MODULE 3: MAP YOUR FORM DATA INTO SQL

Using DBXL you can create mappings from your XML forms to your SQL tables.

In order to execute the steps in this document, you will need an InfoPath form that submits to DBXL. Luckily, we have already done this in the first two modules.

This module is compatible with both scenarios discussed so far in this training: Event Receiver and Integrated.

Here is an overview of this module:

CREATE A SQL TABLE TO STORE THE DATA

These steps will be demonstrated by the presenter.

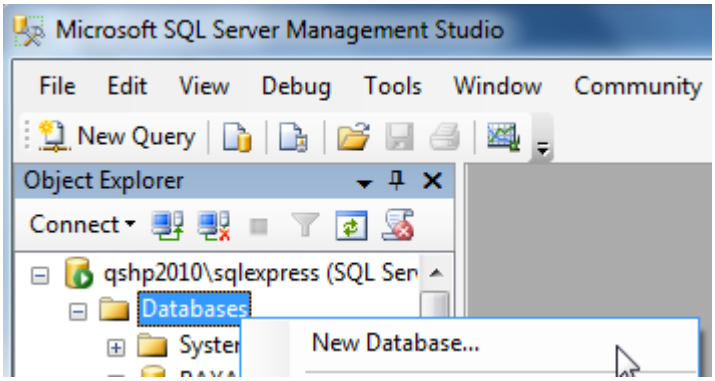
First, you will need to create a SQL database. This exercise will initially require one table.

1. Open SQL Server Management Studio Express.

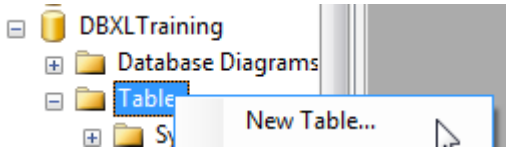
2. Connect to the SQL instance. For the purposes of this training module, we recommend using the same instance that was used to install the QdabraDBXL and QdabraUtility databases.

Note that we don't recommend adding tables to the same databases (**QdabraDBXL** and **QdabraUtility**) that DBXL uses.

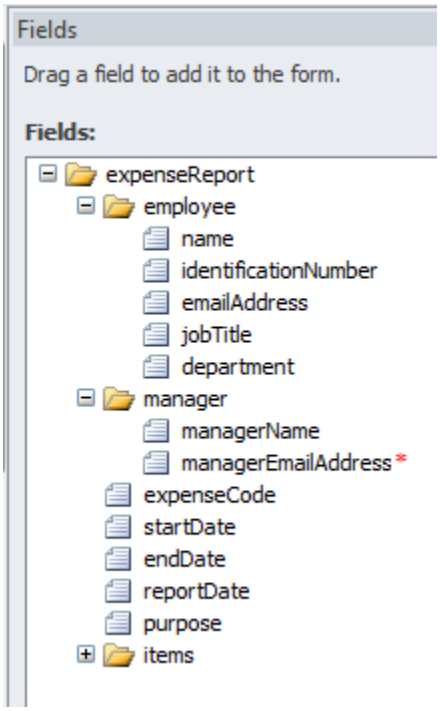
3. Create a new database and call it **DBXLTraining**.



4. Right click on that database, select **New Table**.



Recall your InfoPath form template's schema:



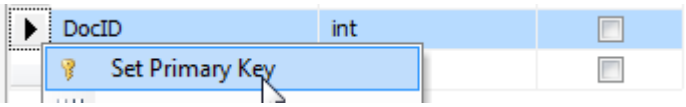
5. Add columns for the data you wish to store in the table, as shown below:

Column Name	Data Type
EmployeeName	nvarchar(50)
EmployeeIDNumber	nvarchar(50)
EmployeeEmailAddess	nvarchar(50)
EmployeeJobTitle	nvarchar(50)
EmployeeDepartment	nvarchar(50)
ManagerName	nvarchar(50)
ManagerEmailAddress	nvarchar(50)
ExpenseCode	nvarchar(50)
StartDate	date
EndDate	date
ReportDate	date
Purpose	nvarchar(50)

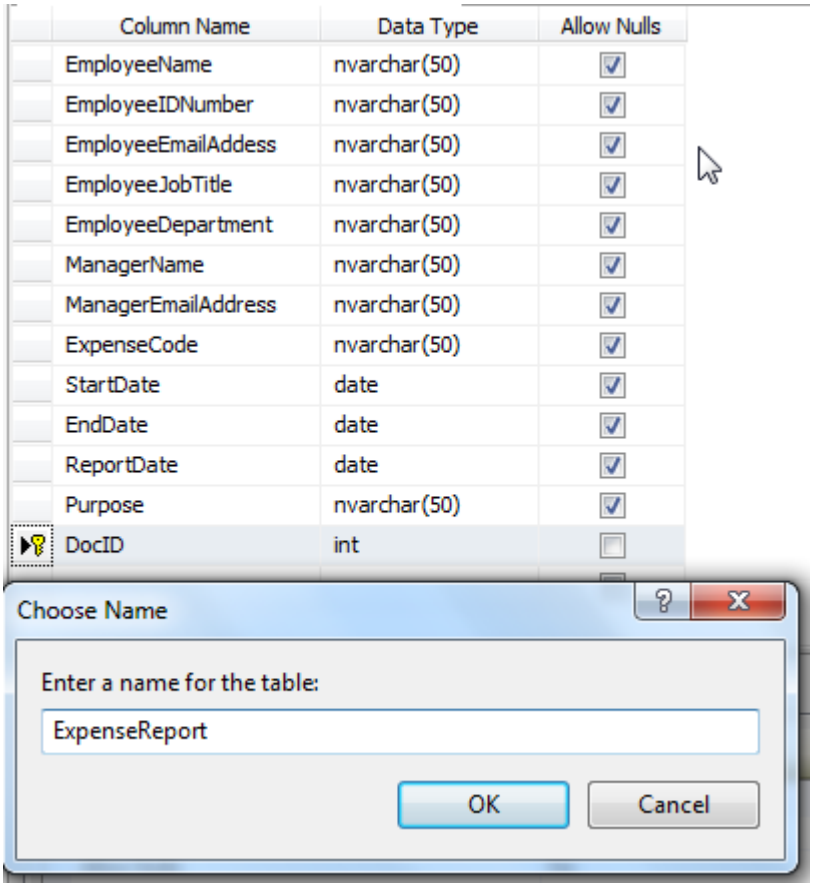
Your table can have a few columns or many columns. It depends on which data from the form you wish to store in SQL.

Column Name	Data Type	Allow Nulls
EmployeeName	nvarchar(50)	☑
EmployeeEmailAddress	nvarchar(50)	☑
ManagerName	nvarchar(50)	☑
ManagerEmailAddress	nvarchar(50)	☑
ExpenseCode	nvarchar(50)	☑
ReportDate	datetime	☑
Purpose	nvarchar(50)	☑

6. Add an additional column called **DocID**, with data type **int**, and uncheck **Allow Nulls**. Set this column as the Primary Key.



7. Save the table as **ExpenseReport**.


Column Name
Data Type
Allow Nulls
EmployeeName
nvarchar(50)
EmployeeIDNumber
nvarchar(50)
EmployeeEmailAddess
nvarchar(50)
EmployeeJobTitle
nvarchar(50)
EmployeeDepartment
nvarchar(50)
ManagerName
nvarchar(50)
ManagerEmailAddress
nvarchar(50)
ExpenseCode
nvarchar(50)
StartDate
date
EndDate
date
ReportDate
date
Purpose
nvarchar(50)
DocID
int
Choose Name
Enter a name for the table:
ExpenseReport
OK
Cancel

CONFIGURE A SIMPLE SQL MAPPING IN DAT

In the following steps, you will be instructed to setup database mapping via the Database Accelerator Administration Tool (DAT) for the published form template. Our goal is to collect submitted information from a non-repeating group in the form into a corresponding row in the table.

8. Open the DBXL Administration Tool.

9. Click **Edit** for the Document Type.

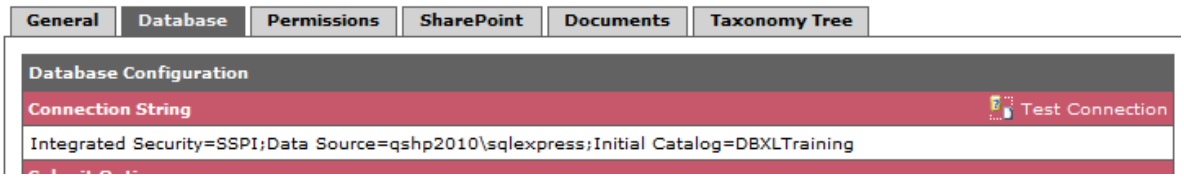
10. In DAT, click on the **Database** tab.

11. Enter the Data Connection String appropriate for your SQL server. For example, ***Integrated Security=SSPI;Data Source=(LOCAL)\sqlexpress;Initial Catalog=DBXLTraining***

Hint! If you don't know the connection string, you can modify any one of the existing document types (installed by DBXL) and copy the connection string. You can then update this string for your purposes.

12. Tab out of the Connection String field.

13. Click **Test Connection**, and click **OK** when success is confirmed.



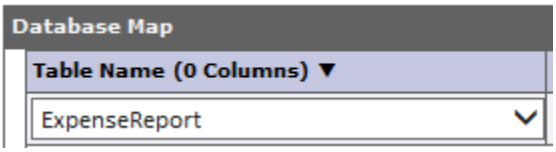
Should you get a permissions error, the easiest way to troubleshoot the error is to compare the Security for the QdabraDBXL database and the DBXLTraining database. The DBXLTraining database might be missing permissions. Simply refer to the QdabraDBXL database as a reference of users and permissions to add.

14. Click **Save**, and then click **OK** in the confirmation dialog.

Now you can construct the database mapping.

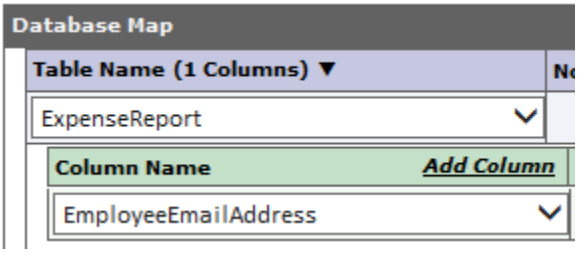
15. Click **Add Table**.

16. Use the dropdown to select the correct table.

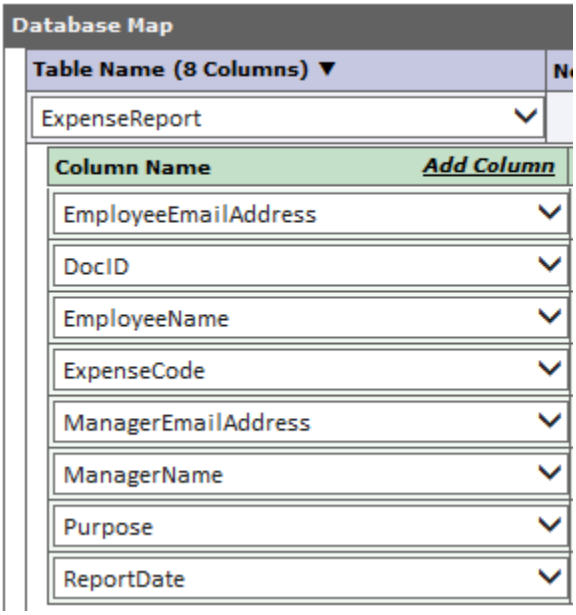


17. Click **Add Column**.

18. Use the dropdown to select the first column, **EmployeeEmailAddress**, in our case.



19. Repeat! Click Add Column and use the dropdown. This allows you to add all the columns for your table.



The steps above added the “destination”, that is, the SQL table and columns where the data will be stored. Now you will add the “source”, that is, the InfoPath form template nodes where the data comes from.

20. Click on the **Select Schema Node** icon ( ) under **Node Path**, for the table.

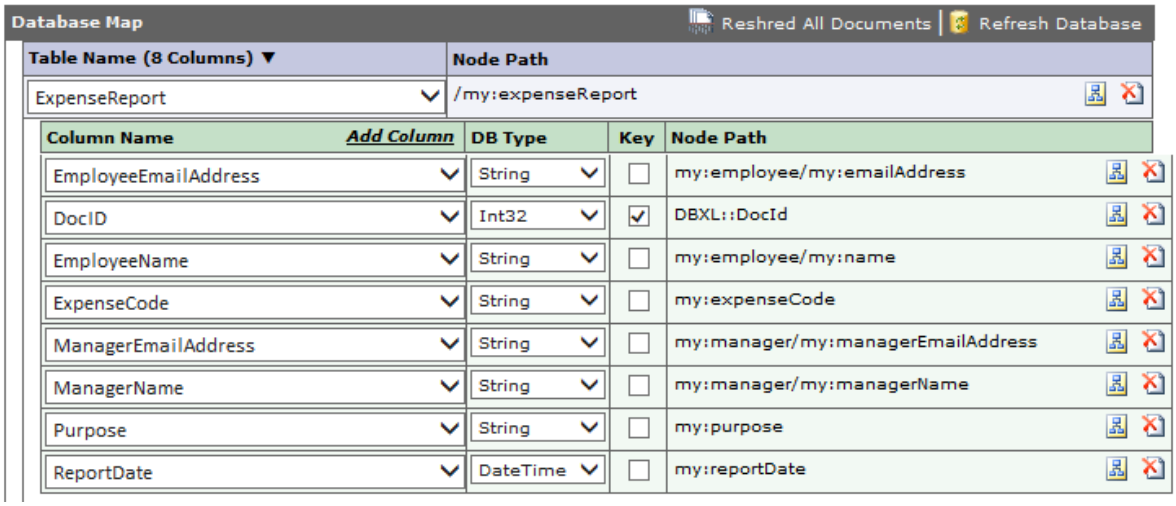
21. In the taskpane, double click on the my:expenseReport node.

22. Click on the **Select Schema Node** icon ( ) under **Node Path**, for the **EmployeeEmailAddress** column.

23. In the taskpane, double click on the **my:employee/my:emailAddress** node.

24. Repeat the previous two steps for the rest of the rows in the mapping, always using the taskpane to select the form node.

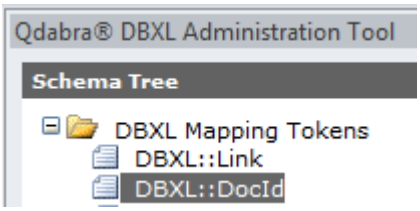
Your mapping should look like this:



What about the DocID?

25. Click on the **Select Schema Node** icon ( ) under **Node Path**, for the **DocID** column.

26. In the taskpane, double click on the **DBXL::DocId** node.



27. Click **Save**, and then click **OK** in the confirmation dialog.

Let's find out if the mapping is working.

VERIFY MAPPING

If you have existing documents, you can simply click “Reshred All Documents” in the Database tab. If no documents exist, submit new ones to test:

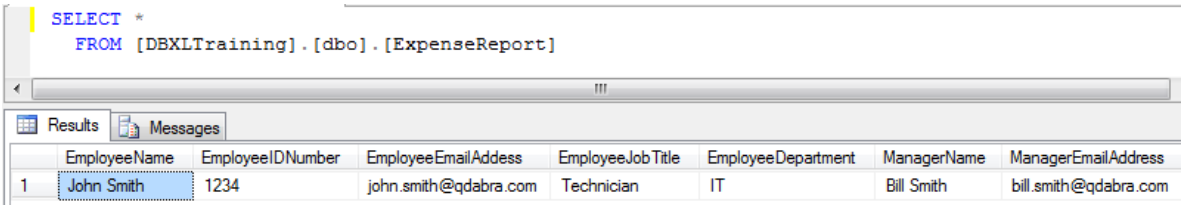
28. Click on the **General** tab.

29. Click **Open**.

30. Fill out the form. When done, click the Submit button.

31. Click the **Documents** tab in DAT. The new document is added to the **Documents** table.

32. Use SQL Server Management Studio Express to verify that the new document has been mapped to your table. This step will be demonstrated by the presenter.



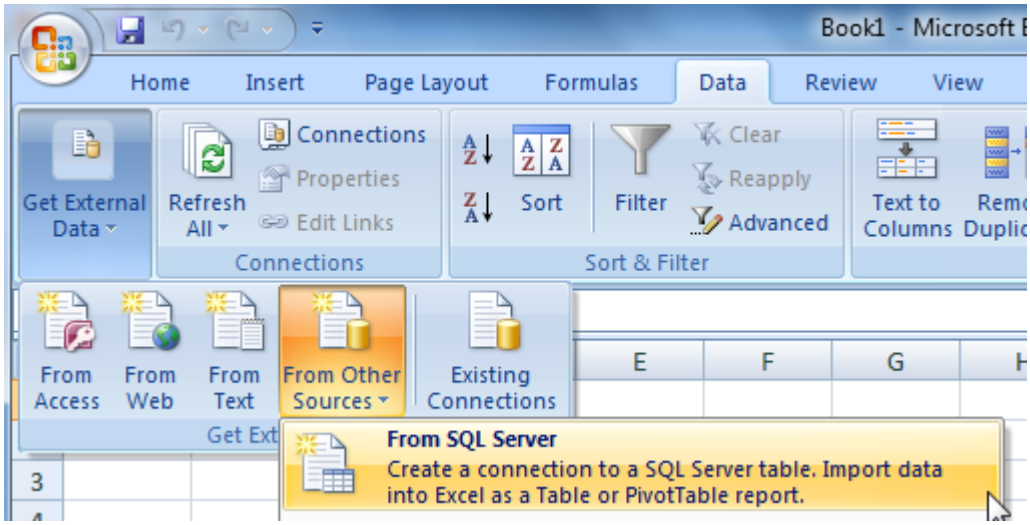
The mapping between DBXL and SQL has now been created and verified!

CREATE AN EXCEL REPORT

These steps will show how to create a simple Excel Report based on this data.

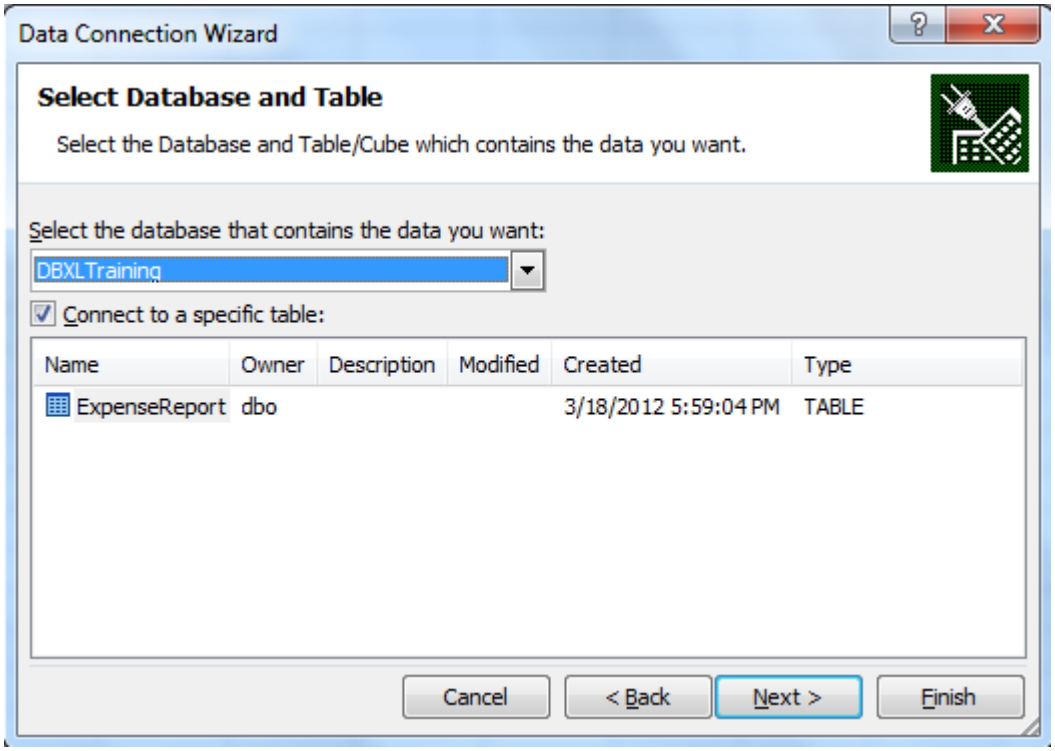
33. Launch Microsoft Excel.

34. Select **Data** > **From Other Sources** > **From SQL Server**.

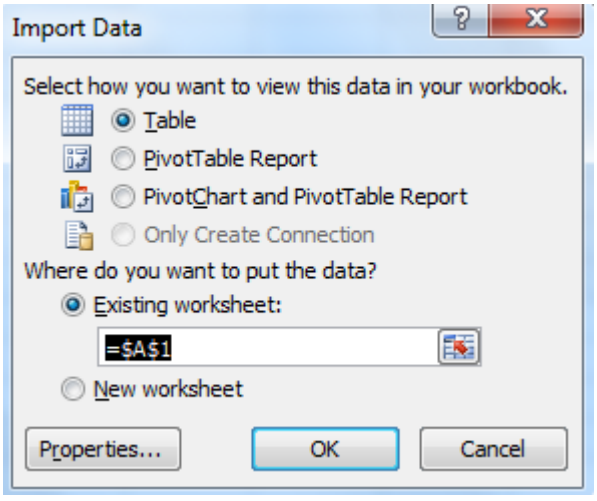


35. Enter the SQL instance name and click **Next**.

36. Select the correct database and table, and click **Finish**.



37. Click OK in the Import Data dialog. Though there are various options in this dialog, we'll just accept the defaults.



Excel now creates a table in the worksheet, showing the data that DBXL mapped to SQL.

	A	B	C	D	E	F
1	EmployeeName	EmployeeIDNumber	EmployeeEmailAddess	EmployeeJobTitle	EmployeeDepartment	ManagerName
2	Ernesto Machado					
3	John Smith	1234	john.smith@qdabra.com	Technician	IT	Bill Smith

38. Go back to the DBXL Administration Tool, switch to the **General** tab, and click **Open**.

39. Fill out and submit a new document.

40. Back in Excel, in the **Data** tab, click **Refresh All**.

The new record is now displayed in your Excel report:

	A	B	C	D	E	F
1	EmployeeName	EmployeeIDNumber	EmployeeEmailAddess	EmployeeJobTitle	EmployeeDepartment	ManagerName
2	Ernesto Machado					
3	John Smith	1234	john.smith@qdabra.com	Technician	IT	Bill Smith
4	Jane Doe	97391	jane.doe@qdabra.com	Project Manager	Sales	Bill Smith

41. Close Excel. Save the report if you'd like.

This is the simplest possible example, where the data is flat. But what happens if the data is not flat? The next section will walk you through modifying the form such that it has a parent-child relationship. Then, you will modify the database mapping to reflect this new structure.

ADD A CHILD TABLE IN SQL

These steps will be demonstrated by the presenter.

Now we want to map the repeating data for **Items**, and to do that we need an additional SQL table.

42. In SQL Server Management Studio, create a new table and call it ExpenseReportItems. It will have the columns shown in the screenshot below. Remember: add as many or as few columns as you'd like.

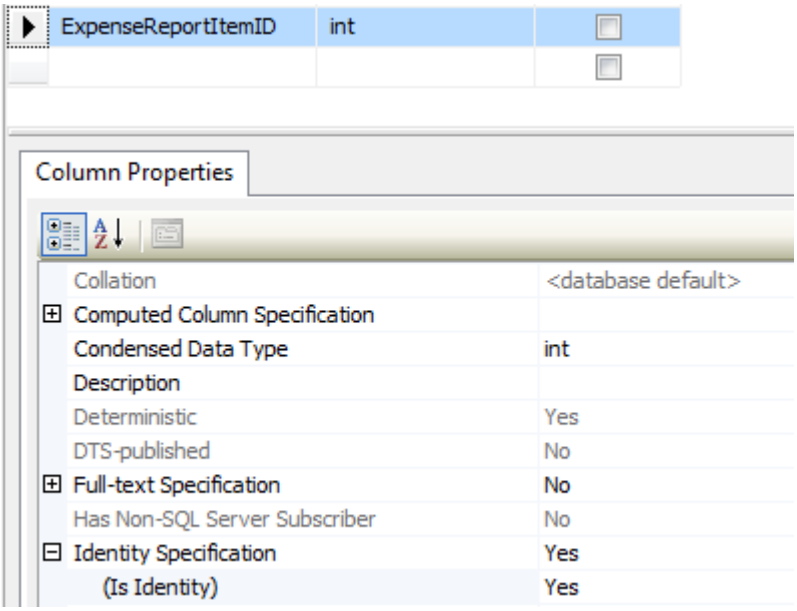
Column Name	Data Type
ExpenseReportId	bigint
Date	datetime
Description	nvarchar(128)
Amount	decimal(18, 2)
Category	nvarchar(50)
MerchantName	nvarchar(128)
EntertainmentPurpose	nvarchar(512)
EntertainmentPlace	nvarchar(100)
AdditionalInfo	nvarchar(MAX)

Why do we need the ExpenseReportId in the child table? Because this is what links our child (Items) data to its parent table.

43. Add an additional column called ExpenseReportItemID.

44. Set this column to be the Primary key, and also set it to auto-increment (Identity Specification = Yes).

45. Save this new table.


ExpenseReportItemID
int
Column Properties
Collation
<database default>
Computed Column Specification
Condensed Data Type
int
Description
Deterministic
Yes
DTS-published
No
Full-text Specification
No
Has Non-SQL Server Subscriber
No
Identity Specification
Yes
(Is Identity)
Yes

ADD MAPPING FOR THE CHILD TABLE

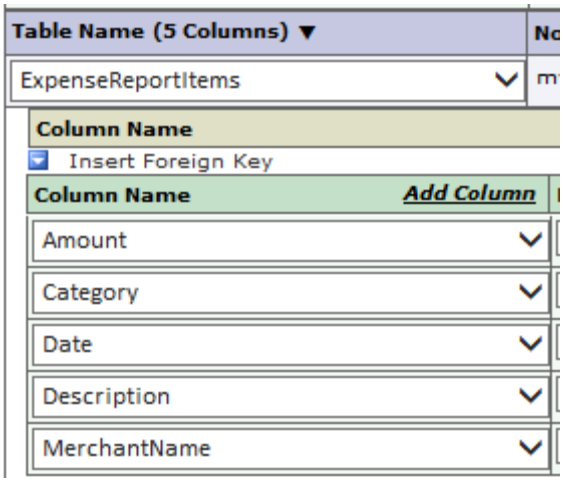
46. Back in the DBXL Administration Tool, click **Edit** for the document type.

47. Switch to the **Database** tab.

48. Click on **Add Sub Table**.

49. Use the dropdown to select the **ExpenseReportItems** table.

50. Click **Add Column**, directly under the new row in the mapping. Use the dropdown to select the column in the SQL table. Repeat!



51. Click on the **Select Schema Node** icon under **Node Path**, for the **ExpenseReportItem** table.

52. In the taskpane, double click on the **my:items/my:item** node.

53. Click on the **Select Schema Node** icon under **Node Path**, for the **Amount** column.

54. In the taskpane, double click on the **my:amount** node.

55. Repeat the previous two steps for the other rows.

Now you will link the two tables together.

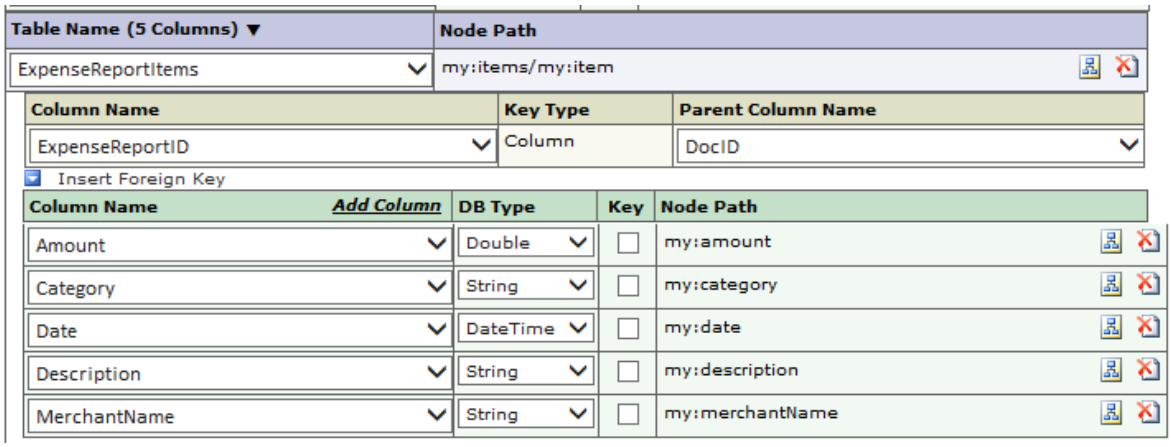
56. Click on **Insert Foreign Key**.

57. Click on the **Select Database Column** icon.

58. Double click on **ExpenseReportId** in the taskpane.

59. Under Parent Column Name, use the dropdown to select the DocID column.

Your mapping should look like this:



You might be left wondering why we did not map the ExpenseReportItemID column. It's because we set that be an autoincrement, primary key when we created the table. Therefore, there is no need to map a value from the InfoPath form.

60. Click **Save**, and then click **OK** in the confirmation dialog.

Now let's verify the mapping!

61. Click **Reshred All Documents**, and then click OK in the confirmation message.

62. Use SQL Server Management Studio Express to verify that the item information has been mapped to your table.

```
SELECT *
  FROM [DBXLTraining].[dbo].[ExpenseReportItem]
```

Results | Messages

	ExpenseReportId	Date	Description	Amount	Category
1	6461	2012-03-02 00:00:00.000	Hotel	100.00	Lodging
2	6461	2012-03-03 00:00:00.000	Dinner	50.00	Meals
3	6462	2012-03-01 00:00:00.000	Lunch	10.00	Meals
4	6462	2012-03-16 00:00:00.000	Dinner	20.00	Meals
5	6463	2012-03-08 00:00:00.000	New computer	1000.00	Office supplies

CONDITIONAL SHREDDING

As we have seen, DBXL allows us to map information from the xml documents into SQL tables. There are instances where we do not want all mapped data to propagate to the tables, let's say, fields containing NULLs. It turns out that one can leverage the power of XPath to create conditions on when shredding can occur.

To implement this, we only need one small change to our existing database mapping.

63. In the DBXL Administration Tool, switch to the Database tab.

64. Modify the Node Path for the ExpenseReport table:

The Node Path is now: ***/my:expenseReport[my:manager/my:managerName !=""]***

65. Modify the Node Path for the ExpenseReportItem table:

The Node Path is now: ***my:items/my:item[my:description !=""]***

66. Click **Save**, and then click **OK** in the confirmation dialog.

67. To verify this change, simply submit a new form, but without any information for items.

This guarantees that incomplete data will not be mapped to SQL and used for reporting.

```
SELECT *
  FROM [DBXLTraining].[dbo].[ExpenseReport]
  SELECT *
  FROM [DBXLTraining].[dbo].[ExpenseReportItem]
```

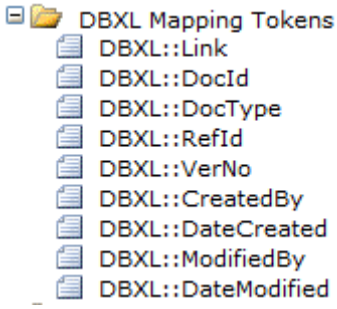
Results | Messages

	EmployeeName	EmployeeIDNumber	EmployeeEmailAddess	EmployeeJobTitle	EmployeeDepartment
1	John Smith	1234	john.smith@qdabra.com	Technician	IT
2	Jane Doe	97391	jane.doe@qdabra.com	Project Manager	Sales

	ExpenseReportId	Date	Description	Amount	Category	MerchantName
1	6462	2012-03-01 00:00:00.000	Lunch	10.00	Meals	NULL
2	6462	2012-03-16 00:00:00.000	Dinner	20.00	Meals	NULL
3	6463	2012-03-08 00:00:00.000	New computer	1000.00	Office supplies	NULL

MAPPING DBXL TOKENS

When creating your database mapping in the DBXL Administration Tool (DAT) you might have noticed nodes under a folder called “DBXL Mapping Tokens”.



That group contains (for example) the document’s link, its DocID, the DocType it belongs to, its RefID and its version number.

In our example above, we only used the DocID, but you can use any of these tokens to map data into your SQL database, as seen below.

Link	String		DBXL::Link
DocID	Int32		DBXL::DocId
DocType	String		DBXL::DocType
RefId	Int32		DBXL::RefId
VerNo	String		DBXL::VerNo
CreatedBy	String		DBXL::CreatedBy
DateCreated	String		DBXL::DateCreated
ModifiedBy	String		DBXL::ModifiedBy
DateModified	String		DBXL::DateModified