Microsoft Dynamics NAV 2013
SQL Readiness Training

Lab Manual for Module 2: Query Objects

Version 1.0

Released: October 1, 2012

Conditions and Terms of Use

Microsoft Confidential

Copyright and Trademarks

About the Authors	
Author:	Gerard Conroy
Bio:	Gerard is an Escalation Engineer in the MICROSOFT DYNAMICS NAV support team based in the United Kingdom. He has been working with MICROSOFT DYNAMICS NAV since 2007 and his previous roles within Microsoft include the SQL Server support team and the internal IT Department.
Author:	Christine Avanessians
Bio:	Christine is a MICROSOFT DYNAMICS NAV Program Manager working on the Server and Tools team based in Denmark who has provided the content and information for this Module.

Table of Contents

LAB 1: WORKING WITH QUERY OBJECTS 1
  EXERCISE 1: CREATE AND EXECUTE A TYPICAL QUERY OBJECT 2
  EXERCISE 2: FILTERING WITH A QUERY 8

Lab 1: Working with Query Objects

During this lab, you will create a simple Query Object

Estimated time to complete this lab: **30 minutes**

Before You Begin

The following are prerequisites before commencing this lab:

- Complete Module 2: Query Objects
- A Microsoft Dynamics NAV 2013 Demo environment (preferably W1)
  - Install Microsoft Dynamics NAV 2013 on your local machine

    .. OR ..
  - Create a VMAS machine using the Microsoft Dynamics NAV 2013 Demo Environment template. The steps to do this are documented in Lab 1 - Setup Considerations and Performance Tuning.

What You Will Learn

After completing this lab, you will be able to:

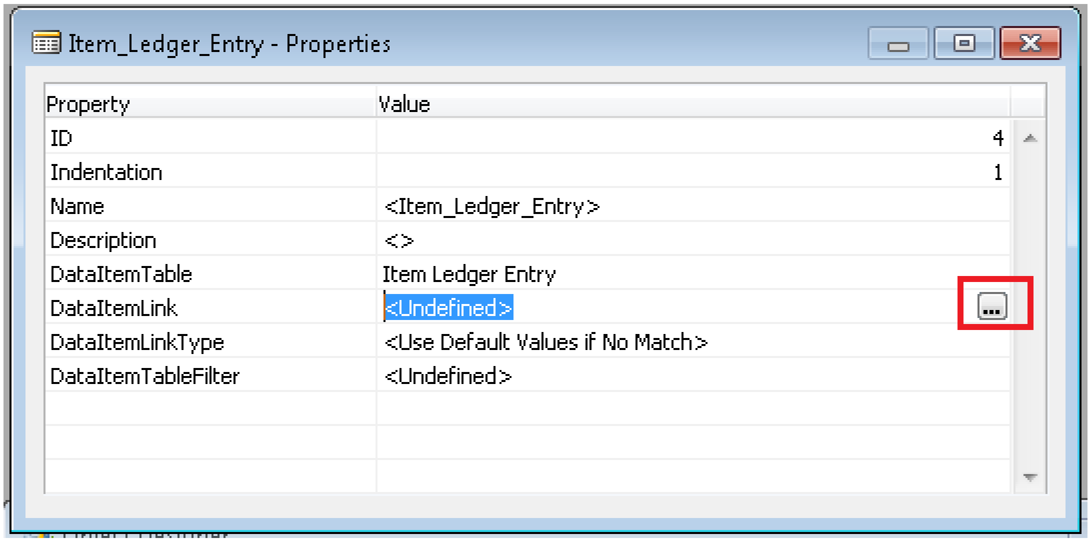
- Create a typical Query Object in Microsoft Dynamics NAV 2013
- Filter with a Query Object.

Scenario

You are developing a Microsoft Dynamics NAV Report which has a requirement to retrieve a significant amount of data from the Microsoft Dynamics NAV database. Processing time must be kept to a minimum. You want to extract information from both the Item and Item Ledger Entry tables in a single query with the data being Joined across the two tables and filtered on SQL Server rather than on the client machine. You decide that a Query Object can provide the functionality you require.

Exercise 1: Create and execute a typical Query Object

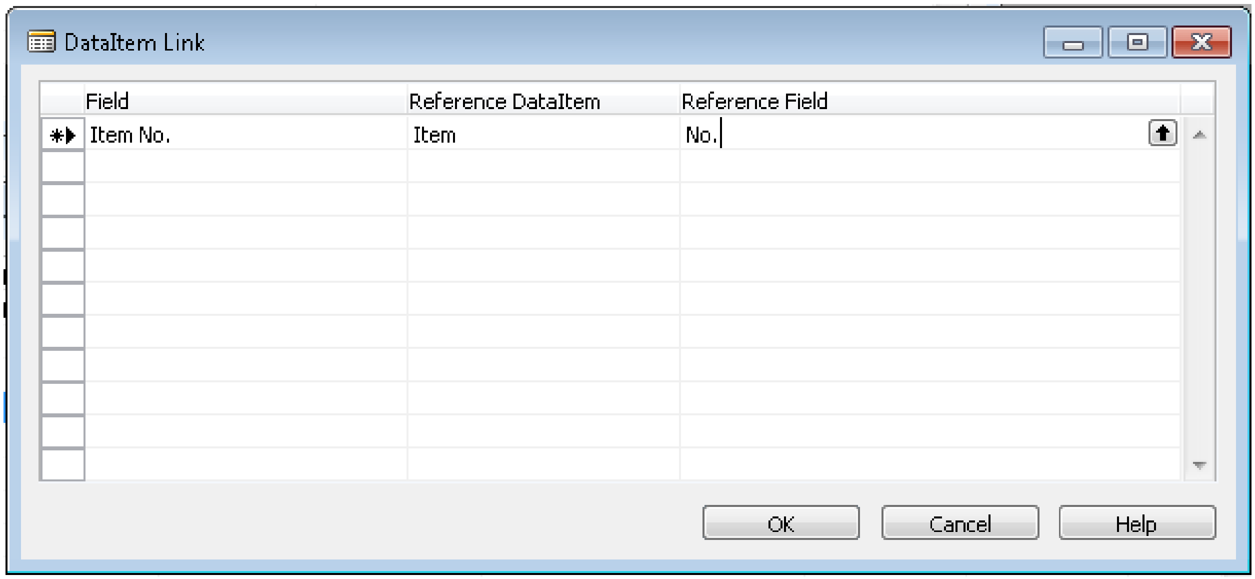
1. Start the Microsoft Dynamics NAV 2013 Developer Environment (referred to as “NAV DE” throughout the rest of this document). The icon has been placed on the Windows Taskbar for convenience. This will automatically open a connection to the default Microsoft Dynamics NAV database.
2. In the NAV DE, select **Tools** and then select **Object Designer**
3. From the list of Object Types on the left, select **Query** and then click **New** to bring up the Query Designer. You can use the Query Designer to specify exactly what Microsoft Dynamics NAV Data Items (tables) you want to retrieve data from. You can define what the relationship between these tables are (i.e. how they should be Joined) and you can also define what SQL Server query artifacts you want to apply, e.g. Grouping, Ordering, etc.
4. On the first line of this Query Object enter the following details:
    a. Type=DataItem
    b. Data Source=Item (you can use the lookup to select the Item table if you prefer)
5. On the 2nd line enter the following:
    a. Type= Column
    b. Data Source=No. (you can use the lookup to select the **No.** column if you prefer)
6. On the 3rd line enter:
    a. Type= Column
    b. Data Source=Description
7. On the 4th line of this Query Object enter the following details:
    a. Type=DataItem
    b. Data Source=Item Ledger Entry
8. You need to specify how the **Item Ledger Entry** DataItem relates to the **Item** DataItem. To do this highlight the **Item Ledger Entry** DataItem you just created, select **View** and then select **Properties** from the file menu.
9. Note that the **DataItemTable** property is already filled in for you. You need to specify the **DataItemLink** property manually however. Click on the ellipse button for an easy user interface that will allow you to



Property	Value
ID	4
Indentation	1
Name	<Item_Ledger_Entry>
Description	<>
DataItemTable	Item Ledger Entry
DataItemLink	<Undefined>
DataItemLinkType	<Use Default Values if No Match>
DataItemTableFilter	<Undefined>

Figure 1

10. Select the criteria as shown in the image below so that SQL Server will join the two tables based on the following columns: [Item Ledger Entry].[Item No.] = [Item][No.]:



Field	Reference DataItem	Reference Field
Item No.	Item	No.

Figure 2

11. Click **OK** and note that the DataItemLink property is filled out as follows:
    - Item No.=Item."No."

    The above could have been typed into this field as a shortcut to avoid the two steps above.

12. Note the DataItemLinkType property is set to the following value by default:

<Use Default Values if No Match>

This means that if there are records in the Item table which do have matching records in the Item Ledger Entry table then those Item records will still be included in the Query results. These result set records will show "Default Values" for any columns that are associated with the Item Ledger Entry table.

If you would prefer that Item records which have no matching Item Ledger Entry records were excluded from the Record set, you could modify the DataItemLinkType property by selecting **Exclude Row If No Match**

In other words, the default type of JOIN which is applied in SQL Server is a LEFT OUTER JOIN but you have the option to change this to an INNER JOIN using the DataItemLinkType property. In fact five different types of SQL Server JOINS can be configured for the Query object.

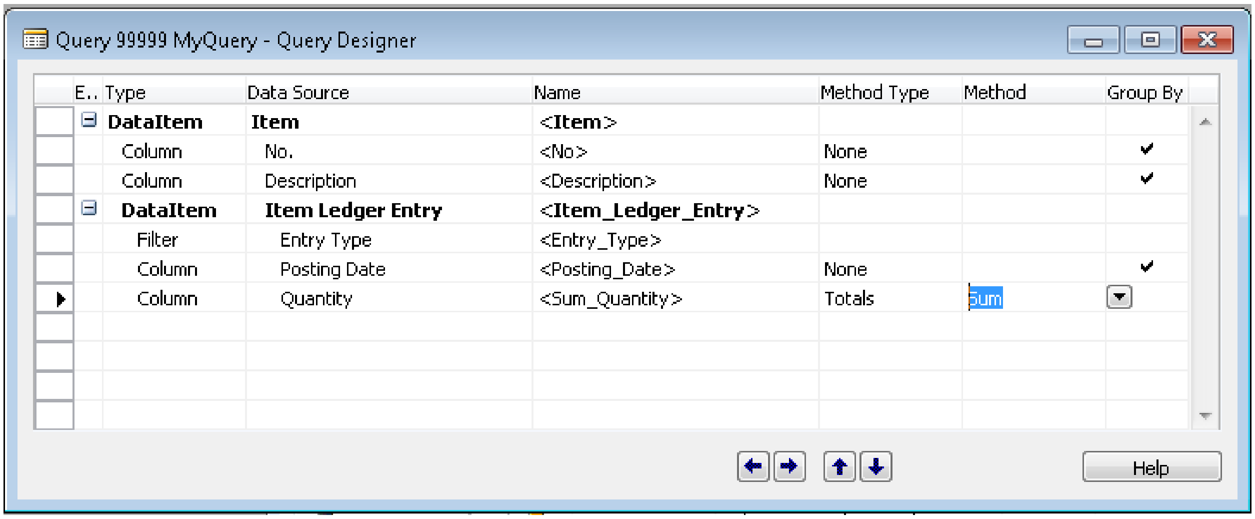
13. We also want to add a filter to our query so that it will only include records where the Entry Type is **Sale**. There are several ways we could achieve this:
    a. We could add a Filter column to our query like the one added in the example in the stud notes for this chapter in the training guide.
    b. The "ColumnFilter" property on the relevant column could be set to do the required filtering.
    c. The "DataItemTableFilter" property on the Data Item can be used to specify the required filter.

    In our case, we are going to use option 'c'. You could use the ellipse button to select the column and filter type you want to use. Alternatively you can enter the filter directly into the property. Enter the following filter value into DataItemTableFilter property: Entry Type=CONST(Sale)
14. Close the Property window to save the above changes.
15. On the 5th line of this Query Object enter the following details:
    a. Type=Column
    b. Data Source=Posting Date
16. On the 6th line of this Query Object enter the following details:
    a. Type=Column
    b. Data Source=Quantity
    c. MethodType=Totals
    d. Method=Sum

    Note that by specifying MethodType=Totals then the "Group By" column becomes checked for all of the other column fields in the Query object. This is because the Quantity field has now been configured as an Aggregate and aggregation in a SQL

Server query is achieved by using a “Group By” clause which must include all non-aggregated columns.

17. Your Query object should now look like this:



E..	Type	Data Source	Name	Method Type	Method	Group By
⊟	**DataItem**	**Item**	**<Item>**			
	Column	No.	<No>	None		✓
	Column	Description	<Description>	None		✓
⊟	**DataItem**	**Item Ledger Entry**	**<Item_Ledger_Entry>**			
	Filter	Entry Type	<Entry_Type>			
	Column	Posting Date	<Posting_Date>	None		✓
	Column	Quantity	<Sum_Quantity>	Totals	Sum	

Figure 3

18. Select the **File** menu option and then click **Save As** to save the Query Object. Specify the following:
    a. ID=50001
    b. Name=MyQuery

    Click **OK** to save the object.

19. Congratulations! You have just created a Query object in MICROSOFT DYNAMICS NAV 2013.

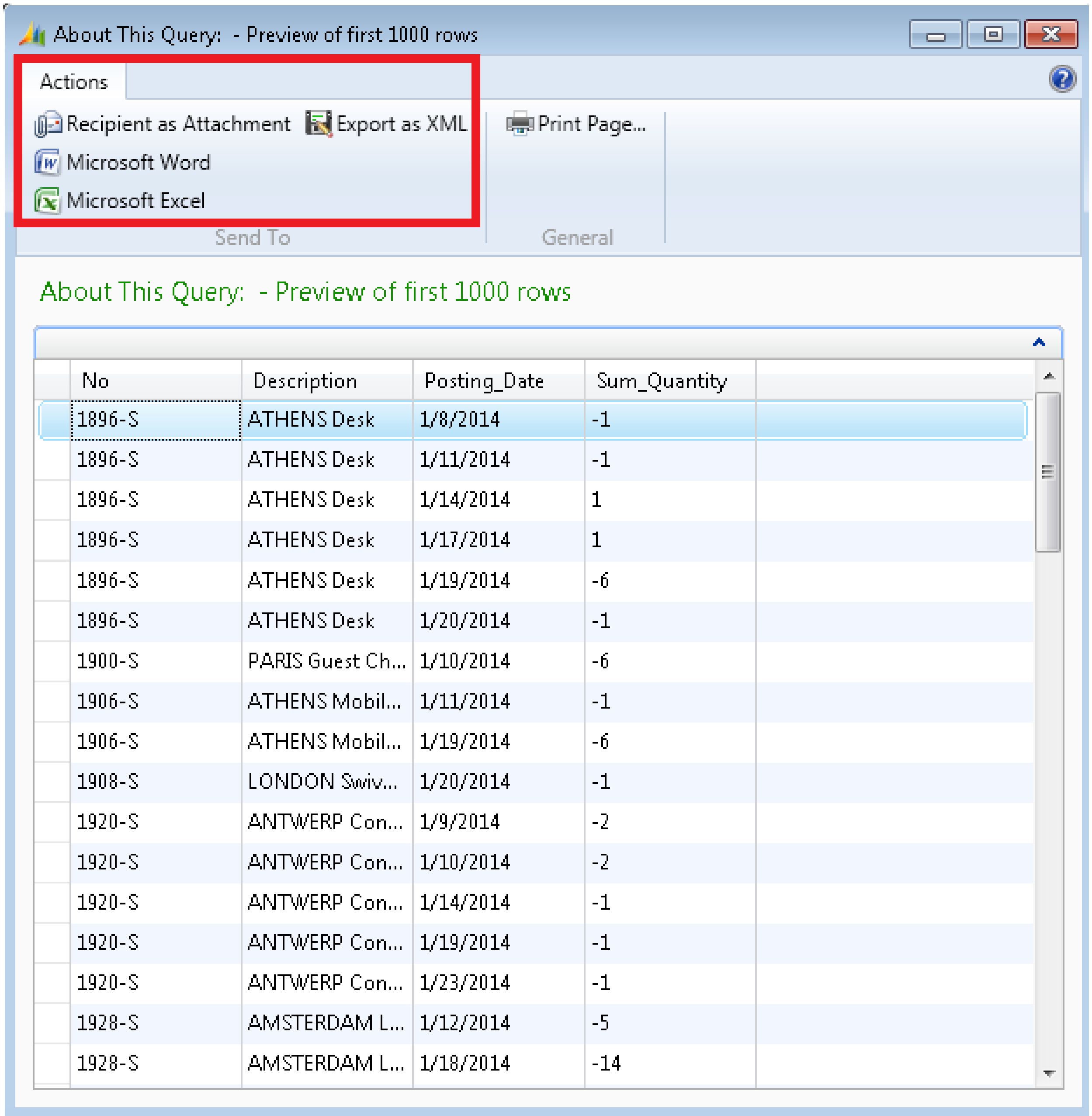
20. Now click **Run** in the Query Designer to execute your Query while your Query is highlighted. Alternatively you can run the following command from a Windows command prompt (e.g: **Start** > **Run**):

    dynamicsnav:////runquery?query=19

21. Note how fast the Query executes because the data is retrieved from SQL Server using a highly optimized method (e.g. set based queries).

22. Note that the results are produced in a Query Preview Page which allows you to export the results from the RTC into a variety of formats:
    a. As an attachment to an email
    b. Microsoft Word
    c. Microsoft Excel
    d. As an XML Document.

Note that the Query results Preview Page is not a normal RTC page. It will only show a preview of a maximum of 1000 rows and is intended as a development tool to validate the Query definition. If they query returns more than 1,000 rows, then some records will not be shown in the Preview page. For this reason, the Preview Page should not be used in production scenarios. See Figure 4 for an example of a Query results Preview Page.

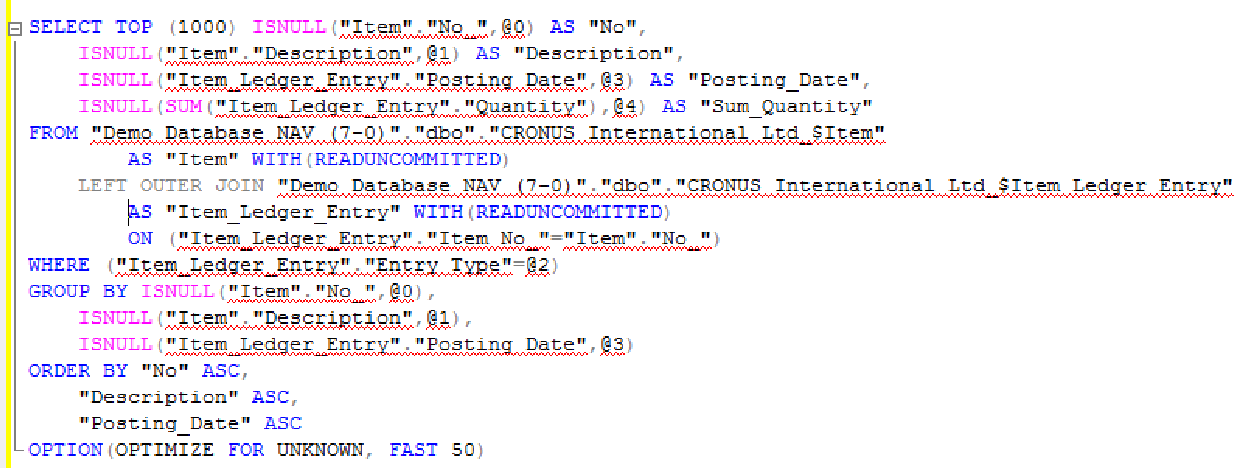


About This Query: - Preview of first 1000 rows

No	Description	Posting_Date	Sum_Quantity
1896-S	ATHENS Desk	1/8/2014	-1
1896-S	ATHENS Desk	1/11/2014	-1
1896-S	ATHENS Desk	1/14/2014	1
1896-S	ATHENS Desk	1/17/2014	1
1896-S	ATHENS Desk	1/19/2014	-6
1896-S	ATHENS Desk	1/20/2014	-1
1900-S	PARIS Guest Ch...	1/10/2014	-6
1906-S	ATHENS Mobil...	1/11/2014	-1
1906-S	ATHENS Mobil...	1/19/2014	-6
1908-S	LONDON Swiv...	1/20/2014	-1
1920-S	ANTWERP Con...	1/9/2014	-2
1920-S	ANTWERP Con...	1/10/2014	-2
1920-S	ANTWERP Con...	1/14/2014	-1
1920-S	ANTWERP Con...	1/19/2014	-1
1920-S	ANTWERP Con...	1/23/2014	-1
1928-S	AMSTERDAM L...	1/12/2014	-5
1928-S	AMSTERDAM L...	1/18/2014	-14

Figure 4

23. See below for the SQL Server T-SQL code which is executed to capture the results for our new Query object:



```
SELECT TOP (1000) ISNULL("Item"."No_",@0) AS "No",
    ISNULL("Item"."Description",@1) AS "Description",
    ISNULL("Item_Ledger_Entry"."Posting Date",@3) AS "Posting_Date",
    ISNULL(SUM("Item_Ledger_Entry"."Quantity"),@4) AS "Sum_Quantity"
FROM "Demo Database NAV (7-0)"."dbo"."CRONUS International Ltd_$Item"
        AS "Item" WITH(READUNCOMMITTED)
    LEFT OUTER JOIN "Demo Database NAV (7-0)"."dbo"."CRONUS International Ltd_$Item Ledger Entry"
        AS "Item_Ledger_Entry" WITH(READUNCOMMITTED)
        ON ("Item_Ledger_Entry"."Item No_"="Item"."No_")
WHERE ("Item_Ledger_Entry"."Entry Type"=@2)
GROUP BY ISNULL("Item"."No_",@0),
    ISNULL("Item"."Description",@1),
    ISNULL("Item_Ledger_Entry"."Posting Date",@3)
ORDER BY "No" ASC,
    "Description" ASC,
    "Posting_Date" ASC
OPTION(OPTIMIZE FOR UNKNOWN, FAST 50)
```

Figure 5

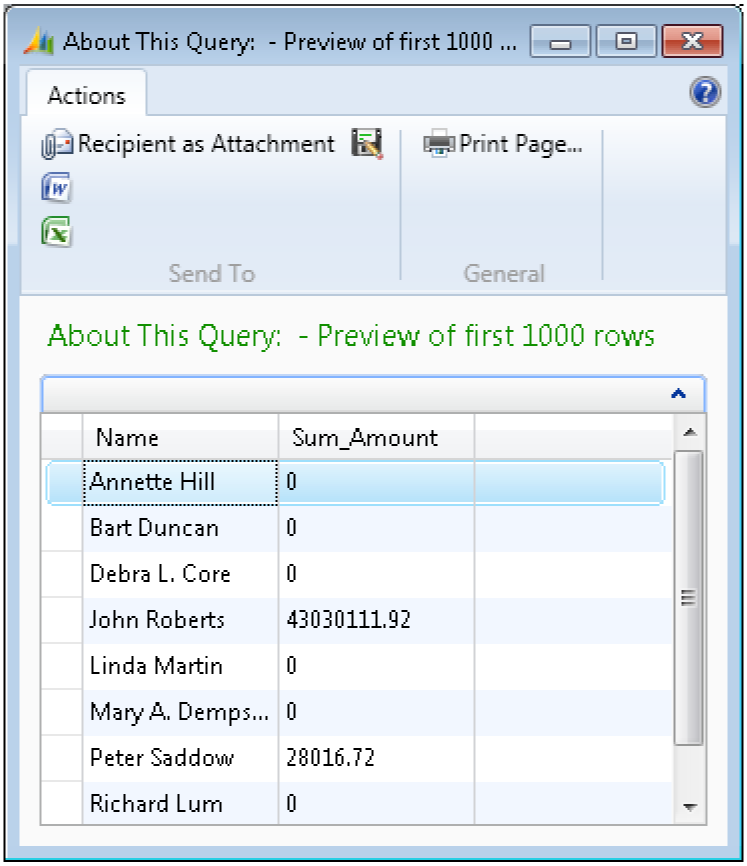
Exercise 2: Filtering with a Query

For the next exercise we will create a new Query object and see how filtering behaves with two DataItems.

1. Click **New** in the Query Explorer to create a new Query object.
2. In the first row of the Query Designer window enter the following:
   a. Type = DataItem
   b. DataSource=Salesperson/Purchaser
3. In the 2nd row enter:
   a. Type = Column
   b. DataSource=Name
4. In the 3rd row enter:
   a. Type = DataItem
   b. DataSource=Sales Header
   c. Select the **View** menu and then click **Properties** to access the DataItem properties. Alternatively select the red check box icon to access the Properties.
   d. Specify the following for the DataItemLink property:
      - Salesperson Code=Salesperson_Purchaser.Code
   e. Note that the DataItemLinkType property is set to the following by default:

      <Use Default Values if No Match>

      As you will know from the earlier exercise, this means we expect to see a list of ***every*** Salesperson/Purchaser record (even if they do not have any matching Sales Header records).
   f. Close the properties window.
5. In the 3rd row enter:
   a. Type = Column
   b. DataSource=Amount
   c. Method Type=Totals
   d. Method = Sum
6. Save your new Query as object ID 50002 with the name MyQuery2.
7. Run your new query. You will see the following:



About This Query: - Preview of first 1000 rows

Name	Sum_Amount
Annette Hill	0
Bart Duncan	0
Debra L. Core	0
John Roberts	43030111.92
Linda Martin	0
Mary A. Demps...	0
Peter Saddow	28016.72
Richard Lum	0

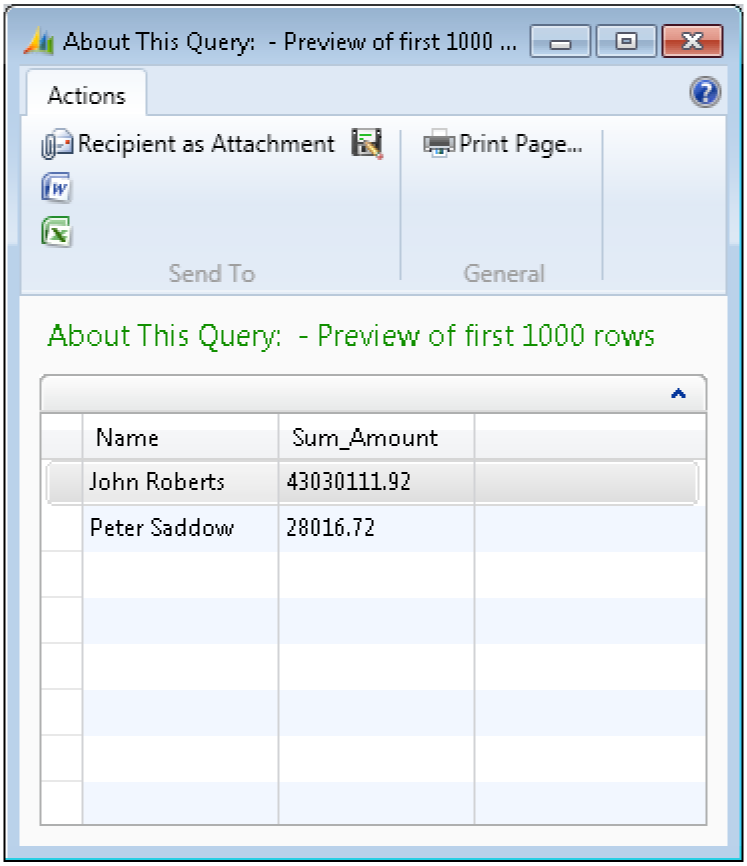
Figure 6

As expected, this is a list of all Salesperson/Purchaser records even where there is no Sales Record to match.

8. Edit the MyQuery2 object and open the Properties for the “Sales Header” DataItem.
9. Modify the DataItemTableFilter property to specify the following and then save the Query object:

   Order Date=FILTER(01/01/14..31/12/14)

10. Intuitively, one might expect that, as we have only changed the filtering criteria for the Sales Header DataItem, the results should show a list of every Salesperson/Purchaser with only the Sum_Amount slightly different to what we saw earlier based on the fact that only Sales Header amounts with an order date in 2014 would be included. In fact, something different happens.
11. Save the changes made to MyQuery2 and run the new version. You will see the following:

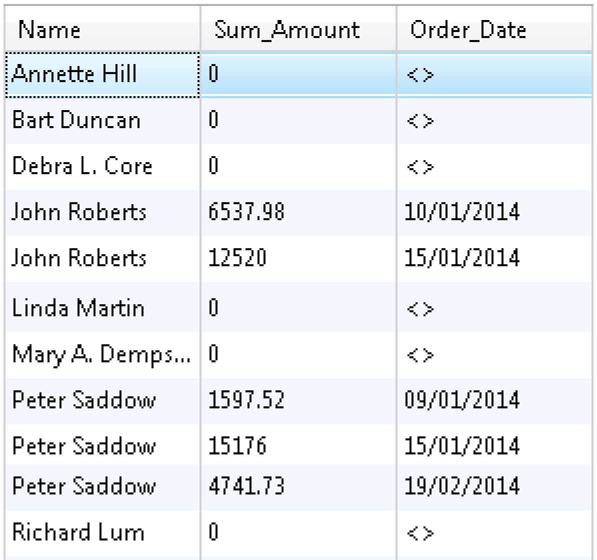

About This Query: - Preview of first 1000 ...
Actions
Recipient as Attachment
Print Page...
Send To
General
About This Query: - Preview of first 1000 rows

Name	Sum_Amount
John Roberts	43030111.92
Peter Saddow	28016.72

Figure 7

12. The above result happens because of the logic used by the back-end SQL Server T-SQL query. First it will create an intermediate result by joining the two tables according to the criteria specified by the following two Query Properties:
    a. DataItemLink = Salesperson Code=Salesperson_Purchaser.Code
    b. DataItemLinkType = <Use Default Values if No Match> (i.e. a left join)

You can imagine the intermediate result set would be something like this (although you do not see these intermediate results):



Name	Sum_Amount	Order_Date
Annette Hill	0	<>
Bart Duncan	0	<>
Debra L. Core	0	<>
John Roberts	6537.98	10/01/2014
John Roberts	12520	15/01/2014
Linda Martin	0	<>
Mary A. Demps...	0	<>
Peter Saddow	1597.52	09/01/2014
Peter Saddow	15176	15/01/2014
Peter Saddow	4741.73	19/02/2014
Richard Lum	0	<>

Figure 8

Next the T-SQL query logic will apply the filtering specified by the following Query Property:

DataItemTableFilter= Order Date=FILTER(01/01/14..31/12/14

This cause the intermediate result set to be filtered so that all records which do not have a qualifying Order Date in 2014 will be removed. This is why the final Query results will only show records for John Roberts and Peter Saddow despite the SQL Left Join approach.

The T-SQL query could be modified to include all Salespersons but only show their Sum_Amount values which qualify based on the Order Date filter. This can be achieved by specifying the Order Date filter as part of the LEFT JOIN statement. A future version of MICROSOFT DYNAMICS NAV may enable the Query object to have this extra filtering option.