



SQL Queries

Advanced Queries

SQL Tool

- Overview
- Input / Outputs
- Testing Queries
- Non-Select Queries

Advanced Queries

- Retrieving the correct data from the database can often be complex

The screenshot displays the 'GetProductionPeople' query builder interface. At the top, there are tabs for 'Sources', 'Filters', 'Sorting', and 'Test Values'. The 'Sources' tab is active, showing a list of data sources: 'Person', 'PersonMovieRole', and 'PersonRole'. Below this, there is an 'Add Source' button. The 'JOINS' section shows two joins defined:

- Join 1: 'Person' (Source 1) joined with 'PersonMovieRole' (Source 2) using the condition 'Person.Id = PersonMovieRole.PersonId'.
- Join 2: 'PersonMovieRole' (Source 1) joined with 'PersonRole' (Source 2) using the condition 'PersonMovieRole.PersonRoleId = PersonRole.Id'.

Below the joins, there is an 'Add Join' button. At the bottom, a table displays the results of the query. The table has six columns: 'Name', 'Surname', 'DateOfBirth', 'DateOfDeath', 'Label', and 'NameAndRole'. The first row of data shows 'Ben', 'Stiller', '1965-11-30', '1900-01-01', 'Director', and 'Ben Stiller (Director)'.

Person	Person	Person	Person	PersonRole	Name + " " + Sur...
Name	Surname	DateOfBirth	DateOfDeath	Label	NameAndRole
Ben	Stiller	1965-11-30	1900-01-01	Director	Ben Stiller (Director)

Advanced Queries

- Retrieving the correct data from the database can often be complex

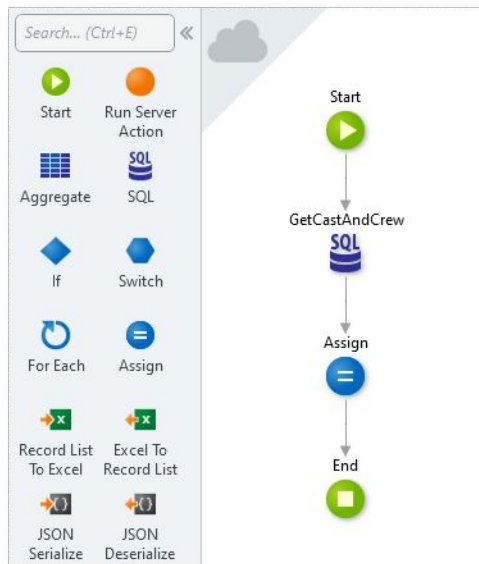
The screenshot shows the 'GetProductionPeople' query builder interface. It has tabs for 'Sources', 'Filters', 'Sorting', and 'Test Values'. The 'Sources' tab is active, showing a list of sources: Person, PersonMovieRole, and PersonRole. Below this is an 'Add Source' button. The 'JOINS' section shows two joins: 1. Person (Source 1) joined with PersonMovieRole (Source 2) using the condition 'Person.Id = PersonMovieRole.PersonId'. 2. PersonMovieRole (Source 1) joined with PersonRole (Source 2) using the condition 'PersonMovieRole.PersonRoleId = PersonRole.Id'. Below the joins is an 'Add Join' button. At the bottom, there is a table with 6 columns: Name, Surname, DateOfBirth, DateOfDeath, Label, and NameAndRole. The first row of data shows Ben Stiller with birth date 1965-11-30 and death date 1900-01-01, labeled as Director. The NameAndRole column shows 'Ben Stiller (Director)'. There is a 'New attribute' button to the right of the table.

Person Name	Person Surname	Person DateOfBirth	Person DateOfDeath	PersonRole Label	PersonRole Name + " " + Sur...
Ben	Stiller	1965-11-30	1900-01-01	Director	Ben Stiller (Director)

- It is also possible to write SQL statements
 - SQL Tool

SQL Tool

- Can be used in server-side flows
 - Server Actions
 - Screen Data Actions



The screenshot shows the configuration window for the 'GetCastAndCrew' SQL action. The left pane shows a tree structure with 'Parameters' (Movied, ActorRole, CrewRole) and 'Output Entities / Structures' (GetCastAndCrew). The right pane shows the SQL query editor with the following query:

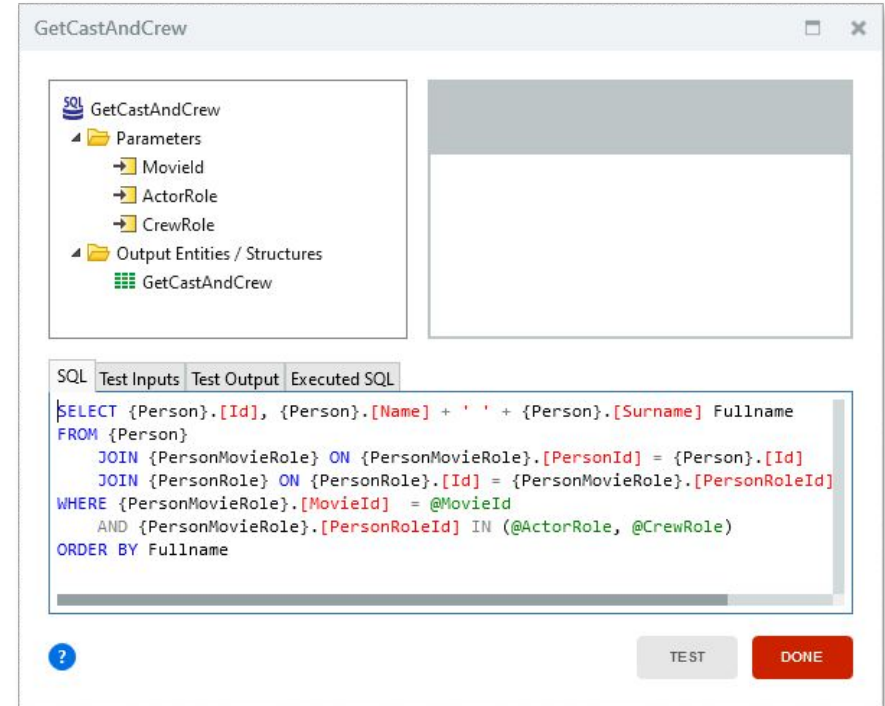
```
SELECT {Person}.[Id], {Person}.[Name] + ' ' + {Person}.[Surname] Fullname
FROM {Person}
JOIN {PersonMovieRole} ON {PersonMovieRole}.[PersonId] = {Person}.[Id]
JOIN {PersonRole} ON {PersonRole}.[Id] = {PersonMovieRole}.[PersonRoleId]
WHERE {PersonMovieRole}.[MovieId] = @MovieId
AND {PersonMovieRole}.[PersonRoleId] IN (@ActorRole, @CrewRole)
ORDER BY Fullname
```

At the bottom, there are tabs for 'SQL', 'Test Inputs', 'Test Output', and 'Executed SQL'. Below the query editor are buttons for '?', 'TEST', and 'DONE'.



SQL Tool

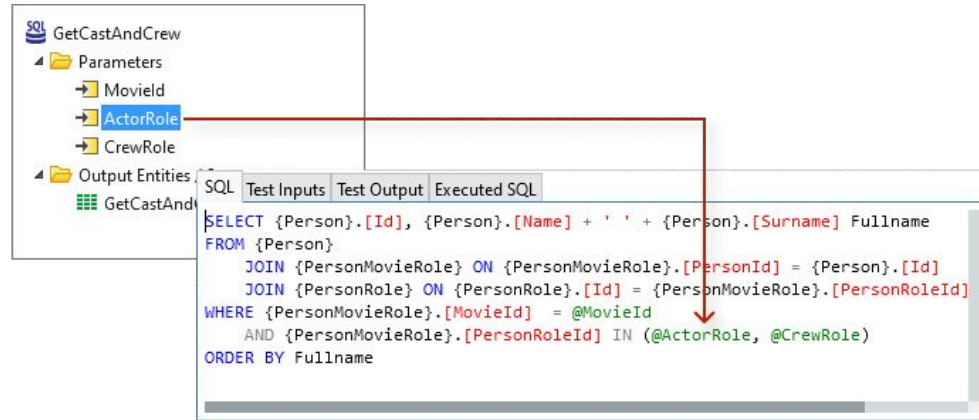
- Can be used in server-side flows
 - Server Actions
 - Screen Data Actions
- Input and outputs are specified visually
- Allows writing SQL statements
 - SQL syntax checking





Inputs

- SQL Tools are “black boxes”
 - Parameters pass information to it
 - No access to outside scope





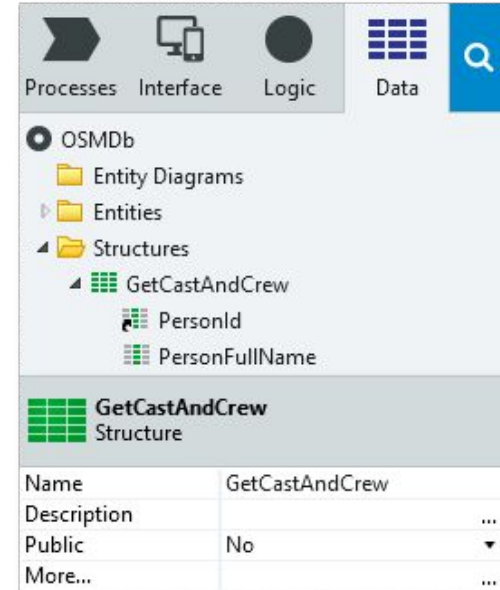
SQL Query

- SQL is written manually and DBMS-specific
 - TOP vs LIMIT
- Use of abstract names for Entities and Attributes
 - Write Entity names and attribute names as {Entity}.[Attribute]

```
SQL Test Inputs Test Output Executed SQL
SELECT {Person}.[Id], {Person}.[Name] + ' ' + {Person}.[Surname] Fullname
FROM {Person}
    JOIN {PersonMovieRole} ON {PersonMovieRole}.[PersonId] = {Person}.[Id]
    JOIN {PersonRole} ON {PersonRole}.[Id] = {PersonMovieRole}.[PersonRoleId]
WHERE {PersonMovieRole}.[MovieId] = @MovieId
    AND {PersonMovieRole}.[PersonRoleId] IN (@ActorRole, @CrewRole)
ORDER BY Fullname
```

Structures

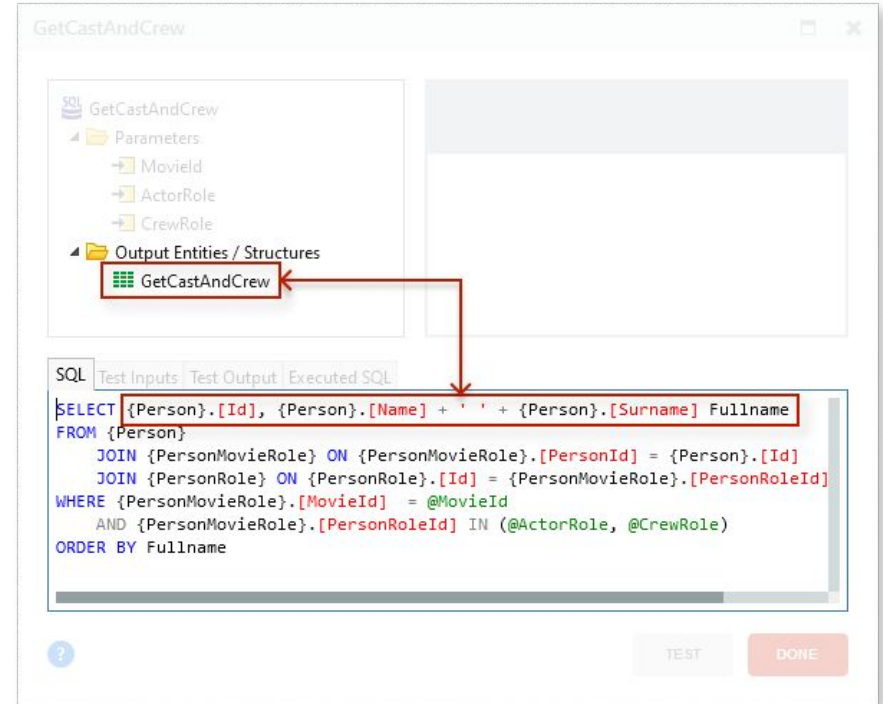
- Structures are custom compound data types
- Used to store compound data in memory
 - Definition of a data type
 - The Structure itself does not hold any value
 - Structures are not variables
- Structures are defined by attributes of any data type
 - Including other Structures, Entities and Lists





Outputs

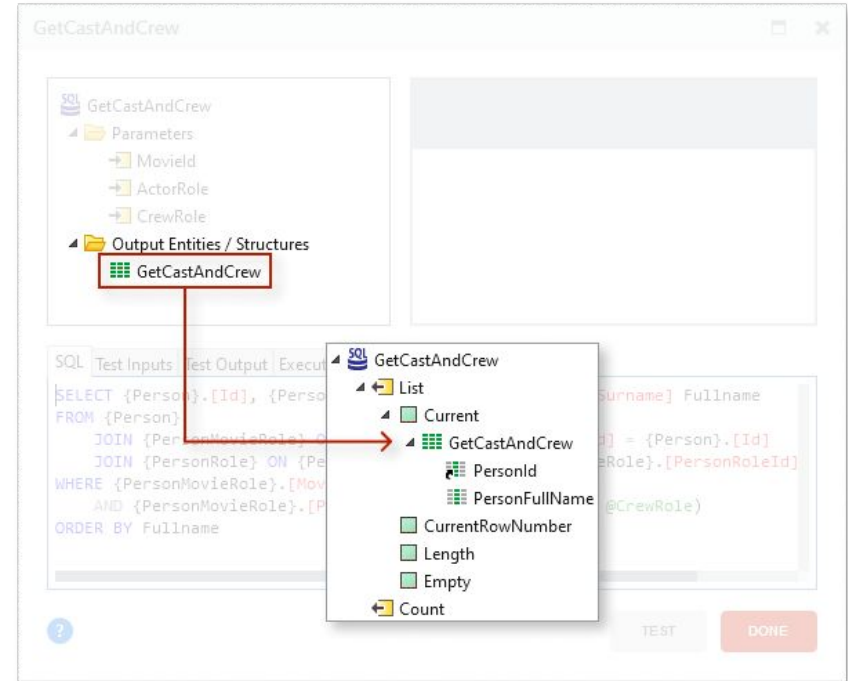
- Outputs Entities / Structures
 - Mandatory to have at least one
 - Can use Entities or Structures
 - Need to map the columns in the SELECT clause





Outputs

- Outputs Entities / Structures
 - Mandatory to have at least one
 - Can use Entities or Structures
 - Need to map the columns in the SELECT clause
- The output of the SQL Tool is the same as in Aggregates
 - List
 - Count



SQL Testing Queries

Setting Input Parameters

SQL	Test Inputs	Test Output	Executed SQL
	Parameter	Test Value	
	Moviefld	8	
	ActorRole	Entities.PersonRole.Actor	
	CrewRole	Entities.PersonRole.Crew	

Test Max. Records:

? TEST DONE

Testing the SQL Tool query

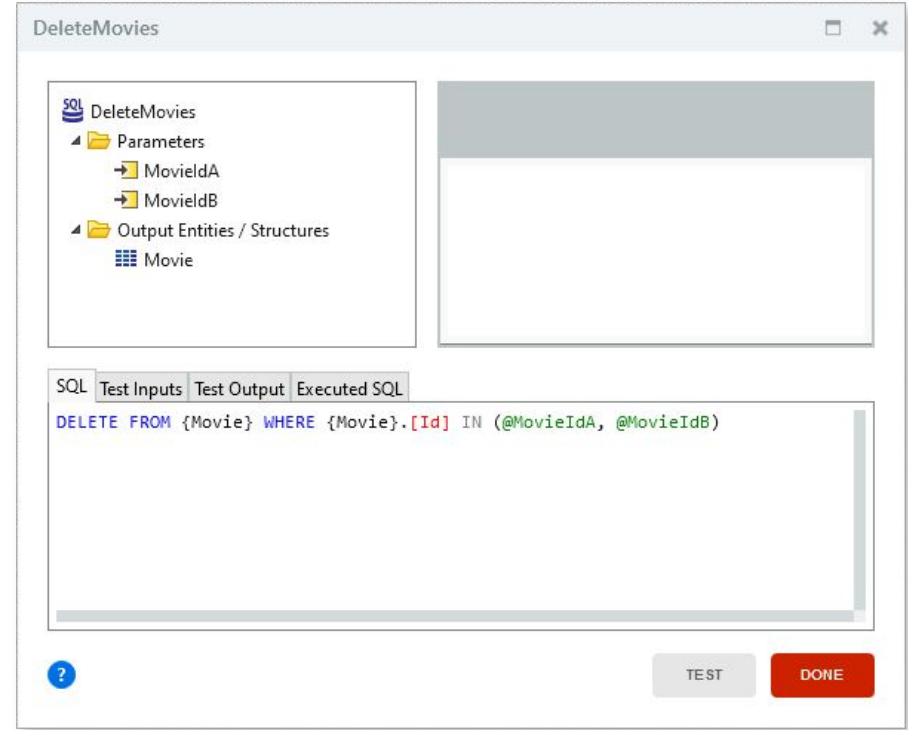
SQL	Test Inputs	Test Output	Executed SQL
	GetCastAndCrew.PersonId	GetCastAndCrew.PersonFullName	
5		Alan Rickman	
10		Robert Downey Jr.	
7		Sean Connery	

3 rows returned.

? TEST DONE

Non-SELECT Queries

- SQL Tools can also be used for non-SELECT statements
 - Delete
 - Insert
 - Update
- SQL statement can be written normally
- A “dummy” Output Structure still needs to be provided
 - Structure attributes are irrelevant



Questions?

Thank you
