CHAPTER 8

Concluding the ETL Process with SSIS

Sometimes good things fall apart so better things can fall together.

—Marilyn Monroe

Thus far, the ETL process has taken us through a lot of decision making. Data has had to be cleaned up and handled with care to ensure it is accurate and that it can be worked with. In Chapter 7, we created an SSIS package and set up our data connections. We also configured and tested the Execute SQL tasks.

In this chapter, we continue the SSIS process by configuring data flows and data destinations that enable us to load our data warehouse, and we handle errors and round out this subject by executing the entire package.

Data Flows

Data Flows are the most complex of the SSIS tasks, because they are made up of a composite of many subcomponents. There are three different types of data flow subcomponents: sources, transformations, and destinations. Most often sources pull data from tables and views, but you can also extract data from multiple file types, and even SSAS cubes. Transformations modify, summarize, and clean data. Destinations load data into tables, cubes, files, or in-memory datasets that can be used by other tasks within the SSIS package itself.

Data flows are configured using the Data Flow tab. You can access this tab at the top of the package designer window or right-click any data flow task and select Edit from the context menu. Either way, the user interface navigates to the Data Flow tab (Figure 8-1).

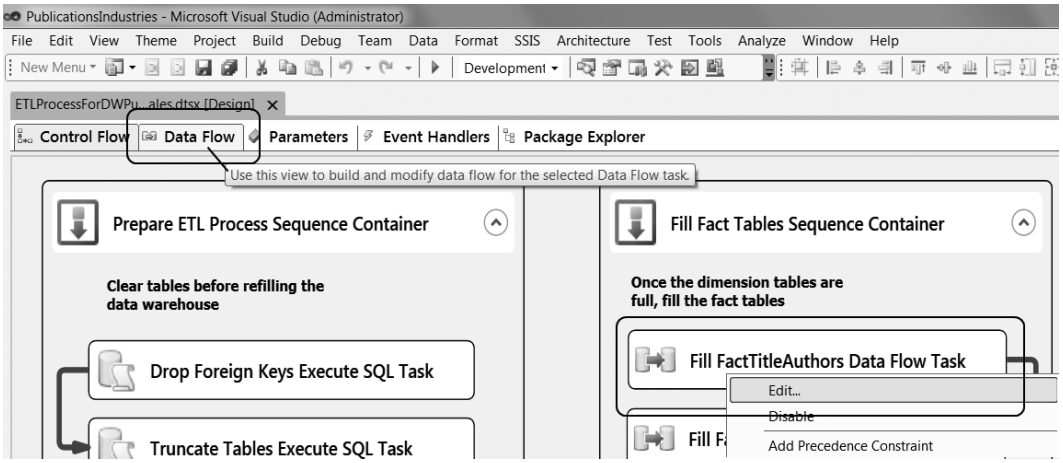


Figure 8-1. *Editing a data flow task*

All the data flows in your SSIS package are edited from one tab. The dropdown box at the top of the Data Flow tab allows you to select which data flow task you would like to work with, as shown in Figure 8-2.

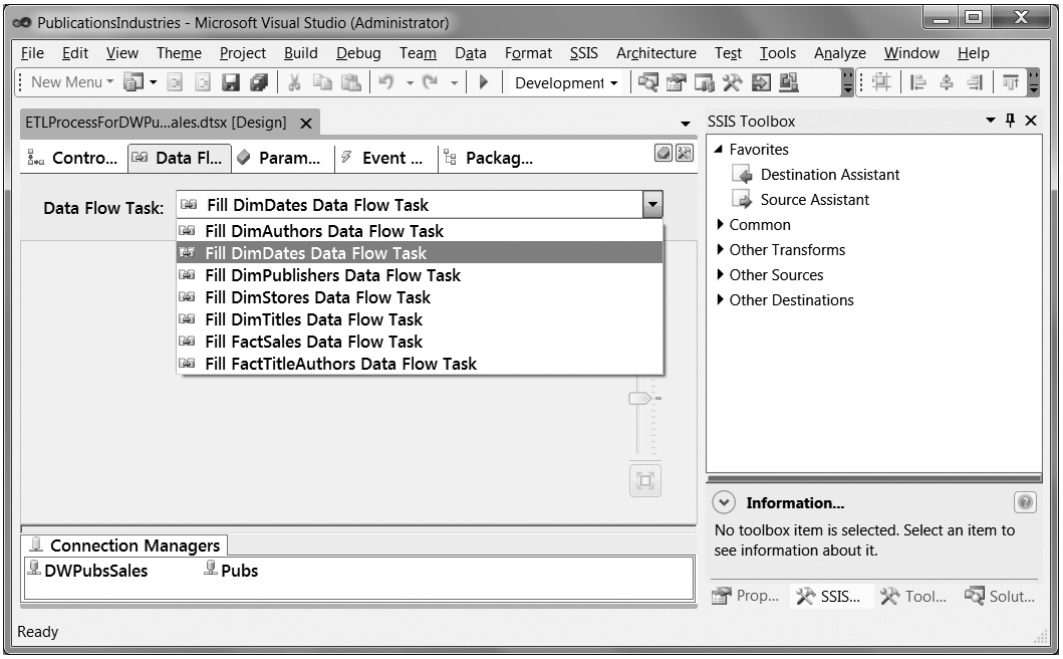


Figure 8-2. *Selecting between different data flow tasks*

Outlining a Data Flow Task

Data flow tasks require at least one data source component and one data destination component. Often you just need one of each, but you can have more than one data source and more than one data destination per data flow.

To outline your Data Flow task steps, begin by placing data source and data destination components from the Toolbox onto the designer surface (Figure 8-3). At this point, even though it is possible to connect them with arrows between each of the components, do not do so! Adding arrows before configuring the components causes problems. The arrows of the data flow do not work the same as the arrows on the Control Flow tab. Control flow arrows represent a precedent constraint and control the order of execution.

Data flow arrows represent a data flow path and pass metadata from the component at the arrow's origin to the component at the end of the arrow. If arrows are connected between components before the source component is configured, there will be no metadata to pass on to the end component. Therefore, connect the arrows only *after* each source component is configured.

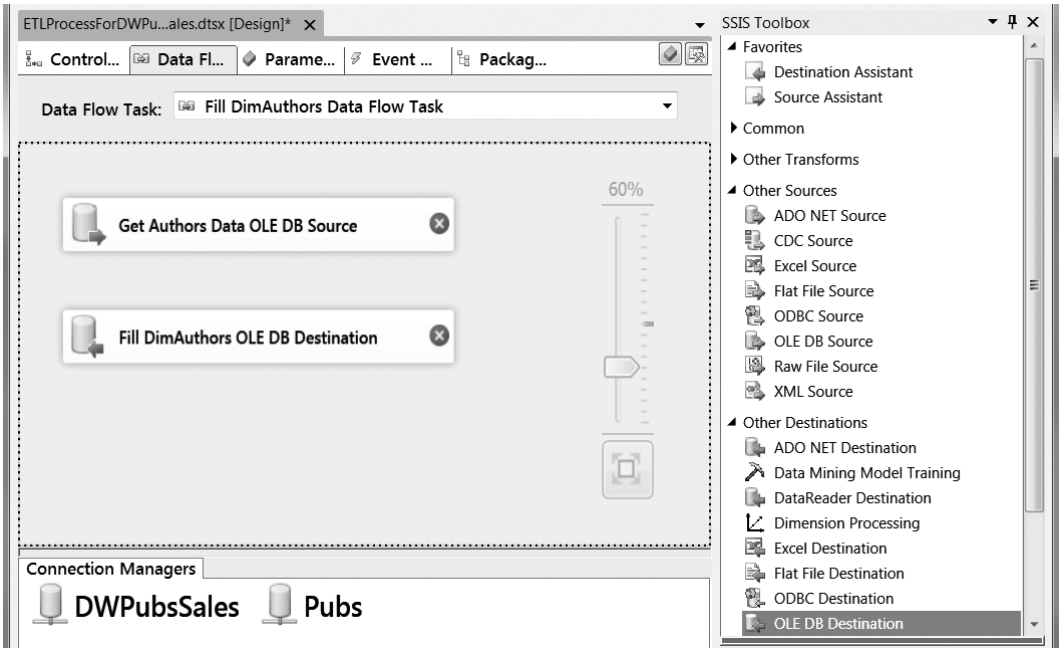


Figure 8-3. *Outlining a data flow task should not include a data flow path*

Configuring the Data Source

A data source component is the heart of each data flow task. You can add a data source component to the data flow by clicking the SSIS Toolbox and locating either the Source Assistant or one of the data source component lists under the Other Sources category. Selecting the Source Assistant will launch a wizard that helps you select one of the other sources. Therefore, you will not need to use it when you already know which type of component you want.

You should always choose the data source and destination components to match the connection manager objects in your SSIS package. We are using OLE DB connections in our SSIS package; therefore, we want to use an OLE DB Source component.

When you click a component, you will see data path arrows displayed, but remember that until the component is properly configured, you cannot use them to connect to other components. A data source component is configured by right-clicking the component and choosing Edit from the context menu.

■ **Note** In Figures 8-3 and 8-4, the two data path arrows represent a blue data flow path and a red error output path. Previous versions of SQL use green for the data flow rather than blue, but the function is the same. For more information, see "Data Flows: Data Flow Paths" later in this chapter.

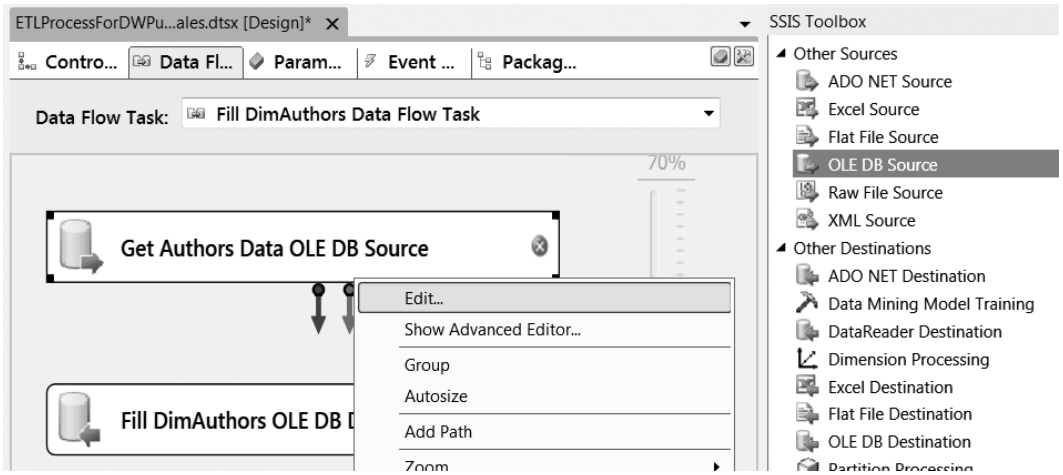


Figure 8-4. *Editing a data source*

The OLE DB Source Editor

Selecting Edit from the context menu shown in Figure 8-4 launches the OLE DB Source Editor window—a window that you will become quite familiar with if you work with SSIS for any length of time!

The OLE DB Source Editor window consists of three separate pages:

- The Connection Manager page
- The Columns page
- The Errors Output page

The Connection Manager Page

On the Connection Manager page, select which connection manager object to use from the "OLE DB connection manager" dropdown box (Figure 8-5). If you have not previously created a connection manager for the SSIS package, the dropdown box will not offer any selections. The connection mangers are typically created first, but if you have not done so yet, click the New button and create one.

■ **Tip** One connection manager can be used by many different data flow components. It is a common mistake for developers to create a new connection for each separate component using the New button.

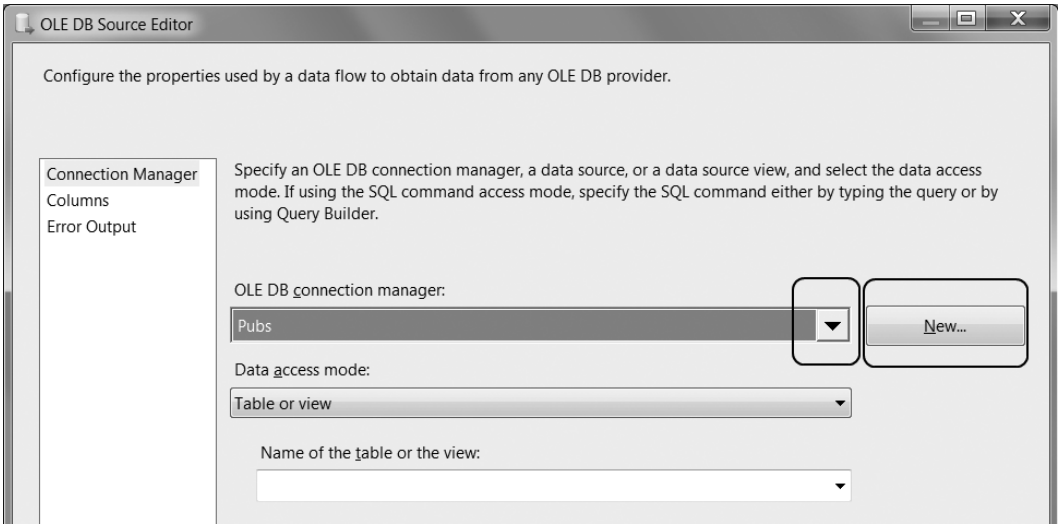


Figure 8-5. *Configuring the Data Source connection manager*

After you have chosen your connection, you will want to choose your data access mode. As you can see in Figure 8-6, you can select from four separate options in an OLE DB connection manager:

- Table or view
- Table name or view name variable
- SQL command
- SQL command from a variable

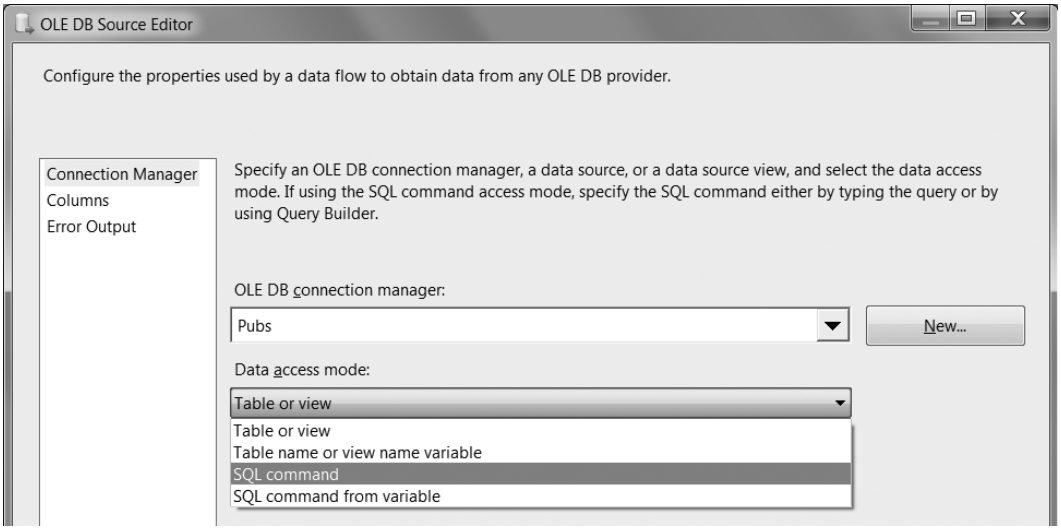


Figure 8-6. *Configuring the data source data access mode*

The OLE DB Source Connection Manager Page

The "Table or view" option shown in Figure 8-7 allows you to select a table or view from the data source. The table name and view name variables do the same thing, as long as you place the name of the table or view in an SSIS variable. When you use the "Table or view" option, you are presented with a list of tables and views to select from, based on your chosen connection manager.

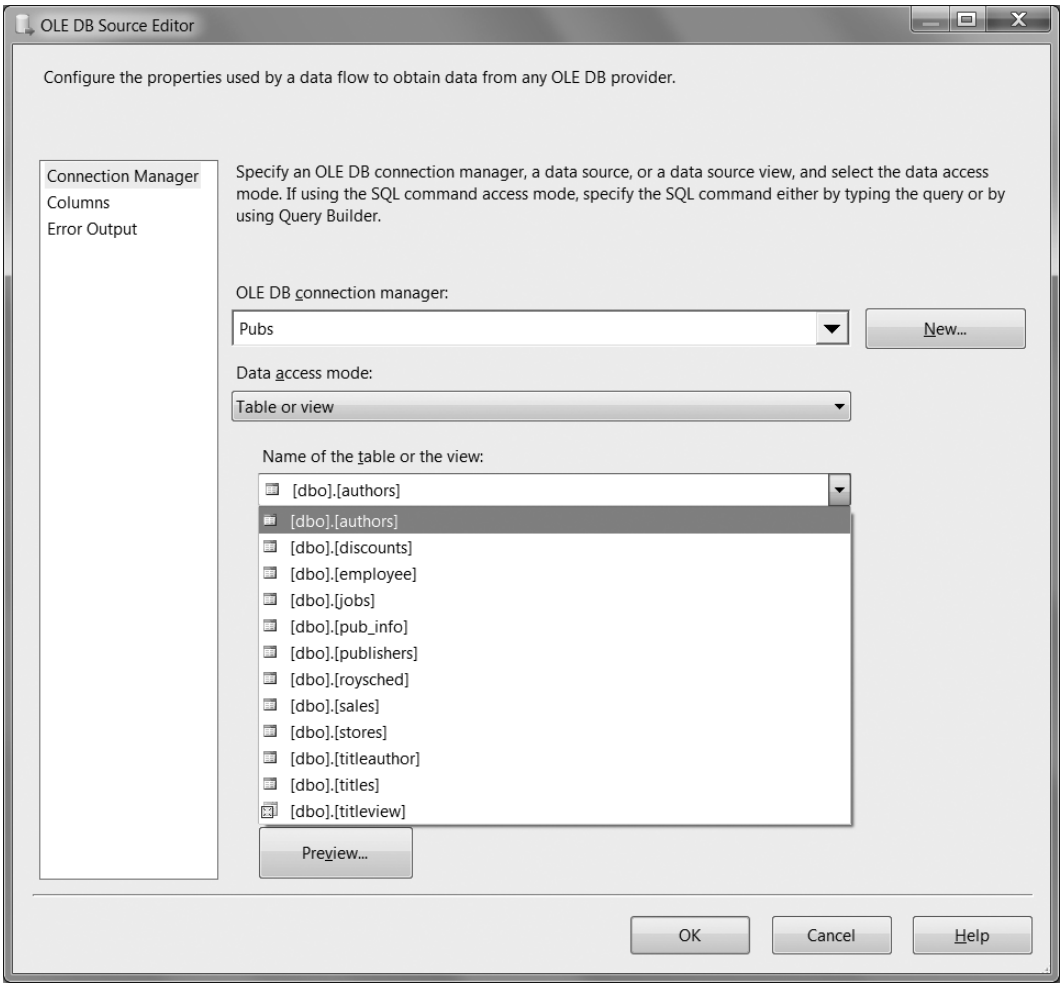


Figure 8-7. *Using the "Table or view" data access mode*

■ **Note** Using either one of these options is similar to using `SELECT * FROM <Some table>`, bringing all of the columns from that table even if they will never be of use. Therefore, always try to use either a SQL view that restricts the data you get from a table or the "SQL command" data access mode and include only the required columns in your SQL code.

SQL Command and SQL Command from a Variable

The "SQL command" choice allows you to type or copy and paste a SQL statement into the command window (Figure 8-8). As you might have already concluded, the SQL command from a variable does the same thing, as long as you place your SQL code into an SSIS variable first. You can use either of these data access modes to execute stored procedures as well.

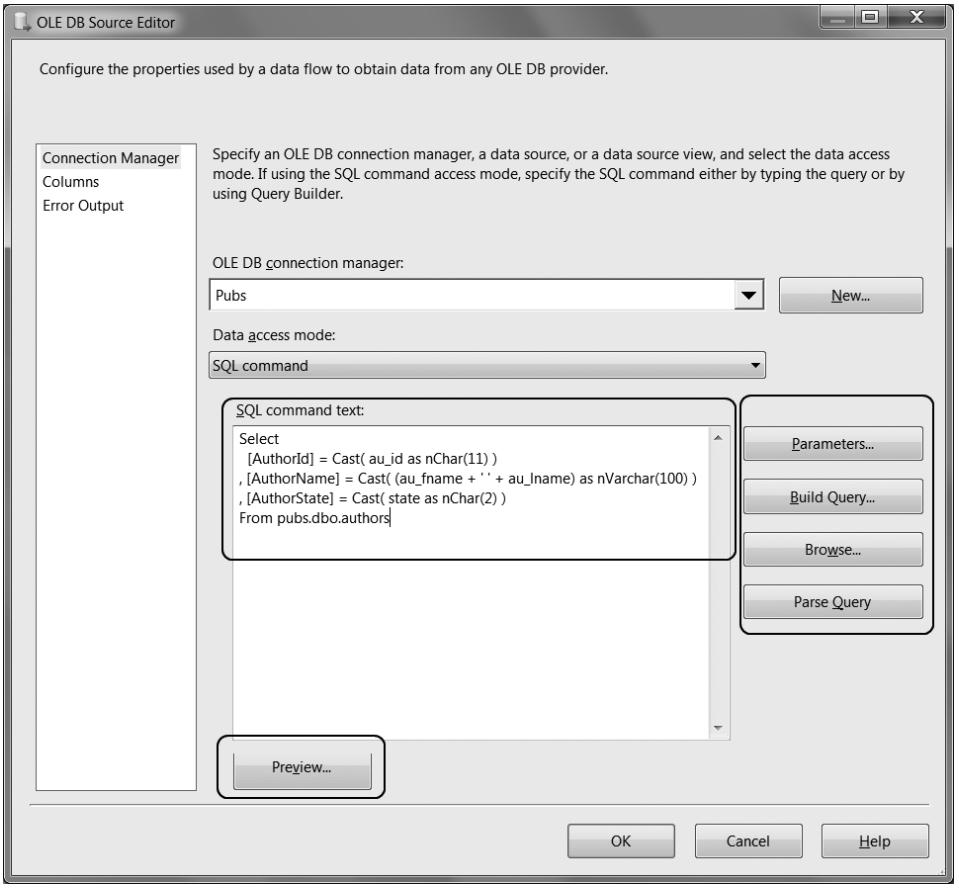


Figure 8-8. *Using the SQL command data access mode*

■ **Tip** The recommended practice is to use SQL Server Management Studio to create and test your SQL code. The steps involved include writing the select statement, wrapping the statement into a SQL stored procedure, testing that it works as expected, and then using the stored procedure's name in the SQL command text window. The stored procedure would include most, if not all, the ETL transformations required and provides your source component with clean transformed data. All the actual ETL transformation processing happens within the database engine as the stored procedure is executed. This is both faster and less prone to errors. You may remember that in Chapter 6 we created all the select statements we needed, but to keep things simple, we did not put them into stored procedures. In a production environment, you should consider using stored procedures instead.

Let's discuss the purpose of several useful buttons in the OLE DB source editor window, as indicated in Figure 8-8:

- *Parameters button*: Allows you enter a parameterized query in the query text using the question mark symbol (?) for OLE DB sources or the at (@)sign for ADO.NET sources
- *Build Query button*: Allows you to build a SQL with the Query Builder dialog window, similar to the one in SQL Server Management Studio (also discussed in Chapter 6)
- *Browse button*: Allows you to look for a file that contains the SQL code you want to run
- *Parse Query button*: Checks the syntax of the SQL code and verifies if the objects exist in the connected database
- *Preview button*: Runs the SQL code and shows you the results

---

■ **Note** Some of these buttons do not become available until code is typed in the window.

The Columns Manager Page

The Columns Manager page allows you to filter out any columns that were part of the input but no longer wanted as part of the output. Unchecking the checkbox next to a column name will remove it from the output of your data source.

Figure 8-9 shows all of the columns checked. Remember that we explicitly selected the SQL query, and typically we would use all of them. If we chose to use a table or view name, however, instead of a SQL query, we could uncheck any columns we did not need.

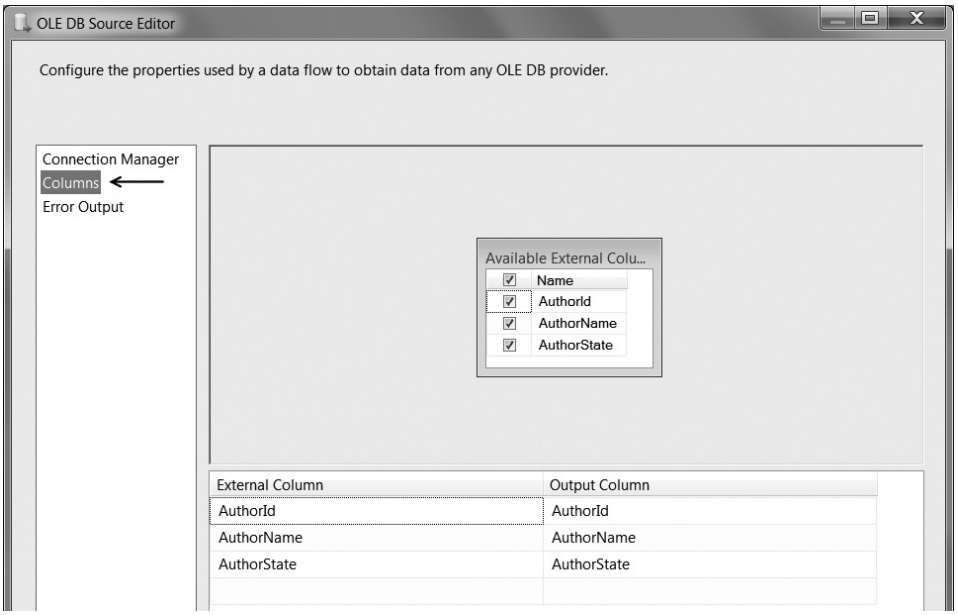


Figure 8-9. *Viewing the Columns page of the OLE DB data source editor*

■ **Tip** Sometimes, the data source component will not properly register the internal XML code unless you click the Columns page. Therefore, we recommend you always click the Columns page even if you are not going to make any additional configurations. It is simple to do and helps avoid this occasional bug.

The Error Output Page

The Error Output page allows you to redirect data that produces an error to a separate error output path. The idea is that if an error of some type occurs, you could route the rows causing these errors to a file on your hard drive or perhaps a SQL table that was made to hold incorrect or inconsistent data. Figure 8-10 shows this configuration page.

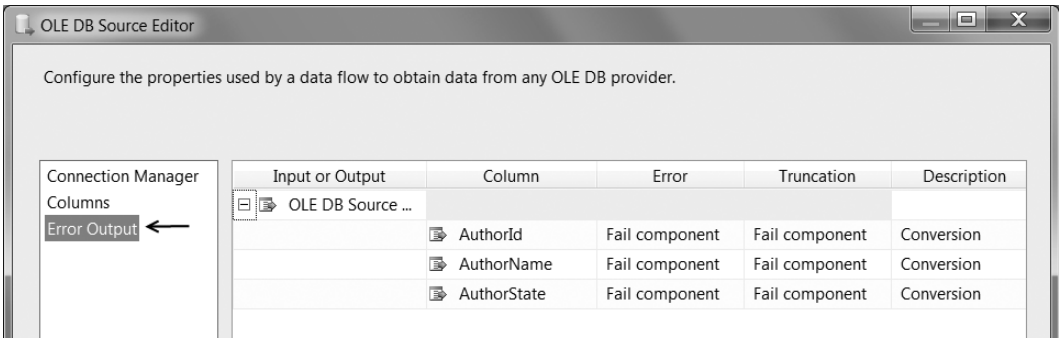


Figure 8-10. *Viewing the Error Output page of the OLE DB Source Editor window*

To understand error outputs better, you need to first understand what an error output path is. To understand that, you need to understand what a data flow path is. So, without further ado, let's take a look at each of these in order.

Data Flow Paths

To advance our discussion of data flow paths, let's start with a short review. When you place a data source component on the surface of the Data Flow designer surface, you will notice that both a blue line and a red line appear (Figure 8-4). These lines represent data flow and error flow paths, respectively. Data flow paths look similar to control flow precedence constraints, but they behave very differently.

Precedence constraints contain conditional logic that defines the programmatic flow of the tasks in an SSIS package. Each data flow path originates in a source component and ends in a destination component and does not have a way to conditionally define program flow. Instead, data flow paths contain metadata from the original data source component and any subsequent transformation components that come after. The idea is to pass all the metadata from the source component and all intermediate components on to the final destination component so that the final destination component can be properly configured. To see this metadata, double-click the data flow path arrow, and a dialog window will appear like the one shown in Figure 8-11.

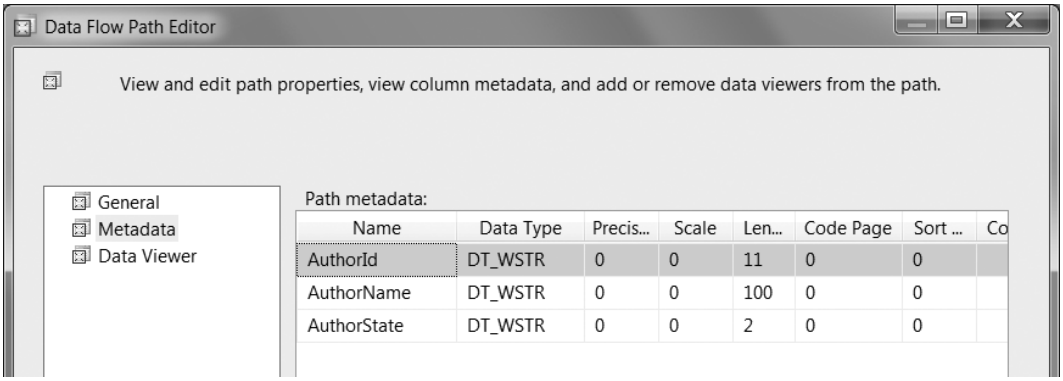


Figure 8-11. *The Data Flow Path Editor window*

If you try to connect the arrows before the source component is configured, there will be no metadata to pass on to the destination component. This causes errors that are tedious to resolve—not difficult, just tedious. We recommend avoiding this practice and connecting the arrows only after properly configuring each beginning-point component. Be sure that you do not open the destination component editor until after the source component metadata is available.

■ **Important** The most common problem we see when working with data flows is connecting the blue or red arrows during the outlining process. If, for some reason, you connect an arrow between two components before the source component is configured, simply delete the destination component and try again. It is possible to fix the issue without deleting components, but most of the time, deleting is the simplest or fastest way to resolve the issue.

Error Outputs Paths

Error flows have the ability to route data to a separate location in case of an error. The error output path arrow looks similar to the data flow path arrow, but it is red in color, instead of blue (Figure 8-12). To use this feature, first add an additional destination component, such as the additional OLE DB destination component shown in Figure 8-12, and then connect the error output path arrow to the new destination component. The destination component will usually be a flat file or a SQL table.

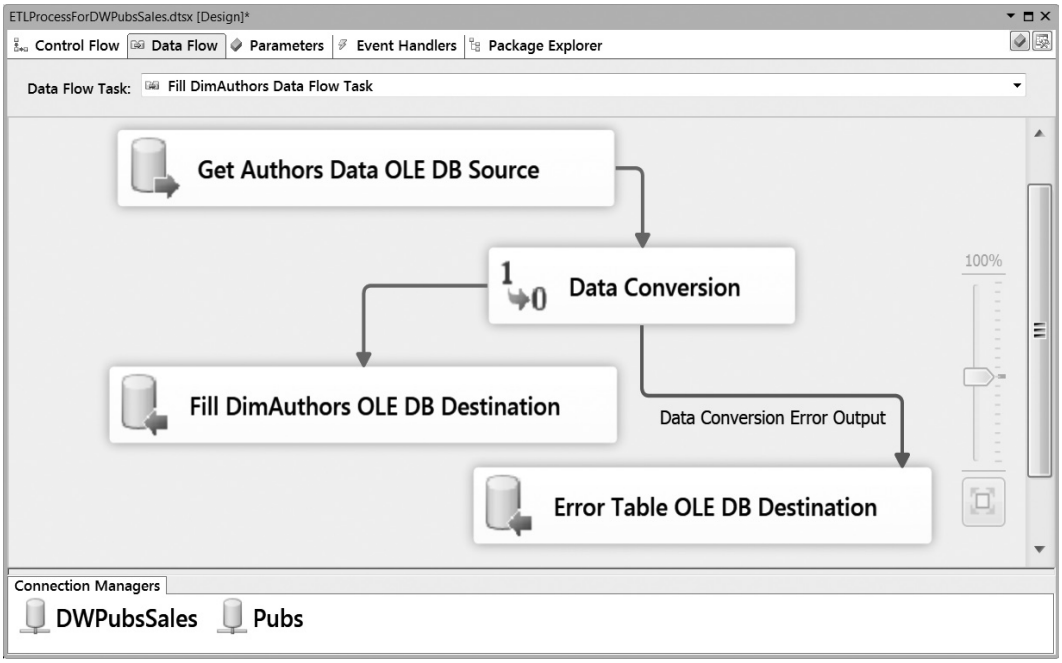


Figure 8-12. *Adding an error output path to the data flow*

Connect the error flow path from a transformation task to a destination component by dragging and dropping the red arrow just as you would when working with a data flow path output. Once you connect the arrow, the Configure Error Output window automatically appears (Figure 8-13).

In the Configure Error Output window (Figure 8-13), you can configure SSIS to redirect the row causing the error, ignore the error, or fail the source component. The default setting is Fail, but leaving it set to default defeats the purpose of the error output, because no rows would be redirected to the log file or table.

In Figure 8-13, an arrow points to the Selected Cells dropdown box at the bottom of the dialog window. Its purpose is to allow you to highlight multiple cells within the dialog window and set all of their values at once. This is an optional feature, and it can be safely ignored until you have many outputs that you want to configure collectively. (The only reason to point out this option is because it is so prominent on the dialog window that it seems more significant than it is.)

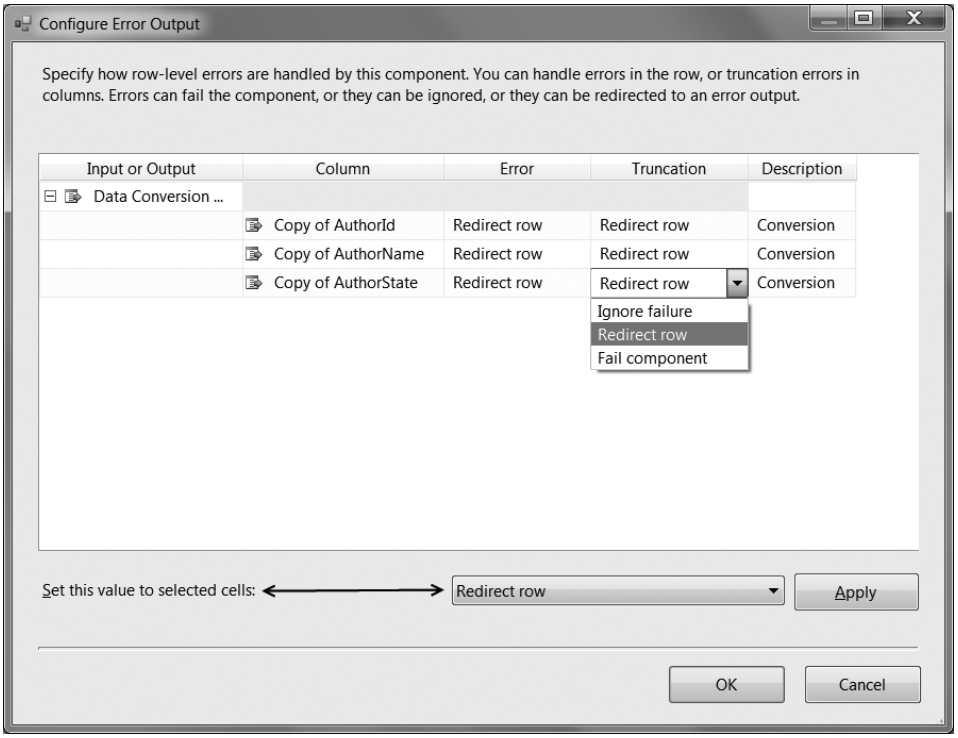


Figure 8-13. *Configuring the data flow error output path*

■ **Note** Error outputs are a common feature in production ETL processing, but we keep things simple by not using error outputs in our exercises.

Configuring the Data Destination

Data destinations insert data into database tables, files, spreadsheets, and so on. The OLE DB Destination Editor has three pages: Connection Manager, Mappings, and Error Output. You can see these listed on the left of Figure 8-14.

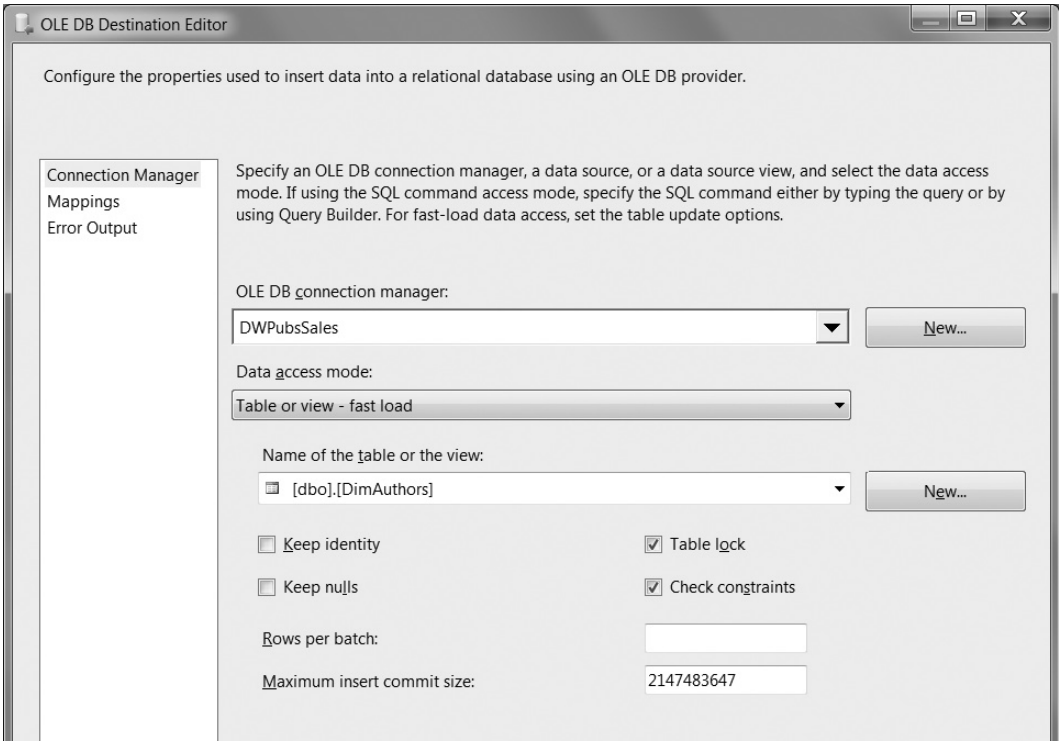


Figure 8-14. *Configuring an OLE DB data destination*

Each data destination component can look slightly different depending on which type of destination component you use. The SSIS Toolbox has several data destination components to choose from, but you are likely to use the OLE DB destination the majority of the time. Therefore, we focus on the settings of this type of data destination.

■ **Important** If you are using an OLE DB source task, you should choose the OLE DB destination versus the SQL Server destination, even when you are importing data into a SQL Server database. Although the SQL Server destination works, it may expect different metadata types than the OLE DB destination provides. If you try to use an OLE DB data source with a SQL Server destination, you can anticipate errors unless you have transformed all the data types appropriately.

The Connection Manager Page

On the Connection Manager page, set the connection to point to your destination table and decide what access mode you are going to use (Figure 8-15).

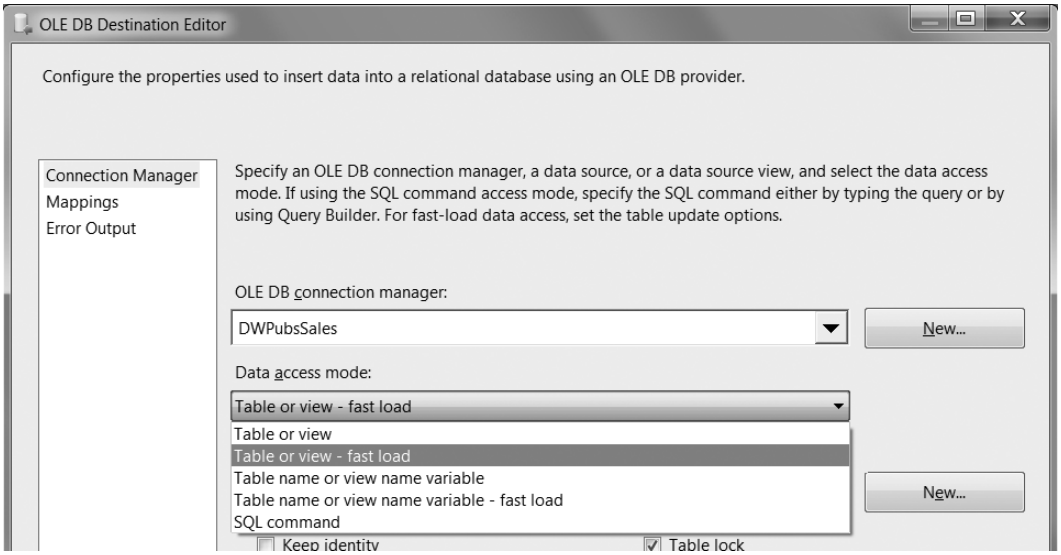


Figure 8-15. *Selecting a data access mode*

To select your data connection, use the "OLE DB connection manager" dropdown box, shown in Figure 8-15, and select the OLE DB connection object that has already been created in the SSIS package. If, for some reason, you have not created an OLE DB connection before you get to this point, clicking the New button allows you to create one.

After you select your connection, configure the data access mode you want by using the "Data access mode" dropdown box (Figure 8-15). An OLE DB destination has five data access modes to choose from:

- *Table or view*: Insert values into a new or existing table or view.
- *Table or view - fast load*: Bulk insert into a new or existing table or view.
- *Table name or view name variable*: Same as the "Table or view" option, but you pass the table or view name through an SSIS variable
- *Table name or view name variable - fast load*: Same as the "Table or view - fast load" option, but you pass the table or view name through a SSIS variable.
- *SQL command*: You use a SQL query to describe the table and column names.

You will use the "Table or view - fast load" option for most occasions, because it provides additional configuration options as well as it is easy to use.

■ **Tip** Fast load uses SQL Server bulk insert statements. The other option uses regular SQL Server insert statements. Since SQL Server 2005 and later, BULK INSERT enforces new, stricter data validation. The two most common issues occur when Unicode text data is not used or when the decimal data type is not used for floating-point numbers. You can find more information in SQL Books Online, but since we converted all of our examples to Unicode and decimal data types, this should not be an issue for our exercises.

Mappings Page

The Mappings page allows you to dictate which input columns are mapped to which destination output columns. SSIS automatically maps the input and destinations for you if the names of both source and destination columns are the same. If you use column aliases SQL code, as we did, the column names will be the same. If, on the other hand, the column names are not the same, you have to map them manually.

Mapping the input and destination columns is an easy process. Just drag and drop the available input columns in the table to destination columns. Another option is to use the dropdown box to select an input column and an associated destination column (Figure 8-16).

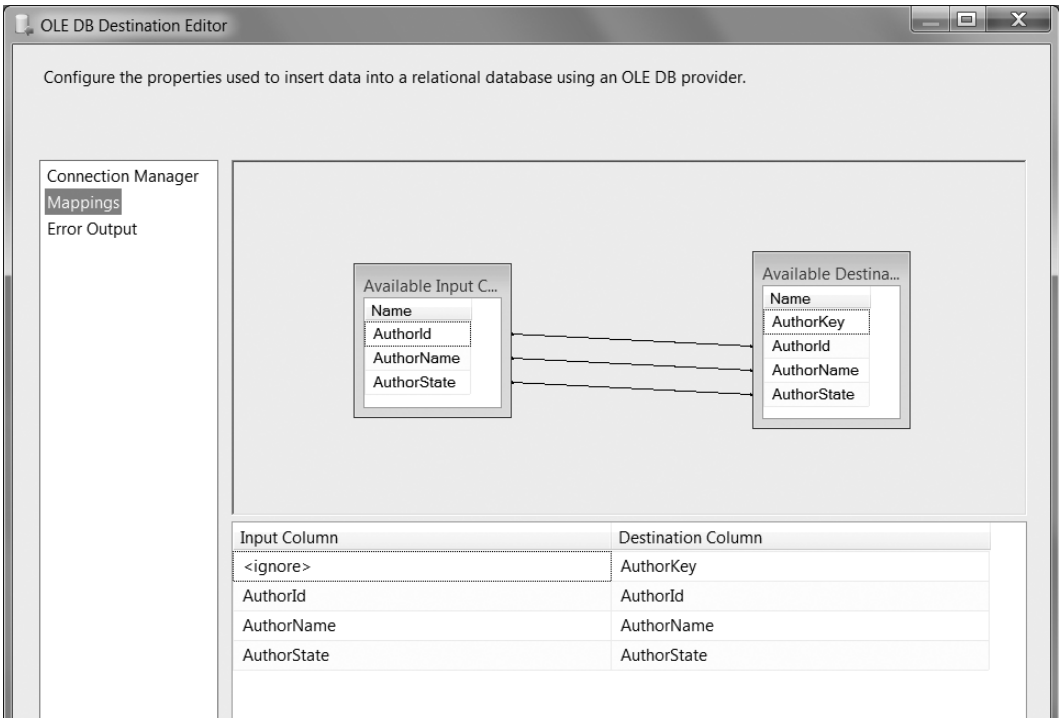


Figure 8-16. *The Mappings page*

You do not have to map input columns to all destination columns. If you have more input columns than you have destination columns, set the input column to <ignore>, thus excluding columns from the output.

■ **Tip** Sometimes the XML code in the SSIS package will not be properly written unless you click the Mappings page. Because of this, we recommend you always click the Mappings page even if you are not going to make any additional configurations.

Error Output Page

Just like the data sources and transformation tasks, the Error Output page allows you to redirect data that causes an error to a file or SQL table. This can be convenient in some cases, but typically you want your errors to be taken care of before you get to this stage of the data flow.

■ **Note** The destination components were not specifically set up to handle errors but rather to insert clean data into a destination. All duplications, conversions, foreign key values, and so on, need to be configured before you get to the destination!

Now that you have seen how to configure a data flow, it is time to put that knowledge into practice. Let's do so in the next exercise.

EXERCISE 8-1. COMPLETING THE SSIS PACKAGE

In this exercise, you complete the SSIS package by configuring all of the remaining tasks and then testing them to verify that the entire SSIS package can execute as a whole. Use the same code file from Exercise 7-3 of Chapter 7 for all of SQL code required in this exercise.

Tip: If you have any problems with this exercise, don't worry, we have you covered. You will still be able to complete the exercises in the rest of the book using a prefilled version of the DWPubsSales database. You will find this file in the `C:\_BookFfiles\Chapter07Files\FilledDWPubsSalesDB folder.`

Reset the Database Objects Before Testing Your ETL Process

Before we begin configuring and testing our tasks, let's make sure that the database is in its normal empty state by running the SQL code that re-creates the database.

1. Locate the Create DWPubsSales Script.sql file in Visual Studio or SQL Server Management Studio (Figure 8-17). You should find this file under the DWPubsSales solution folder you created in Exercise 5-4.

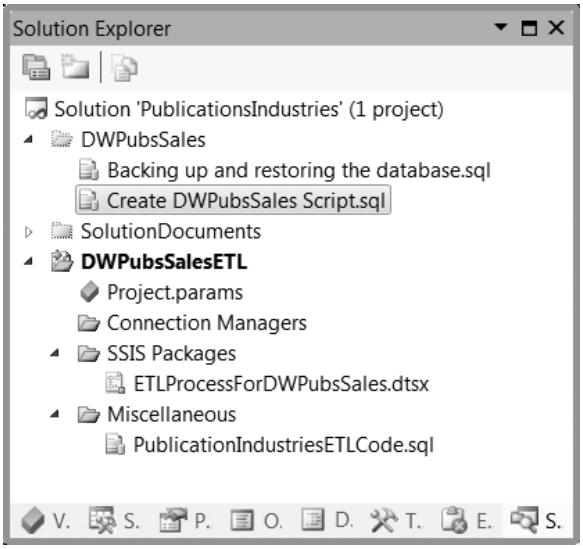


Figure 8-17. *Locating the create DWPubsSales script*

2. Open this file and execute its code in either Visual Studio or SQL Server Management Studio to re-create the database.

You can do this by right-clicking an area in the SQL code window to bring up the context menu and select Execute SQL. If the Connect to Database Engine dialog box appears, type in your server name and click the Connect button. In a few seconds, you should receive a message stating "The DWPubsSales data warehouse is now created."

Fill DimAuthors Data Flow Task

Now that you located the ETL code, you need to use it in the task that fills DimAuthors. This is the same process we used in Chapter 7.

1. Open the PublicationIndustriesETLCode.sql SQL script file in Visual Studio or SQL Server Management Studio (Figure 8-17) and locate the code that is commented as `-- Step 2 Code used to fill tables`.
2. Navigate to the Data Flow tab and select Fill DimAuthors Data Flow Task from the Data Flow Task dropdown box.
3. Add an OLE DB Data Source to the data flow surface. Rename it to Get Authors Data OLE DB Source.
4. Locate the comment `-- Step 2a) Get source data from pubs.dbo.` authors in the SQL code file and review the code beneath it. The code should look like Listing 8-1.

Listing 8-1. SQL Code from pubs.dbo.authors

```
Select
  [AuthorId]=Cast( au_id as nChar(11) )
```

```
, [AuthorName]=Cast( (au_fname+' '+au_lname) as nVarchar(100) )
, [AuthorState]=Cast( state as nChar(2) )
From pubs.dbo.authors
```

5. Add this code to the SQL command text code area of the OLE DB data source as shown in Figure 8-8.
6. Click the Preview button to verify that the query worked successfully.
7. Click the Columns page to force the XML to be written properly in the .dtsx file.
8. Close the OLE DB Source Editor window by clicking the OK button.
9. Add an OLE DB Data Destination to the data flow. Rename it to Fill DimAuthors OLE DB Destination.
10. Connect the data flow path from the source to the destination.
11. Edit the OLE DB data destination as shown in Figure 8-14.
12. Click the Mappings page to force the XML to be written properly in the .dtsx file and verify that the mappings look like Figure 8-16.
13. Verify that the data flow looks like Figure 8-18.

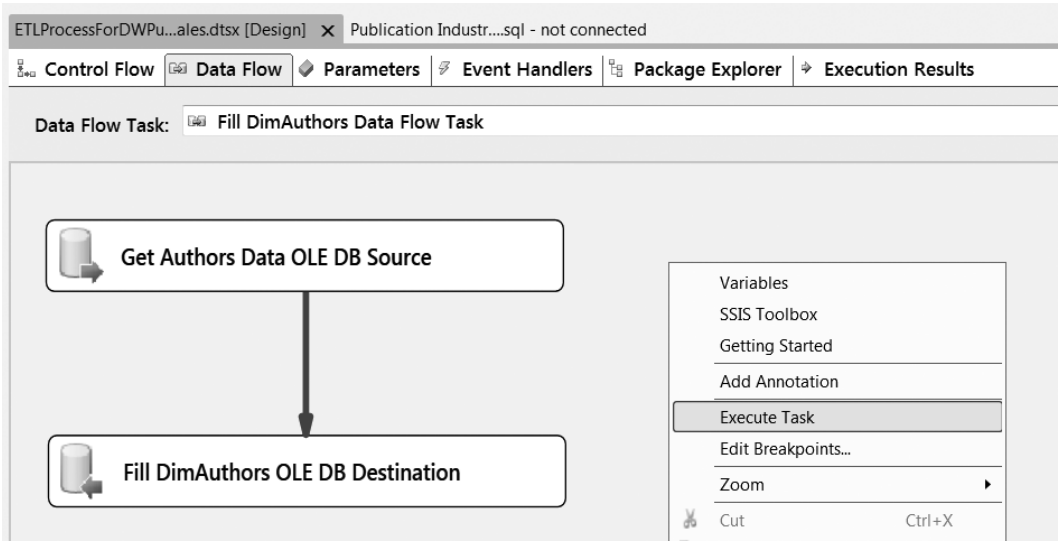


Figure 8-18. *Testing the Fill DimAuthors data flow task*

Important: You do not need to configure anything more than the connection and the table name. Leave all the checkboxes on their default settings as shown. Nor should you need to do anything on the mappings page since our code uses column aliases that match the column names in the data warehouse tables. Just verify that the mappings are correct and move on. In fact, we will save some trees by not showing screenshots for the other tables, since the configurations are all the same (with the exception of the table and column names, of course).

14. Right click the data flow design surface and choose Execute Task from the context menu. Visual Studio launches this data flow task with its debugging engine (Figure 8-18).
15. When the task is complete, stop Visual Studio's debugger and verify that the task ran successfully; or, if there are errors, troubleshoot and try again. You can stop the debugger using the Debug menu or by clicking the link at the bottom of the screen.
16. Open SQL Server Management Studio and verify that DimAuthors has data in it.

Tip: Common errors include primary and foreign key violations. If those happen to you, try running the Execute SQL tasks that you created in Exercise 7-3 to clear the table data and drop the foreign key constraints. You can run both by executing the Prepare ETL Process sequence container.

Fill DimStores Data Flow Task

Congratulations, DimAuthors has data! Let's do the same for DimStores.

1. Navigate to the Data Flow tab and select the Fill DimStores Data Flow Task from the Data Flow Task dropdown box.
2. Add an OLE DB Data Source to the data flow surface. Rename it to Get Stores Data OLE DB Destination.
3. Locate the comment `-- 2b) Get source data from pubs.dbo.stores` in the SQL code file and review the code beneath it. The code should look like Listing 8-2.

   Listing 8-2. Source Data from pubs.dbo.stores

   ```
   Select
     [StoreId]=Cast( stor_id as nChar(4) )
   , [StoreName]=Cast( stor_name as nVarchar(50) )
   From pubs.dbo.stores
   ```

4. Add this code to the SQL command text code area of the OLE DB data source (similar to Figure 8-8).
5. Click the Preview button to verify that the query worked successfully.
6. Click the Columns page to force the XML to be written properly in the .dtsx file.
7. Close the OLE DB Source Editor window by clicking the OK button.
8. Add an OLE DB Data Destination to the data flow surface. Rename it to Fill DimStores OLE DB Destination.
9. Connect the data flow path from the source to the destination.
10. Edit the OLE DB data destination so that it imports data to the DimStores table (similar to Figure 8-14).

11. Click the Mappings page to force the XML to be written properly in the .dtsx file and verify that the mappings look logically correct.
12. Right click the data flow design surface and choose Execute Task from the context menu (similar to Figure 8-18).
13. When the task is complete, stop Visual Studio's debugger and verify that the task ran successfully or troubleshoot any errors and try again.
14. Open SQL Server Management Studio and verify that DimStores has data in it.

Configure the Fill DimPublishers Data Flow Task

DimPublishers is next on the list. Let's tackle it now.

1. Navigate to the Data Flow tab and select the Fill DimPublishers Data Flow Task from the Data Flow Task dropdown box.
2. Add an OLE DB Data Source to the data flow surface. Rename it to Get Publishers Data OLE DB Source.
3. Locate the comment `-- 2c) Get source data from pubs.dbo.publishers` in the SQL code file and review the code beneath it. The code should look like Listing 8-3.

   Listing 8-3. SQL Code from pubs.dbo.publishers

   ```
   Select
     [PublisherId]=Cast( pub_id as nChar(4) )
   , [PublisherName]=Cast( pub_name as nVarchar(50) )
   From pubs.dbo.publishers
   ```

4. Add this code to the SQL command text code area of the OLE DB data source (similar to Figure 8-8).
5. Click the Preview button to verify that the query worked successfully.
6. Click the Columns page to force the XML to be written properly in the .dtsx file.
7. Close the OLE DB Source Editor window by clicking the OK button.
8. Add an OLE DB Data Destination to the data flow surface. Rename it to Fill DimPublishers OLE DB Destination.
9. Connect the data flow path from the data source to the data destination.
10. Edit the OLE DB data destination so that it imports data to the DimPublishers table (similar to Figure 8-14)
11. Click the Mappings page to force the XML to be written properly in the .dtsx file and verify that the mappings look logically correct.
12. Right click the data flow design surface and choose Execute Task from the context menu (similar to Figure 8-18).

13. When the task is complete, stop Visual Studio's debugger and verify that the task ran successfully or troubleshoot any errors and try again.

14. Open SQL Server Management Studio and verify that DimPublishers has data in it.

Configure the Fill DimDates Execute SQL Task

We should to fill up the DimDates table before we fill the DimTitles table. This is because of the foreign key relationship between the two tables. This is not strictly necessary since we have dropped the foreign key constraints in the database, but filling the tables in the same order as you would as if the foreign key constraints were still there documents the logical progression of the ETL process.

1. Locate the comment `-- 2d) Create values for DimDates as needed` in the SQL code file and review the code beneath it. The code should look like Listing 8-4.

   Listing 8-4. Code That Fills the DimDates Table

```
-- 2d) Create values for DimDates as needed.

-- Create variables to hold the start and end date
Declare @StartDate datetime = '01/01/1990'
Declare @EndDate datetime = '01/01/1995'

-- Use a while loop to add dates to the table
Declare @DateInProcess datetime
Set @DateInProcess = @StartDate

While @DateInProcess <= @EndDate
 Begin
 -- Add a row into the date dimension table for this date
 Insert Into DimDates
 ( [Date], [DateName], [Month], [MonthName], [Quarter], [QuarterName], [Year],
 [YearName] )
 Values (
 @DateInProcess -- [Date]
 , DateName( weekday, @DateInProcess ) -- [DateName]
 , Month( @DateInProcess ) -- [Month]
 , DateName( month, @DateInProcess ) -- [MonthName]
 , DateName( quarter, @DateInProcess ) -- [Quarter]
 , 'Q'+DateName( quarter, @DateInProcess )+' - '+Cast( Year(@DateInProcess) as
 nVarchar(50) ) -- [QuarterName]
 , Year( @DateInProcess )
 , Cast( Year(@DateInProcess ) as nVarchar(50) ) -- [YearName]
 )
-- Add a day and loop again
Set @DateInProcess = DateAdd(d, 1, @DateInProcess)
End
```

2. Highlight this code, right-click it, and choose Copy from the context menu.

3. Navigate to the Control Flow tab and select the Fill DimDates Execute SQL Task.

4. Right-click this task and select Edit from the context menu.

5. Edit the Fill DimDates Execute SQL Task by right-clicking the task and selecting Edit from the context menu.
6. Select the Connection property. A dropdown box appears in the dialog window that will let you configure this property.
7. Configure the Connection property by selecting the option DWPubsSales from the dropdown box, as shown in Figure 8-19.

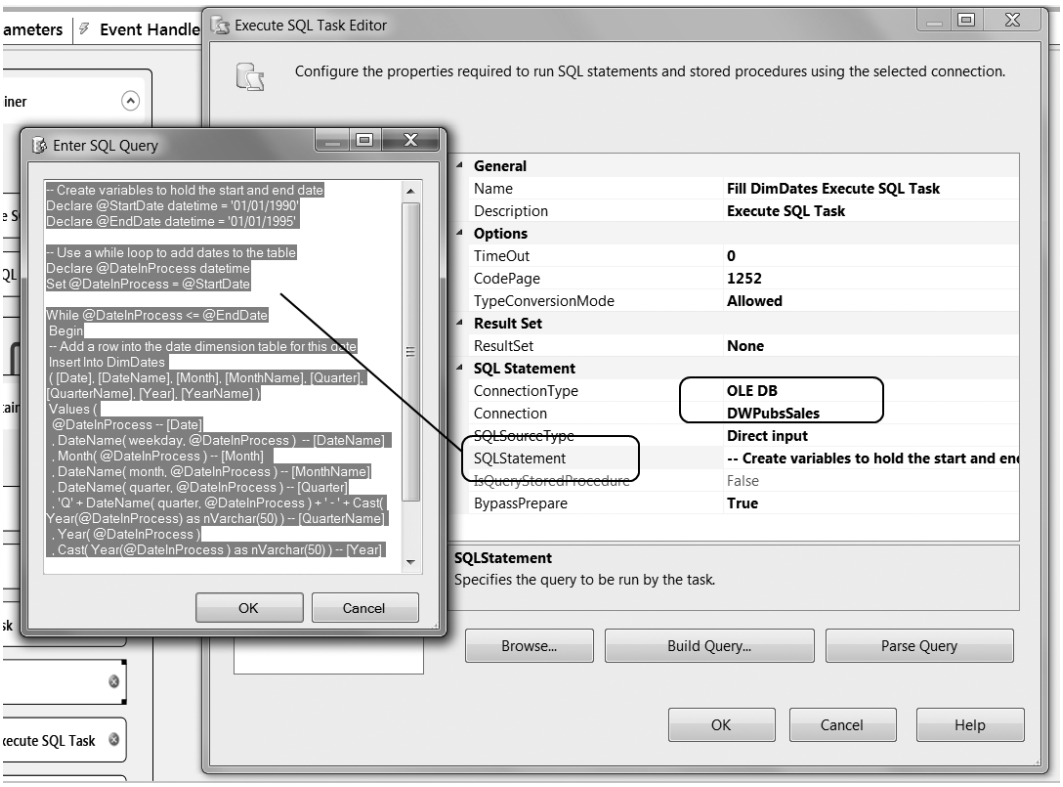


Figure 8-19. *Configuring the Fill DimDates Execute SQL Task*

8. Select the SQL Statement property. An ellipsis button appears in the dialog window that will let you configure this property.
9. Configure the SQL Statement property by clicking the ellipsis button and pasting the code from Listing 8-4 (that you copied earlier) into the Enter SQL Query dialog window that appears (Figure 8-19).
10. Remove any SQL Go statements from your code. SQL Go statements will cause the Execute SQL Task to fail.
11. Click the OK button to close the Enter SQL Query dialog window.
12. Click the OK button to close the Execute SQL Task editor.

13. Right-click the Fill DimDates Execute SQL Task and choose Execute Task from the context menu (Figure 8-20).

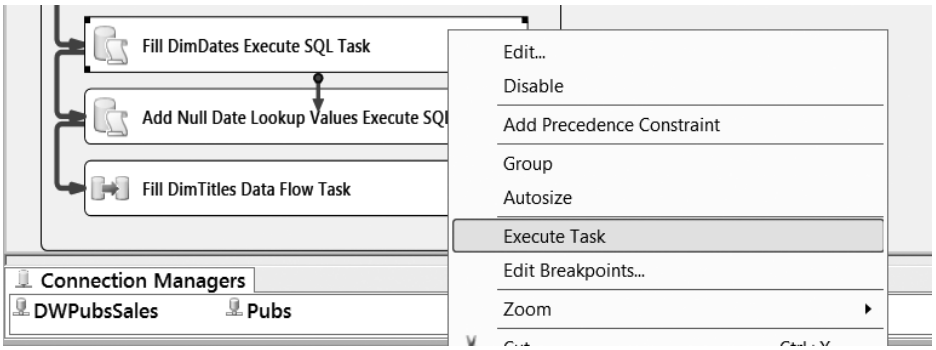


Figure 8-20. *Executing the Fill DimDate Execute SQL Task*

14. When the task is complete, stop Visual Studio's debugger and verify that the task ran successfully; if there are errors, troubleshoot and try again.

15. Open SQL Server Management Studio and verify that DimDates has data in it.

Configure the Add Null Date Lookup Values Execute SQL Task

Now we need to add a couple of dates used to for null value lookups to the DimDates table. This was an option we discussed in Chapter 6 but have waited until now to implement it.

1. Locate the comment `-- 2e) Add additional lookup values to DimDates` in the SQL code file and review the code beneath it. The code should look like Listing 8-5.

Listing 8-5. Code That Adds New Dates to the DimDates Table

```
Set Identity_Insert [DWPubsSales].[dbo].[DimDates] On
Insert Into [DWPubsSales].[dbo].[DimDates]
 ( [DateKey]
 , [Date]
 , [DateName]
 , [Month]
 , [MonthName]
 , [Quarter]
 , [QuarterName]
 , [Year], [YearName] )
 Select
 [DateKey]=-1
 , [Date]=Cast('01/01/1900' as nVarchar(50) )
 , [DateName]=Cast('Unknown Day' as nVarchar(50) )
 , [Month]=-1
 , [MonthName]=Cast('Unknown Month' as nVarchar(50) )
 , [Quarter]=-1
 , [QuarterName]=Cast('Unknown Quarter' as nVarchar(50) )
 , [Year]=-1
```

```
, [YearName]=Cast('Unknown Year' as nVarchar(50) )
Union
Select
[DateKey]=-2
, [Date]=Cast('01/01/1900' as nVarchar(50) )
, [DateName]=Cast('Corrupt Day' as nVarchar(50) )
, [Month]=-2
, [MonthName]=Cast('Corrupt Month' as nVarchar(50) )
, [Quarter]=-2
, [QuarterName]=Cast('Corrupt Quarter' as nVarchar(50) )
, [Year]=-2
, [YearName]=Cast('Corrupt Year' as nVarchar(50) )
Set Identity_Insert [DWPubsSales].[dbo].[DimDates] Off
```

2. Highlight this code, right-click it, and choose Copy from the context menu.
3. Navigate to the Control Flow tab and select the Add Null Date Lookup Values Execute SQL Task.
4. Edit the Add Null Date Lookup Values Execute SQL Task by right-clicking the task and selecting Edit from the context menu.
5. Select the Connection property. A dropdown box appears in the dialog window that will let you configure this property.
6. Configure the Connection property by selecting the option DWPubsSales from the dropdown box (Figure 8-21).

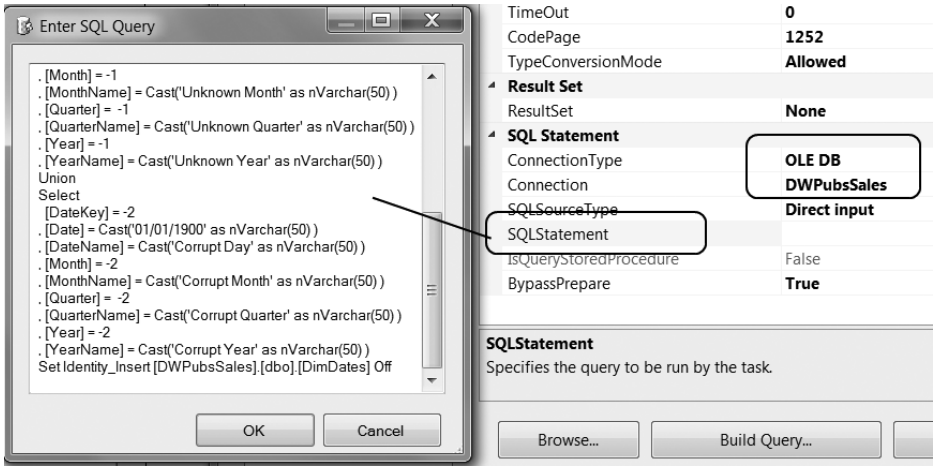
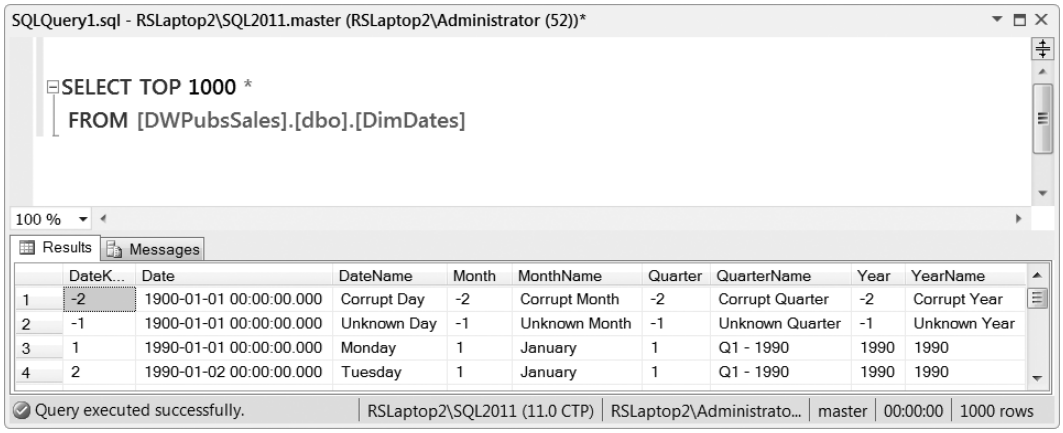


Figure 8-21. *Configuring the Add Null Dates Lookup Values Execute SQL Task*

7. Select the SQL Statement property. An ellipsis button appears in the dialog window that will let you configure this property.
8. Configure the SQL Statement property by clicking the ellipsis button and pasting the code from Listing 8-5 that you copied earlier into the Enter SQL Query dialog window that appears (Figure 8-21).

9. Verify that there are no SQL Go statements your code. SQL Go statements will cause the Execute SQL Task to fail.
10. Click the OK button to close the Enter SQL Query dialog window.
11. Click the OK button to close the Execute SQL Task editor.
12. Right click then Add Null Date Lookup Values Execute SQL Task and choose Execute Task from the context menu (similar to Figure 8-20). Visual Studio will launch this Execute SQL task with its debugging engine.
13. When the task is complete, stop Visual Studio's debugger and verify that the task ran successfully; if there are errors, troubleshoot and try again.
14. Open SQL Server Management Studio and verify that DimDates has data in it. Figure 8-22 displays what the data should look like.



	DateK...	Date	DateName	Month	MonthName	Quarter	QuarterName	Year	YearName
1	-2	1900-01-01 00:00:00.000	Corrupt Day	-2	Corrupt Month	-2	Corrupt Quarter	-2	Corrupt Year
2	-1	1900-01-01 00:00:00.000	Unknown Day	-1	Unknown Month	-1	Unknown Quarter	-1	Unknown Year
3	1	1990-01-01 00:00:00.000	Monday	1	January	1	Q1 - 1990	1990	1990
4	2	1990-01-02 00:00:00.000	Tuesday	1	January	1	Q1 - 1990	1990	1990

Figure 8-22. *Verifing the DimDates data*

Configure the Fill DimTitles Data Flow Task

The last dimension table we need to fill is DimTitles. So…

1. Navigate to the Data Flow tab and select the Fill DimTitles Data Flow Task from the Data Flow Task dropdown box.
2. Add an OLE DB Data Source to the data flow surface. Rename it to Get Titles Data OLE DB Destination.
3. Locate the comment `-- 2e) Get source data from pubs.dbo.titles` in the SQL code file and review the code beneath it. The code should look like Listing 8-6.

   Listing 8-6. SQL Code for pubs.dbo.titles

```
Select
   [TitleId]=Cast( isNull( [title_id], -1 ) as nvarchar(6) )
 , [TitleName]=Cast( isNull( [title], 'Unknown' ) as nvarchar(50) )
 , [TitleType]=Cast( isNull( [type], 'Unknown' ) as nvarchar(50) )
```

```
  , [PublisherKey]=[DWPubsSales].[dbo].[DimPublishers].[PublisherKey]
  , [TitlePrice]=Cast( isNull( [price], -1 ) as decimal(18, 4) )
  , [PublishedDateKey]=isNull( [DWPubsSales].[dbo].[DimDates].[DateKey], -1)
From [Pubs].[dbo].[Titles]
Join [DWPubsSales].[dbo].[DimPublishers]
 On [Pubs].[dbo].[Titles].[pub_id]=[DWPubsSales].[dbo].[DimPublishers].
 [PublisherId]
Left Join [DWPubsSales].[dbo].[DimDates] -- The "Left" keeps dates not found in
DimDates
 On [Pubs].[dbo].[Titles].[pubdate]=[DWPubsSales].[dbo].[DimDates].[Date]
```

4. Edit the OLE DB Data Source as shown in Figure 8-23. (We are using the DWPubsSales connection, but reference the original Pubs.dbo.titles table in our code.)

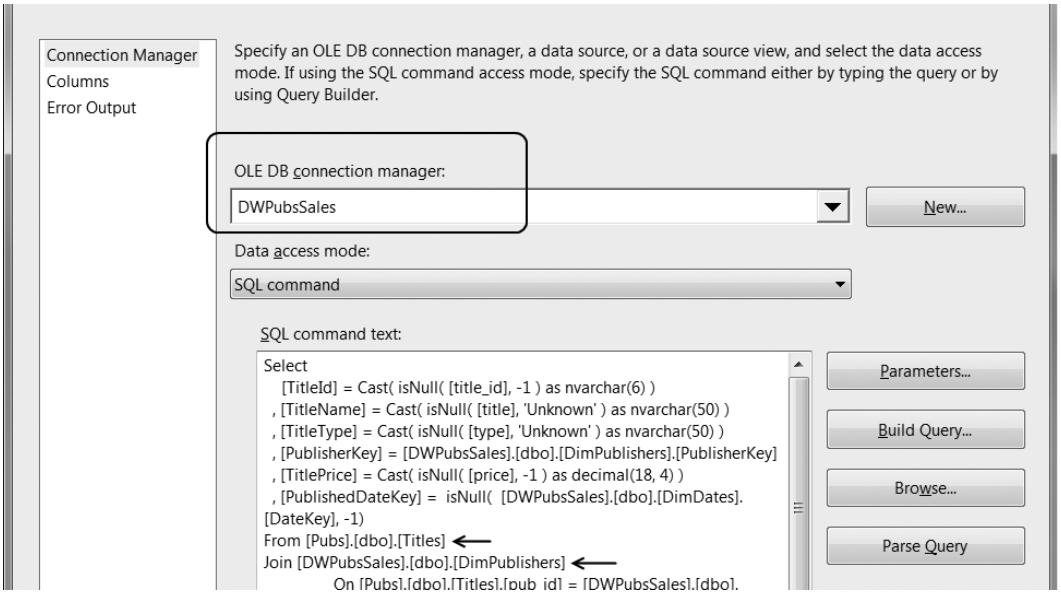


Figure 8-23. *The OLE DB source for the Fill DimTitles Data Flow Task*

5. Click the Preview button to verify that the query works successfully.
6. Click the Columns page to force the XML to be written properly in the .dtsx file.
7. Close the OLE DB Source Editor window by clicking the OK button.
8. Add an OLE DB Data Destination to the Data Flow surface. Rename it to Fill DimTitles OLE DB Destination.
9. Connect the data flow path from the source to the destination.
10. Edit the OLE DB data destination so that it imports data to the DimTitles table (similar to Figure 8-14)
11. Click the Mappings page to force the XML to be written properly in the .dtsx file and verify that the mappings look logically correct.

12. Right click the Data Flow design surface and choose Execute Task from the context menu. Visual Studio will launch this data flow task with its debugging engine.
13. When the task is complete, stop Visual Studio's debugger and verify that the task ran successfully; if there are errors, troubleshoot and try again.
14. Open SQL Server Management Studio and verify that DimTitles has data in it.

Configure the Fill FactTitlesAuthors Data Flow Task

With all the dimension tables filled, it is time to fill the fact tables. It does not matter which one you begin with, but let's start with the FactTitlesAuthors table.

1. Navigate to the Data Flow tab and select the Fill FactTitlesAuthors Data Flow Task from the Data Flow Task dropdown box.
2. Add an OLE DB Data Source to the data flow surface. Rename it to Get TitleAuthors Data OLE DB Destination.
3. Locate the comment `-- 2f) Get source data from pubs.dbo.titleauthor` in the SQL code file and review the code beneath it. The code should look like Listing 8-7.

   Listing 8-7. SQL Code from pubs.dbo.titleauthor

   ```
   Select
    [TitleKey]=DimTitles.TitleKey
   --, title_id
   ,[AuthorKey]=DimAuthors.AuthorKey
   --, au_id
   ,[AuthorOrder]=au_ord
   From pubs.dbo.titleauthor
   JOIN DWPubsSales.dbo.DimTitles
    On pubs.dbo.titleauthor.Title_id=DWPubsSales.dbo.DimTitles.TitleId
   JOIN DWPubsSales.dbo.DimAuthors
    On pubs.dbo.titleauthor.Au_id=DWPubsSales.dbo.DimAuthors.AuthorId
   ```

4. Edit the OLE DB data source to use the DWPubsSales connection (similar to Figure 8-23).
5. Click the Preview button to verify that the query worked successfully.
6. Click the Columns page to force the XML to be written properly in the .dtsx file.
7. Close the OLE DB Source Editor window by clicking the OK button.
8. Add an OLE DB Data Destination to the Data Flow surface. Rename it to Fill FactTitlesAuthors OLE DB Destination.
9. Connect the Data Flow Path from the source to the destination.
10. Edit the OLE DB data destination so that it imports data to the FactTitlesAuthors table (similar to Figure 8-14).
11. Click the Mappings page to force the XML to be written properly in the .dtsx file and verify that the mappings look logically correct.

12. Right click the data flow design surface and choose Execute Task from the context menu. Visual Studio will launch this data flow task with its debugging engine.
13. When the task is complete, stop Visual Studio's debugger and verify that the task ran successfully; if there are errors, troubleshoot and try again.
14. Open SQL Server Management Studio and verify that DimFactTitlesAuthors has data in it.

Configure the Fill FactSales Data Flow Task

Our last table to fill is the FactSales table.

1. Navigate to the Data Flow tab and select the Fill FactSales Data Flow Task from the Data Flow Task dropdown box.
2. Add an OLE DB Data Source to the Data Flow surface. Rename it to Get Sales Data OLE DB Destination.
3. Locate the comment `-- 2g) Get source data from pubs.dbo.sales` in the SQL code file and review the code beneath it. The code should look like Listing 8-8.

   Listing 8-8. SQL Code from pubs.dbo.sales

   ```
   Select
    [OrderNumber]=Cast(ord_num as nVarchar(50))
   ,[OrderDateKey]=DateKey
   --, title_id
   ,[TitleKey]=DimTitles.TitleKey
   --, stor_id
   ,[StoreKey]=DimStores.StoreKey
   ,[SalesQuantity]=qty
   From pubs.dbo.sales
   JOIN DWPubsSales.dbo.DimDates
    On pubs.dbo.sales.ord_date=DWPubsSales.dbo.DimDates.date
   JOIN DWPubsSales.dbo.DimTitles
    On pubs.dbo.sales.Title_id=DWPubsSales.dbo.DimTitles.TitleId
   JOIN DWPubsSales.dbo.DimStores
    On pubs.dbo.sales.Stor_id=DWPubsSales.dbo.DimStores.StoreId
   ```

4. Edit the OLE DB data source to connect to the DWPubsSales connection.
5. Click the Preview button to verify that the query worked successfully.
6. Click the Columns page to force the XML to be written properly in the .dtsx file.
7. Close the OLE DB Source Editor window by clicking the OK button.
8. Add an OLE DB Data Destination to the Data Flow surface. Rename it to Fill FactSales OLE DB Destination.
9. Connect the Data Flow Path from the source to the destination.
10. Edit the OLE DB data destination so that it imports data to the FactSales table (similar to Figure 8-14).

11. Click the Mappings page to force the XML to be written properly in the .dtsx file and verify that the mappings look logically correct.
12. Right-click the data flow design surface and choose Execute Task from the context menu. Visual Studio will launch this data flow task with its debugging engine.
13. When the task complete, stop Visual Studio's debugger and verify that the task ran successfully or troubleshoot any errors and try again.
14. Open SQL Server Management Studio and verify that FactSales has data in it.

Configure the Add Foreign Key Constraints Execute SQL Task

One last task to go! We need to configure an Execute SQL task that replaces the foreign key constraints we dropped that the beginning of the process.

1. Locate the comment `-- Step 3) Add Foreign Keys back` in the SQL code file and review the code beneath it. The code should look like Listing 8-9.

   Listing 8-9. Code That Adds the Foreign Keys Back to the Database

```
-- Step 3) Add Foreign Keys back (Will be used with SSIS Execute SQL Tasks)
Alter Table [dbo].[DimTitles] With Check Add Constraint
[FK_DimTitles_DimPublishers]
Foreign Key ([PublisherKey]) References [dbo].[DimPublishers] ([PublisherKey])

Alter Table [dbo].[FactTitlesAuthors] With Check Add Constraint
[FK_FactTitlesAuthors_DimAuthors]
Foreign Key ([AuthorKey]) References [dbo].[DimAuthors] ([AuthorKey])

Alter Table [dbo].[FactTitlesAuthors] With Check Add Constraint
[FK_FactTitlesAuthors_DimTitles]
Foreign Key ([TitleKey]) References [dbo].[DimTitles] ([TitleKey])

Alter Table [dbo].[FactSales] With Check Add Constraint [FK_FactSales_DimStores]
Foreign Key ([StoreKey]) References [dbo].[DimStores] ([Storekey])

Alter Table [dbo].[FactSales] With Check Add Constraint [FK_FactSales_DimTitles]
Foreign Key ([TitleKey]) References [dbo].[DimTitles] ([TitleKey])

Alter Table [dbo].[FactSales] With Check Add Constraint [FK_FactSales_DimDates]
Foreign Key ([OrderDateKey]) References [dbo].[DimDates] ([DateKey])

Alter Table [dbo].[DimTitles] With Check Add Constraint [FK_DimTitles_DimDates]
Foreign Key ([PublishedDateKey]) References [dbo].[DimDates] ([DateKey])
```

2. Highlight this code, right-click it, and choose Copy from the context menu.
3. Navigate to the Control Flow tab and select the Add Null Date Lookup Values Execute SQL Task.
4. Edit the Add Null Date Lookup Values Execute SQL Task by right-clicking the task and selecting Edit from the context menu.
5. Select the Connection property. A dropdown box appears in the dialog window that will let you configure this property.

6. Configure the Connection property by selecting the option DWPubsSales from the dropdown box (similar to Figure 8-21).
7. Select the SQL Statement property. An ellipsis button appears in the dialog window that will let you configure this property.
8. Configure the SQL Statement property by clicking the ellipsis button and pasting the code from Listing 8-9 (that you copied earlier) into the Enter SQL Query dialog window that appears (Figure 8-21).
9. Verify that there are no SQL Go statements in your code. SQL Go statements will cause the Execute SQL Task to fail.
10. Click the OK button to close the Enter SQL Query dialog window.
11. Click the OK button to close the Execute SQL Task editor window.
12. When the task is complete, stop Visual Studio's debugger and verify that the task ran successfully; if there are errors, troubleshoot and try again.

In this exercise, you configured the SSIS tasks that filled the dimension and fact tables of the DWPubsSales data warehouse. You also tested each tasks as you went. Now we need to test that the entire SSIS package runs successfully as a whole.

Executing the Entire Package

When you have completed configuring and testing all of the tasks in your SSIS package, you need to test the entire package as a whole. This vital step is often missed by developers, and they only find out later that the package does not work as a unit—usually while giving a presentation showing how well it works!

The best way to reset the database back to its normal, preload state, is to use the SQL code script and then right-click the package in Solution Explorer to access the context menu (Figure 8-24). Clicking the Execute Package option will start the package for the beginning of the first task in the chain of precedence constraints.

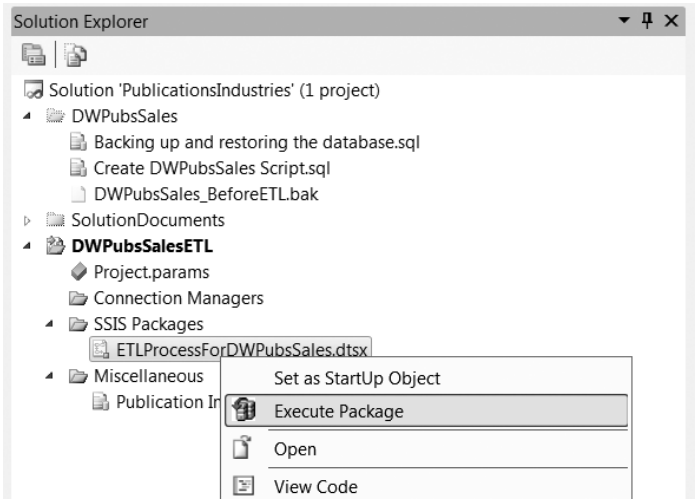


Figure 8-24. *Executing the entire SSIS package*

As each task completes, a green check mark indicates the success of each until all the tasks and sequences containers are done. When the package has completed, it should look like Figure 8-25.

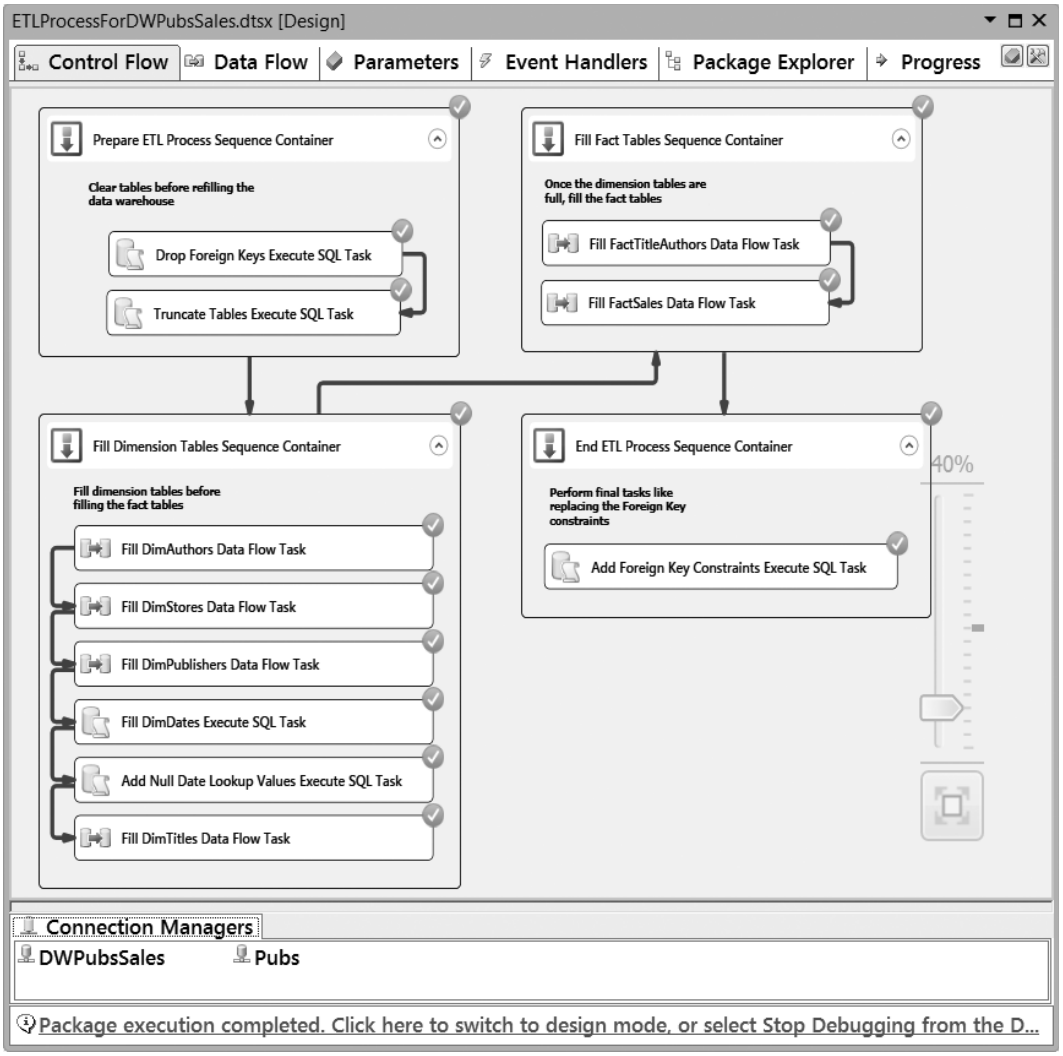


Figure 8-25. *The entire SSIS package succeeded*

Let's finish up this chapter by resetting the database one last time and execute the package in the next exercise.

EXERCISE 8-2. TESTING YOUR SSIS PACKAGE

In this exercise, you will test that the entire package works as expected by resetting the database and executing the package.

Reset the Database Objects Before Testing Your ETL Process

1. Locate the Create DWPubsSales Script.sql file in Visual Studio. You should find this file under the DWPubsSales solution folder you created in Exercise 5-4.
2. Open this file and execute its code. You can do this by right-clicking an empty area in the SQL code window to bring up the context menu and select Execute SQL. You may remember doing this back in Chapter 2 (Figure 2-15).
3. A Connect to Database Engine dialog box will appear. Type in you server name and click the Connect button.
4. In a few seconds, you should receive a message stating "The DWPubsSales data warehouse is now created."

Testing Your SSIS Package

Now it is time to execute the entire SSIS package to verify that it runs as a unit.

1. In Solution Explorer, right-click your SSIS package called ETLProcessForDWPubsSales.dtsx and select Execute Task from the context menu, as shown in Figure 8-24. Visual Studio will launch this Execute SQL task with its debugging engine.
2. When the task is complete, stop Visual Studio's debugger and verify that the task ran successfully or troubleshoot any errors and try again. It should look like Figure 8-25.
3. Open SQL Server Management Studio and verify that all the tables have data in them. You can use the code from Listing 8-10.

   Listing 8-10. SQL Code to Verify DWPubSales' Table Data

   ```
   Select Top 100 * from dbo.DimAuthors
   Select Top 100 * from dbo.DimStores
   Select Top 100 * from dbo.DimPublishers
   Select Top 100 * from dbo.DimDates
   Select Top 100 * from dbo.DimTitles
   Select Top 100 * from dbo.FactSales
   Select Top 100 * from dbo.FactTitlesAuthors
   ```

4. Once you have verified that the data is loaded, create a backup of the database in its filled state using the code in Listing 8-11.

Listing 8-11. Backing Up the Filled DWPubsSales Database

```
Backup Database DWPubsSales
To Disk='C:\_BISolutions\PublicationsIndustries\DWPubsSales\DWPubsSales_AfterETL.bak'
With Init
```

In this exercise, you tested the SSIS package that filled the DWPubsSales database. SSIS can be challenging to master, but with practice you will find it is a superior tool for performing ETL processing.

■ **Note** Remember, if you had any trouble completing this exercise, you can always compare your package to the authors' completed version in the `C:\_BookFiles\Chapter08Files\PublicationsIndustries folder`. Optionally, you can restore the filled database using the SQL database backup file: `C:\_BookFiles\Chapter-08Files\FilledDWPubsSalesDB.bak`. We also included a SQL code file to help you restore the database from the backup file called `C:\_BookFiles\Chapter08Files\RestoreDWPubsSalesDB.sql`.

Moving On

At this point, a BI solution developer has a choice to make: start working on reports or add OLAP cubes to the solution. The decision involves weighing a number of factors, which are best determined by what would be most logical for the particular solution you are working on. As you can see in Figure 8-26, we have chosen Create Cubes as our next step to the process.

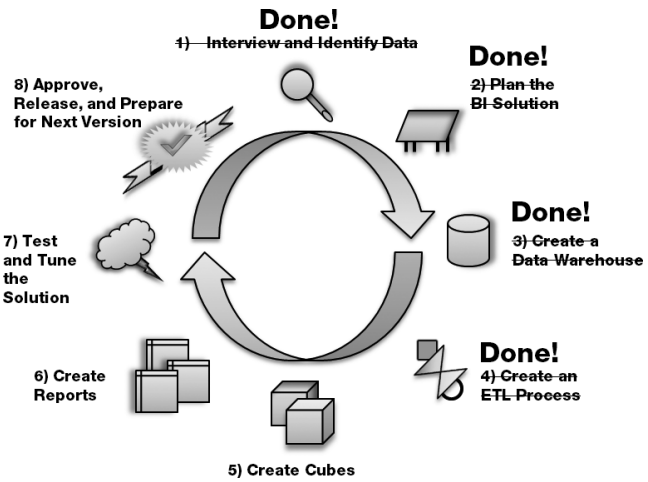


Figure 8-26. *The ETL process is complete*

In Chapter 9, we are going to cover how to create OLAP cubes using Microsoft's SQL Server Analysis Server (SSAS). Once you see what cubes do and how you create them, you will have a much better chance to make the correct decision process for your own BI solution, namely, about whether to include them in your solution or move past them and start working directly on preliminary reports.

LEARN BY DOING

In this "Learn by Doing" exercise, you will create an ETL process similar to the one defined in this chapter. This time you need to use the Northwind database. We have included an outline of the steps you performed in this chapter and an example of how the authors handled them in two Word documents. These documents are found in the folder `C:\_BISolutionsBookFiles\_LearnByDoing\Chapter08Files`. Please see the `ReadMe.doc` for detailed instructions.

What's Next?

We would like to show you more on SSIS logging, events, and configuration, but we will have to stop here and move on to the next process in our BI solution. Meanwhile, the data warehouse is filled with transformed data, and so the ETL process is officially complete. If you would like to learn more about using SSIS for ETL processing, we have articles, videos, and demonstrations on our website at `http://www.NorthwestTech.org/ProBISolutions/ETLProcessing`.