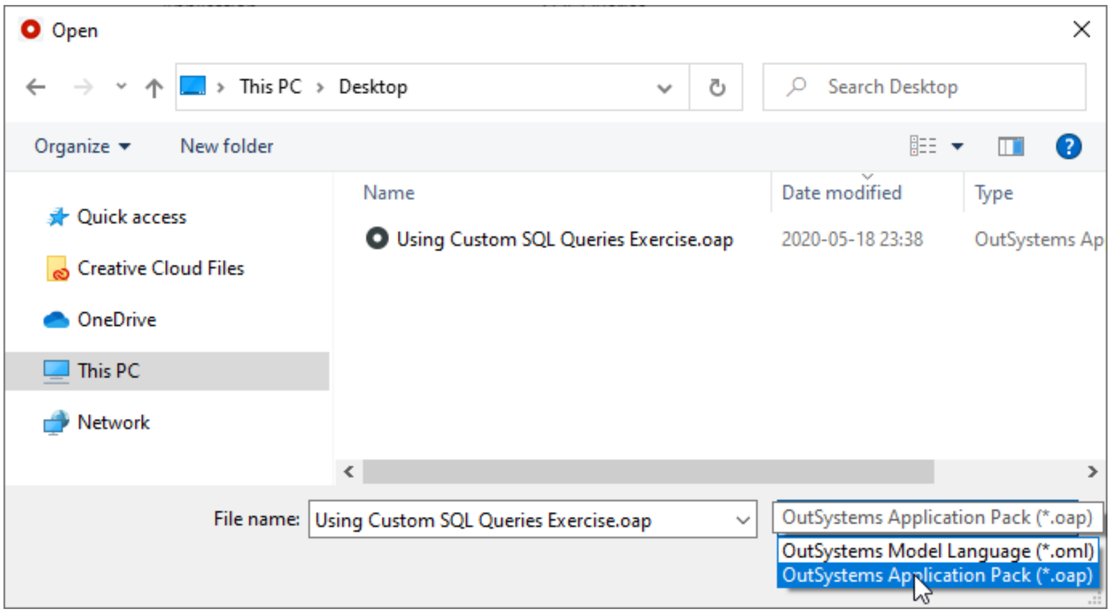
1) In Service Studio's main window, select the **Environment** menu on the top left.



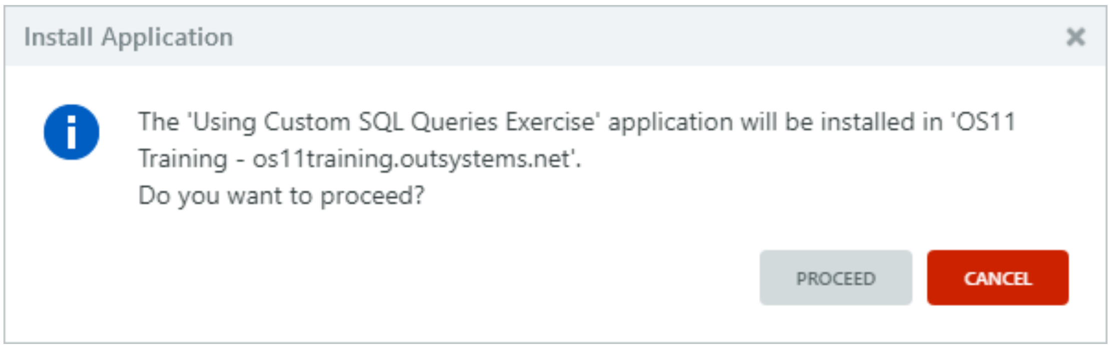
2) Select **Open Files...**.



3) In the following dialog, change the file type to OutSystems Application Pack (.oap), find the location of the Quickstart, and open the file named **Using Custom SQL Queries Exercise.oap**.

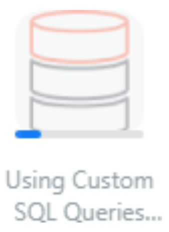

Open
This PC > Desktop
Search Desktop
Organize
New folder
Quick access
Creative Cloud Files
OneDrive
This PC
Network
Name
Date modified
Type
Using Custom SQL Queries Exercise.oap
2020-05-18 23:38
OutSystems Ap
File name:
Using Custom SQL Queries Exercise.oap
OutSystems Application Pack (*.oap)
OutSystems Model Language (*.oml)
OutSystems Application Pack (*.oap)

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

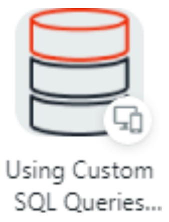

Install Application

The 'Using Custom SQL Queries Exercise' application will be installed in 'OS11 Training - os11training.outsystems.net'.
Do you want to proceed?

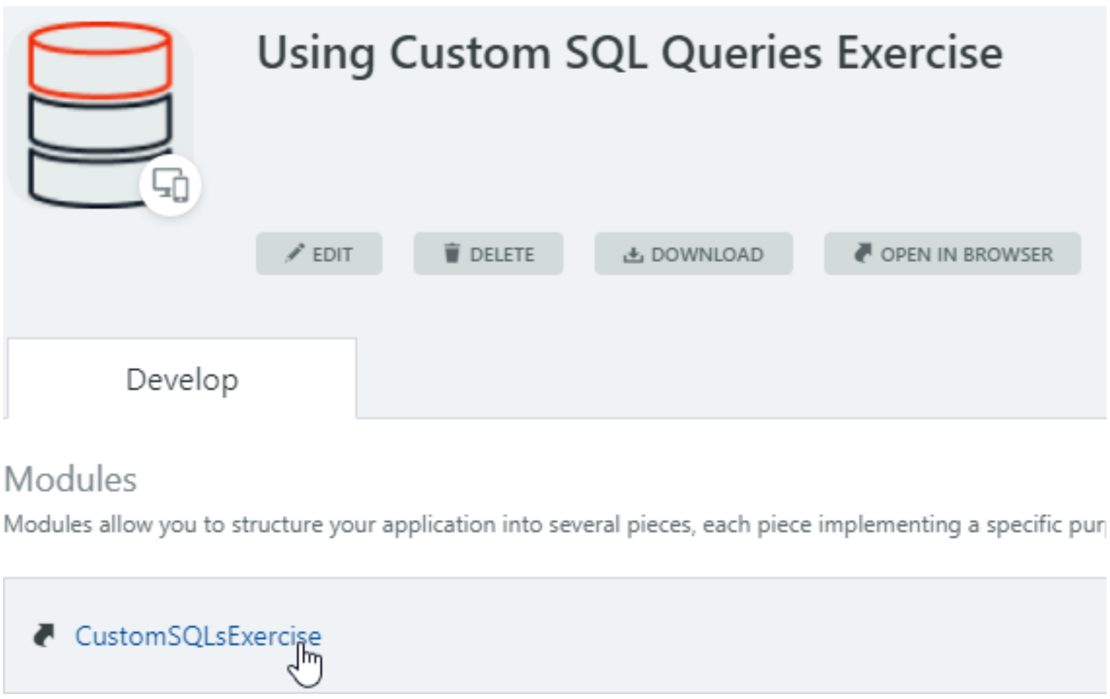
5) The application will begin installing automatically. When it's finished, we're ready to start!



6) Open the application in Service Studio.



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



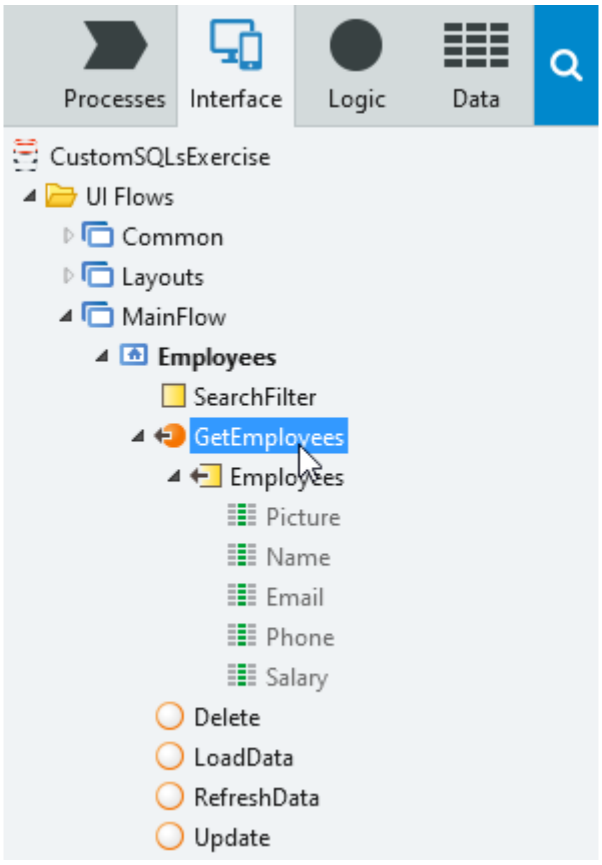
Modules allow you to structure your application into several pieces, each piece implementing a specific pur

Create a SELECT SQL Query

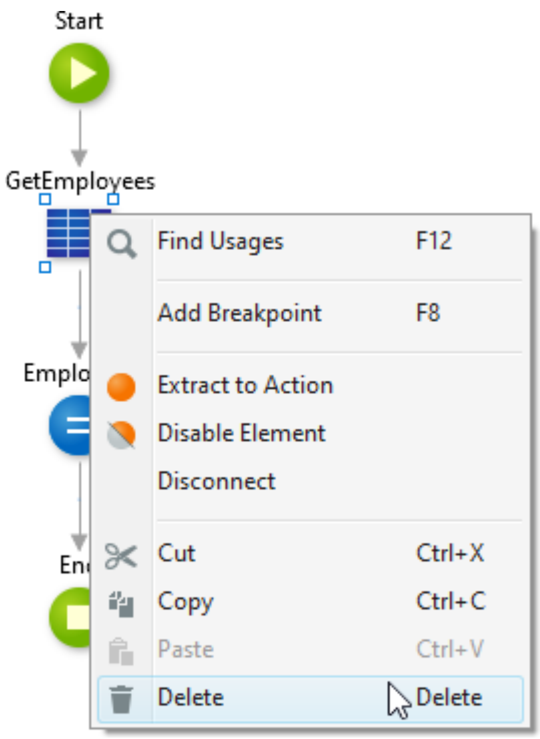
In this section, we will replace the existing Aggregate with an SQL Query that fetches the list of Customers from the Customer Entity. The Aggregate also contains a search filter (using the Local Variable `SearchFilter`) to filter the list of customers. The

Aggregate is located inside the Screen Data Action named GetEmployees, from the Employees screen.

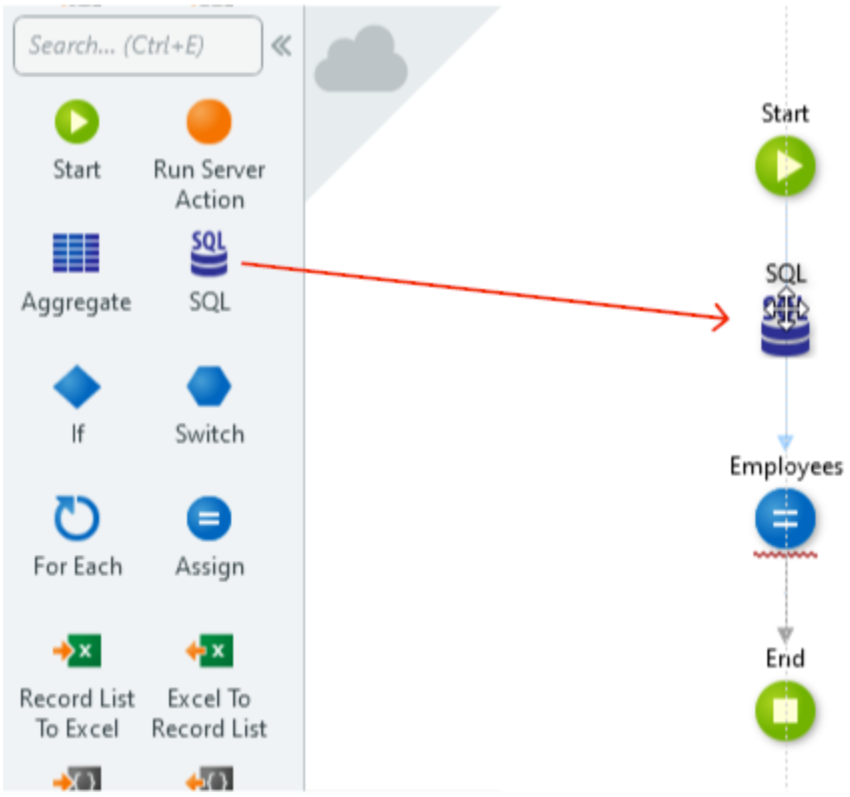
1) Open the **GetEmployees** Data Action.



2) Delete the existing **GetEmployees** Aggregate.

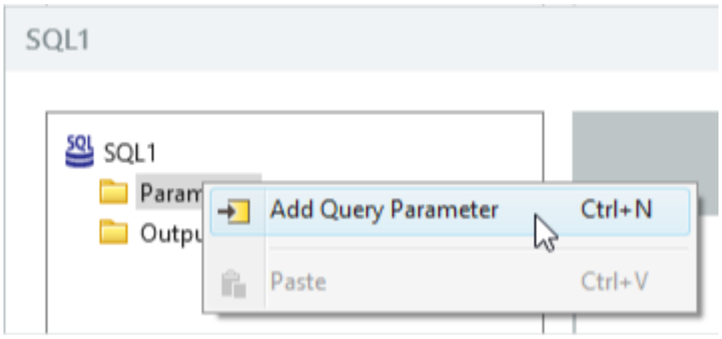


3) Drag an **SQL** and drop it where the Aggregate was.

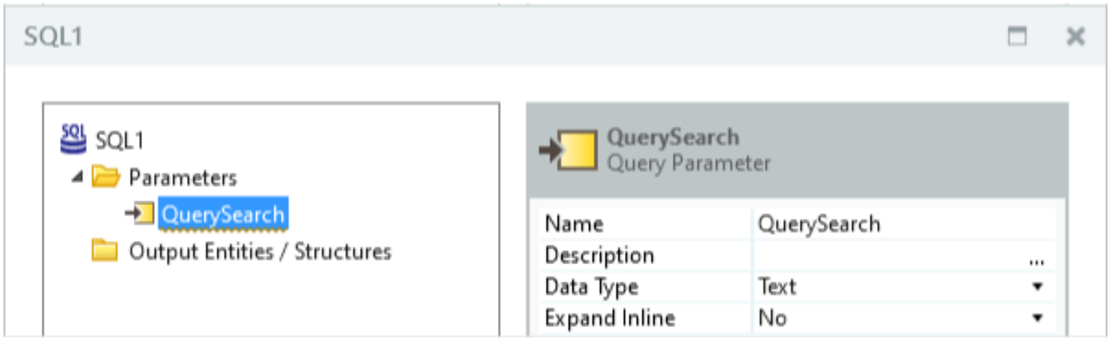


4) Double-click the newly created SQL to open the SQL Editor.

5) Right-click the Parameters folder, and select **Add Query Parameter**.



6) Set the **Name** of the Query Parameter to *QuerySearch*, and make sure its **Data Type** is set to *Text*.

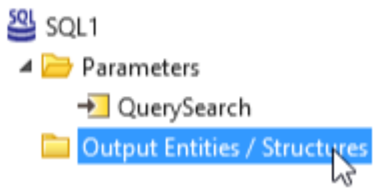


Name	QuerySearch
Description	
Data Type	Text
Expand Inline	No

7) In the SQL tab below, write the following SQL Query

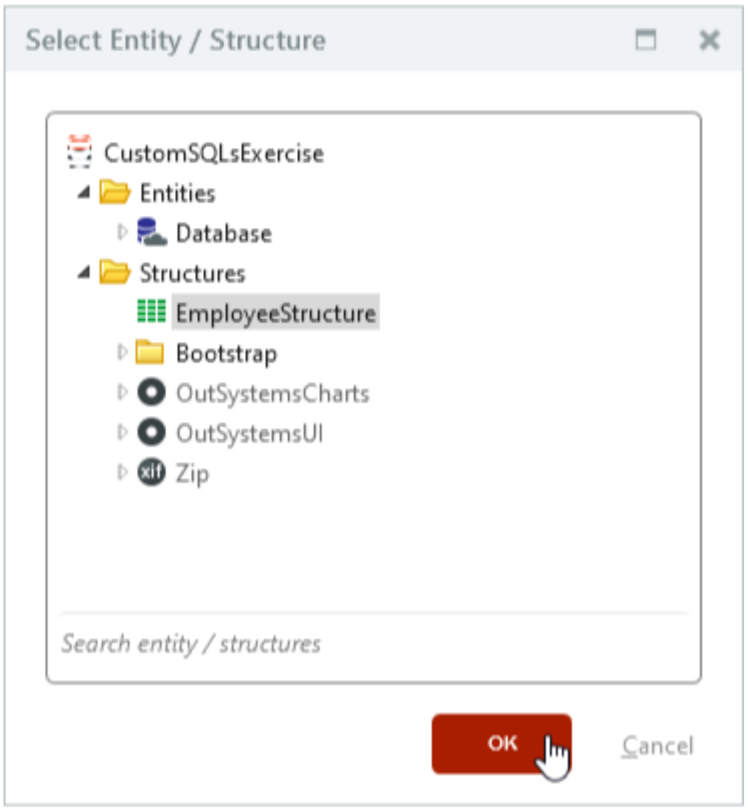
```
SELECT
{EmployeePicture}.[Picture],
{Employee}.[FirstName] + ' ' + {Employee}.[LastName],
{Employee}.[Email],
{Employee}.[Phone],
{Employee}.[Salary]
FROM
{Employee} LEFT JOIN {EmployeePicture} ON {Employee}.[Id] =
{EmployeePicture}.[Id]
WHERE
{Employee}.[FirstName] LIKE '%' + @QuerySearch + '%'
OR
{Employee}.[LastName] LIKE '%' + @QuerySearch + '%'
```

8) Double-click the **Output Entities/Structures** to select the output structure of the query.



You can also right-click the folder and pick from the suggestions, or select the *(Add Another Entity / Structure)* option.

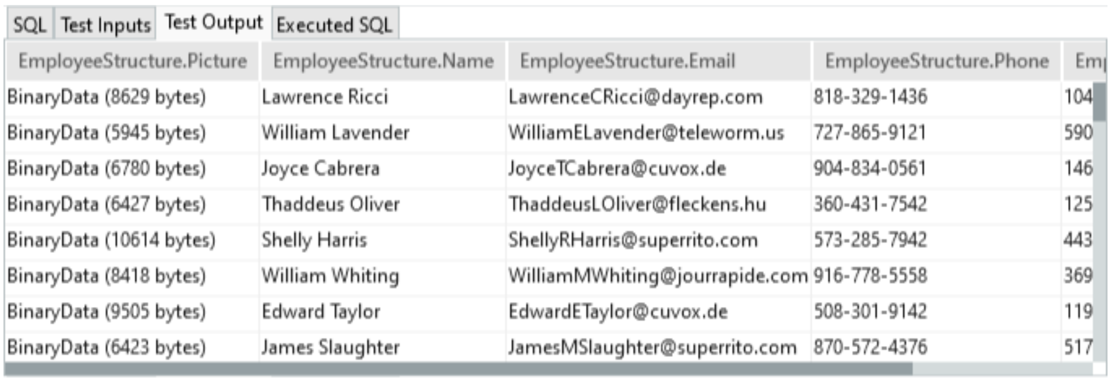
9) In the *Select Entity/Structure* dialog, select the **EmployeeStructure** then click **OK**.



The *EmployeeStructure* has been already provided in the quick start of the application. If you are creating a different output, you may need to create your own Structure. It is recommended to create a custom Structure with only the necessary attributes to be fetched from the Database. This ensures that only the required data is fetched from the database.

You can check the definition of the *EmployeeStructure* in the Data tab, under the Structures folder.

10) Click **Test** to test the query.



SQL | Test Inputs | Test Output | Executed SQL

EmployeeStructure.Picture	EmployeeStructure.Name	EmployeeStructure.Email	EmployeeStructure.Phone	Em
BinaryData (8629 bytes)	Lawrence Ricci	LawrenceCRicci@dayrep.com	818-329-1436	104
BinaryData (5945 bytes)	William Lavender	WilliamELavender@teleworm.us	727-865-9121	590
BinaryData (6780 bytes)	Joyce Cabrera	JoyceTCabrera@cuvox.de	904-834-0561	146
BinaryData (6427 bytes)	Thaddeus Oliver	ThaddeusLOliver@fleckens.hu	360-431-7542	125
BinaryData (10614 bytes)	Shelly Harris	ShellyRHarris@superrito.com	573-285-7942	443
BinaryData (8418 bytes)	William Whiting	WilliamMWhiting@jourrapide.com	916-778-5558	369
BinaryData (9505 bytes)	Edward Taylor	EdwardETaylor@cuvox.de	508-301-9142	119
BinaryData (6423 bytes)	James Slaughter	JamesMSlaughter@superrito.com	870-572-4376	517

Notice that the order of the columns matches the order of the structure attributes.

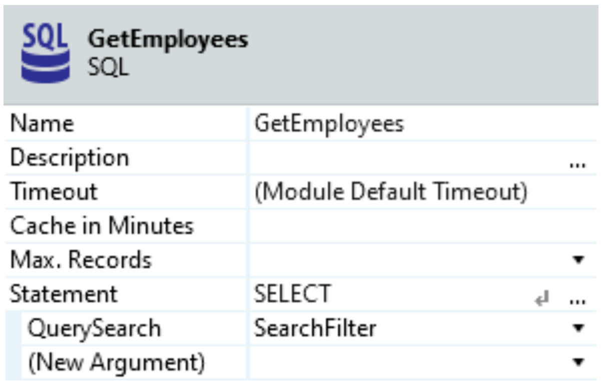
11) In the **Test Inputs** tab, set some value for the *QuerySearch* parameter (e.g. "Ro") then click the **Test** button again to make sure the filter is working. The number of records returned will be smaller, depending on your test value.

12) Click **Done** to close the SQL Editor and return to the flow of the *GetEmployees* Data Action.

13) Select the **SQL1**, then in the properties pane change the **Name** to *GetEmployees*.

By doing this, and setting it to the exact name as the Aggregate we initially had, Service Studio will automatically heal the code (namely the Assign after the created SQL). This works because Aggregates and SQL queries have the same output (apart from what is defined as the Output Entities or Structures).

14) Set the **QuerySearch** to the *SearchFilter* local variable.



GetEmployees
SQL

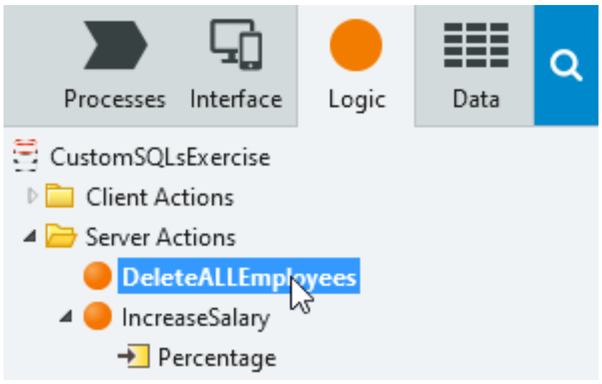
Name	GetEmployees
Description	...
Timeout	(Module Default Timeout)
Cache in Minutes	
Max. Records	
Statement	SELECT ...
QuerySearch	SearchFilter
(New Argument)	

15) Publish the Module using the **1-Click Publish button**, then test it in the browser. Make sure that the list of employees is displayed and the search filter is working properly.

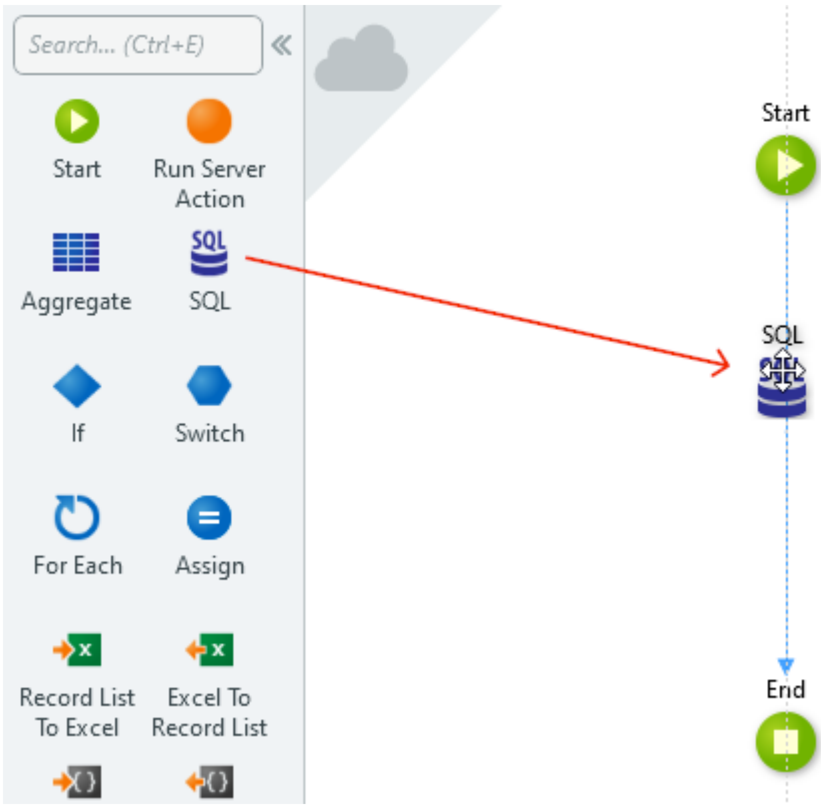
Create a DELETE SQL Query

In this section, we will implement a DELETE SQL Query. This query will be executed when the Delete button (on the Employees screen) is pressed, and clears all Employees. Notice that this is an exercise. Adding such queries to an application should be done carefully, and may actually be irreversible.

1) In the Logic tab, open the **DeleteALLEmployees** Server Action.

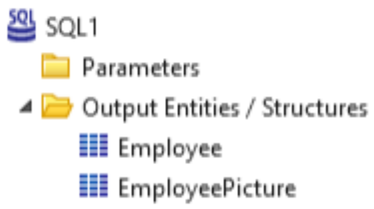


2) Drag an **SQL** and drop it between the Start and End.



3) Double-click the **SQL1** to open the SQL Editor.

4) Add the *Employee* and *EmployeePicture* entities as output.



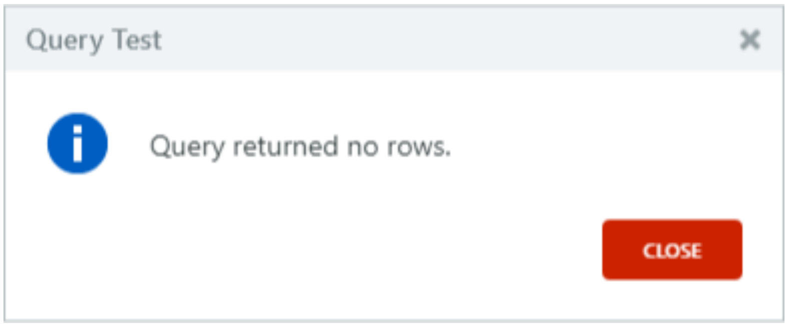
Although we are creating a DELETE SQL Query, we still have to define at least one Output Entity or Structure. In this case, since we will be deleting information from both Employee and EmployeePicture Entiites, we'll add both. Any Entity or Structure could be used, but using the actual Entities in use in the SQL query improves the code readibility.

5) In the SQL tab, write the following SQL Query

```
DELETE FROM {EmployeePicture};
DELETE FROM {Employee}
```

Note that we are actually executing two DELETE queries at once. Since the *EmployeePicture* is an extension of the *Employee* Entity, the order of the DELETEs needs to be defined like that. Otherwise, we wouldn't be able to delete Employees with Picture.

6) Click **Test** to test the query.



Query returned no rows.

Since this is a DELETE Query, no rows will be returned. Also, the Test won't actually delete any data from the database. However, during runtime **the data will be deleted**.

7) Click **Close** then **Done** to close the SQL Editor.

8) **Publish** the module, and then open it in the browser.

9) In the browser, click the **Delete ALL** button at the bottom of the screen. The Table should now be empty.

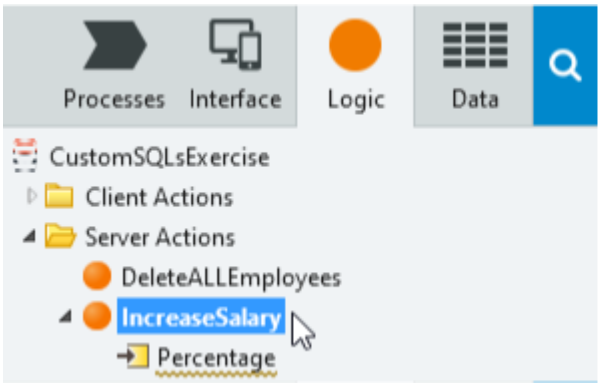
To reload the data, the Quickstart application comes with a **Bootstrap Data** button that will load all data to the database again.

10) Click the **Bootstrap Data** button.

Create an UPDATE SQL Query

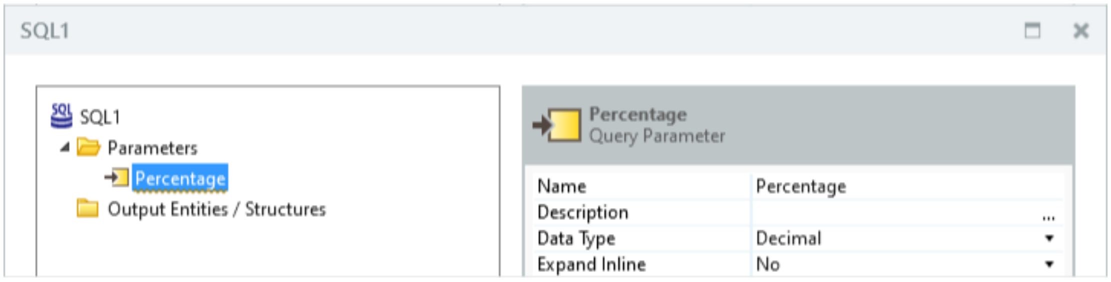
In this section, we will create an UPDATE SQL Query that will enable us to increase the salary of all employees by a percentage.

1) In the Logic tab, locate and open the **IncreaseSalary** Server Action.



2) Drag an **SQL** and drop it between the Start and End, then double-click it to open the SQL Editor.

3) Add a Query Parameter and set its **Name** to *Percentage*, and make sure its **Data Type** is set to *Decimal*.



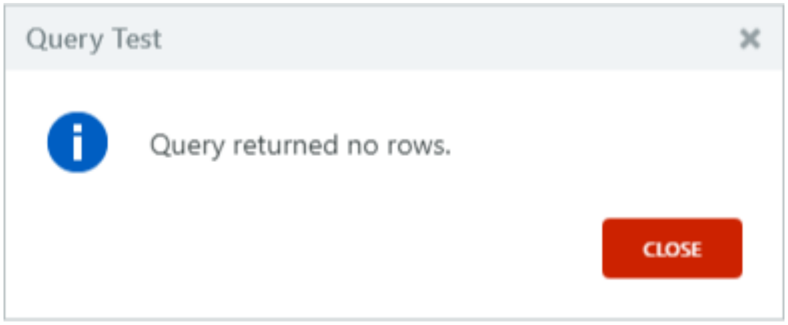
Name	Percentage
Description	
Data Type	Decimal
Expand Inline	No

4) Add the *Employee* entity as an output entity.

5) In the SQL tab write the following SQL Query

```
UPDATE
{Employee}
SET
{Employee}.[Salary] = {Employee}.[Salary] + {Employee}.[Salary] *
@Percentage
```

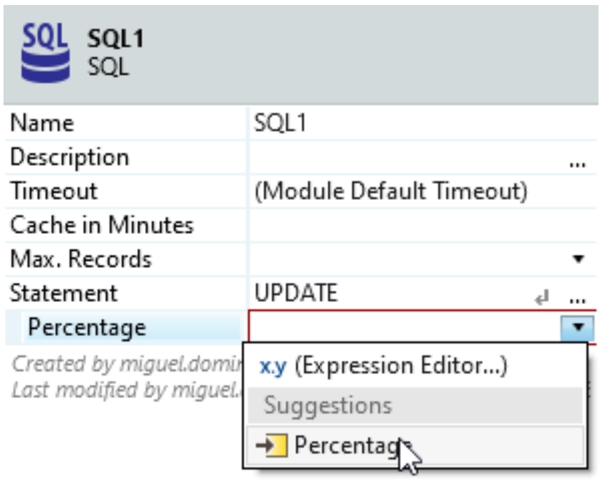
6) In the Test Inputs tab, set the Percentage to *0.03* then click the **Test** button.


Query Test

Query returned no rows.

7) Close the SQL Editor.

8) In the properties of the **SQL1**, set the **Percentage** query parameter to the *Percentage* input parameter.


SQL1
SQL

Name	SQL1
Description	
Timeout	(Module Default Timeout)
Cache in Minutes	
Max. Records	
Statement	UPDATE
Percentage	

Created by miguel.domi
Last modified by miguel.

9) **Publish** the module, then open it in the browser.

10) Click the **Update Salary (5%)** button located at the bottom. The salary of all employees should change immediately and reflect the increase.