# SSIS Data Flow Training – Notes

# SSIS Data Flow Task

# SSIS Data Conversion Transformation

# SSIS Data Flow Advanced Editor

# SSIS Data Flow Error Output

# SSIS Derived Column Transformation

# SSIS Data Viewer

# Data Flow Path Editor

# SSIS Aggregate Transformation

# SSIS Conditional Split Transformation

# SSIS Multicast Transformation

# SSIS Row Count Transformation

# SSIS Sort Transformation

# SSIS Union All Transformation

# SSIS Merge Transformation

# SSIS Merge Join Transformation

# SSIS Lookup Transformation

# SSIS OLE DB Command Transformation

# SSIS Slowly Changing Dimension Transformation

# Create below tables before start to develop the packages

# CREATE TABLE [books] (

# [ID] [int] identity(1,1) not null,

# [Name] varchar(100) not null,

# [Price] float not null,

# [BoughtDate] datetime null

# )

# CREATE TABLE lendout(

# ID INT IDENTITY(1,1) NOT NULL,

# BookName VARCHAR(100) NOT NULL,

# StudentID INT NOT NULL

# )

# INSERT INTO lendout(BookName, studentID) VALUES('Book A', 1)

# INSERT INTO lendout(BookName, studentID) VALUES('Book d', 2)

# INSERT INTO lendout(BookName, studentID) VALUES('Book H', 3)

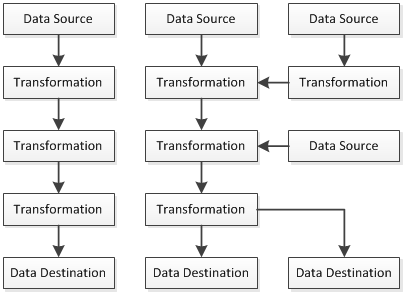
# INSERT INTO lendout(BookName, studentID) VALUES('Book A', 4)

# INSERT INTO lendout(BookName, studentID) VALUES('Book D', 5)

# SSIS Data Flow Task

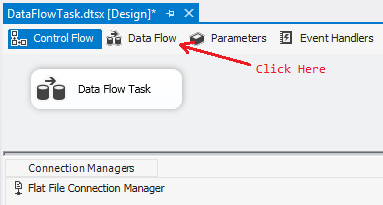
The Data Flow task is the most important and complicated task in a SSIS package. It moves data from sources to destinations and adds transforms between them to update, merge or split data. A Data Flow task can consist of a number of data flows and a data flow can have multiple sources and destinations. If the order of data flows is not significant, Putting the data flows in one task is recommended because at run-time, each data flow can use different CPU of multi-core processors to make the whole process efficiently.

The following is a Data Flow task with multiple data flows. The second data flow contains multiple sources and destinations.

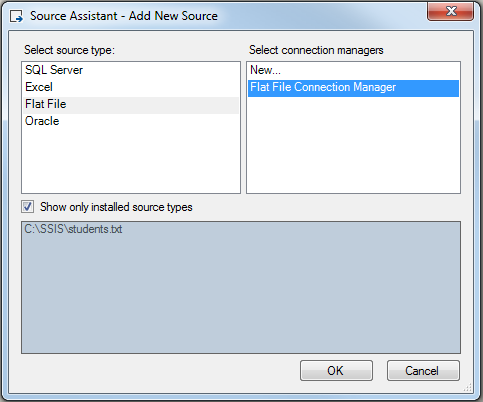


In the following example, I'm going to continue work with the package DataFlowTask.dtsx to create a Data Flow task. we'll use the flat file as a data source and a SQL Server table as the destination.

1. Open the package DataFlowTask.dtsx in the project LearnSSIS2 if it is closed.
2. Drag & drop the Data Flow Task into the Control Flow of the package then click "Data Flow" tab.

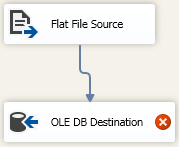


1. Drag and drop Source Assistant into the Data Flow editor to open Source Assistant setting window. Click Flat File and select "Flat File Connection Manager" which was just created. Then click OK button.

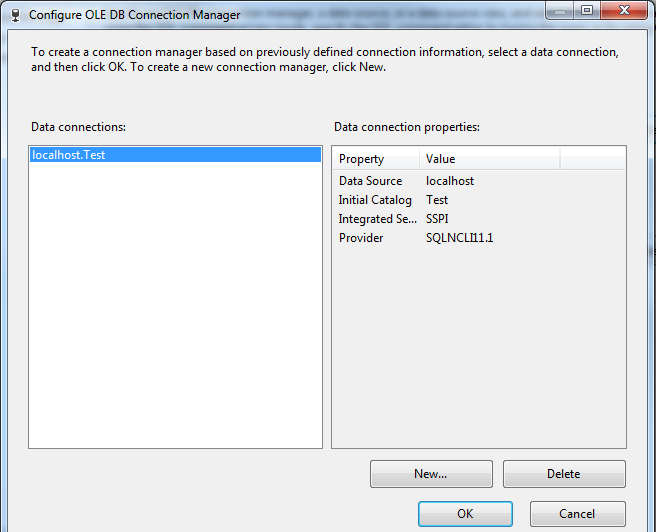


The Flat File Source was created in the Data Flow design panel. Alternatively, you can drag and drop Flat File Source from Other Sources in the Toolbox into design panel directly.

1. Drag and drop OLE DB Destination into Data Flow design panel. Then select Flat File Source and put its dark blue arrow onto OLE DB Destination.

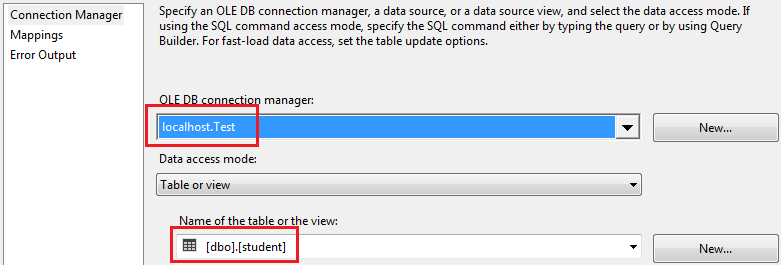


1. Right-click OLE DB Destination and choose "Edit..." to open OLE DB Destination Editor. Then Click "New..." and choose "localhost.Test".

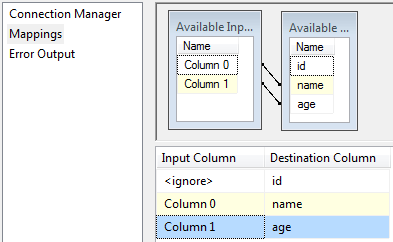


If localhost.Test was not created before, you can click "New..." to create one.

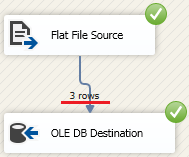
1. Click OK button to close the OLE DB Destination Editor. Then choose student table in the table or the view drop-down list.



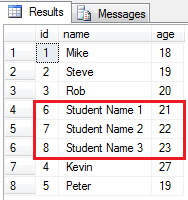
1. Click "Mappings" tab and in the Input Column, choose "Column 0" for name and "Column 1" for age.



1. Click OK button. Then Run the package.



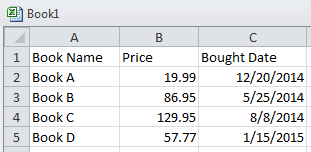
From the result, you can see 3 rows were added. Check the student table in the Test database, you can see the 3 records were inserted.



# SSIS Data Conversion Transformation

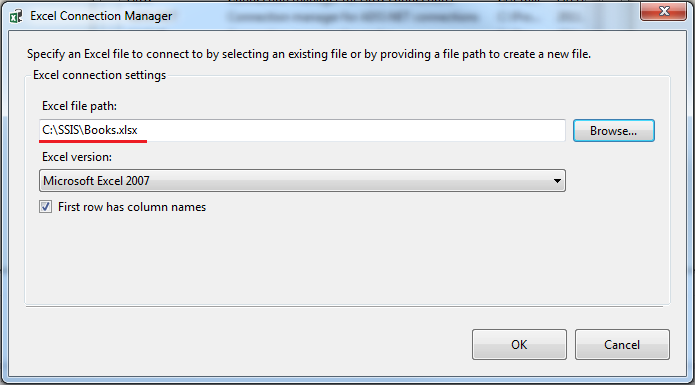
The Data Conversion transformation is like C# type conversions to convert the input data to the output data with different data type. If a data type cannot be converted to another data type, an error will occur. For details about SSIS data types, you can check it here.

In the following example, I'm going to load book information to a new table books in SQL Server Test database. An Excel file books.xlsx was created in C:\SSIS with the content in sheet1 like below.

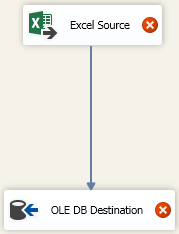


Now follow the steps below to create the sample package.

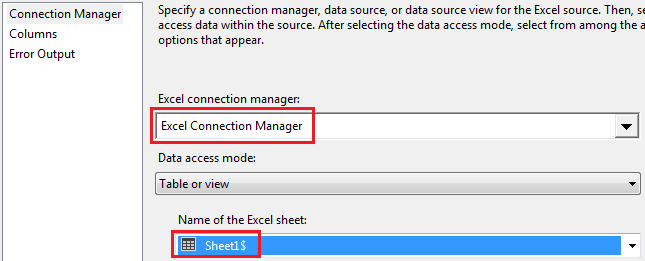
1. Create a new package in the project LearnSSIS2 and rename it to DataConversion.dts.
2. Drag and drop the Data Flow Task into the package and then click Data Flow tab.
3. Create an Excel Connection Manager as the source connection like we did before.



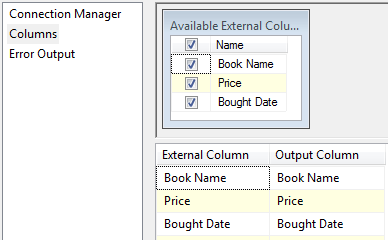
1. Create an OLE DB Connection Manager localhost.Test as the destination connection to link to the SQL Server database Test.
2. Drag and drop an Excel Source and an OLE DB Destination into the Data Flow working area and link them from Excel to OLE DB.



1. Right-click Excel Source and choose "Edit..." to open the Excel Source Editor. Then choose "Sheet1$" as the sheet of the Excel Connection Manager.



1. Click the Columns to check the External Column and Output Column.



1. Click OK button to close the editor. Then open the OLE DB Destination Editor and click "New..." to create the SQL as below.

CREATE TABLE [books] (

[ID] [int] identity(1,1) not null,

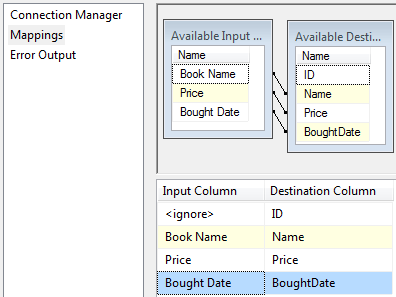
[Name] varchar(100) not null,

[Price] float not null,

[BoughtDate] datetime null

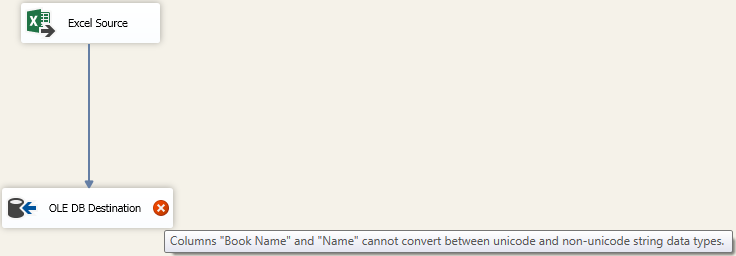
)

1. Click OK to close "Create Table" window. Then click Mappings tab and set the Input Columns as below.



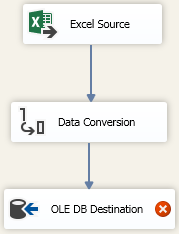
1. Click OK to close the editor.

But there is an error message displayed with a red cross icon. Put your cursor on the icon, an error message will be displayed as below.

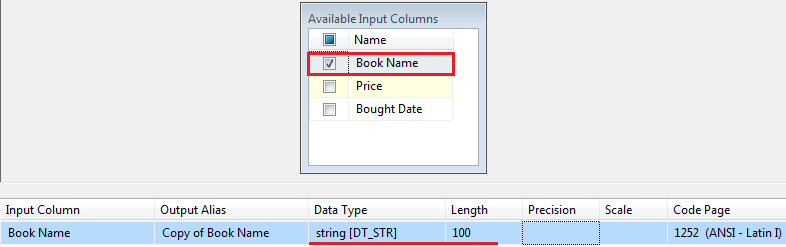


In this case, a Data Conversion transform is needed.

1. Remove the link between the Source and the Destination and drag @ drop a Data Conversion transform between them and have them linked like below.

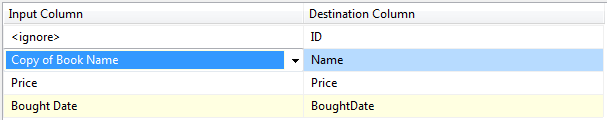


1. Right-click the Data Conversion and choose "Edit..." to open its editor. Then Click "Book Name", change the data type to string and Length to 100 as follows.



Please be noted, the output name of the "Book Name" is called "Copy of Book Name" with string data type.

1. Click OK button. Then open the OLE DB Destination Editor again and change the "Book Name" to "Copy of Book Name" in the Mappings tab.



1. Click OK button to finish the package.

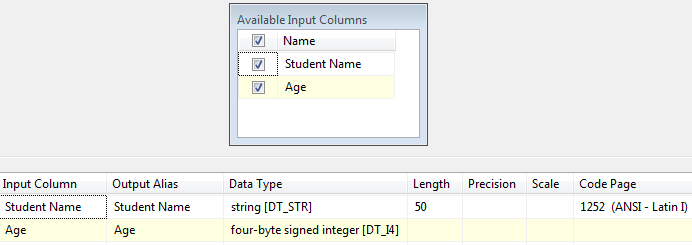
At last run the package and you will find the new table books is created and 4 records are added.

Now, it is time to review the package ImportExportWizard.dtsx which was created in Import Export Wizard section.

1. Open the package and click Data Flow tab.

You can see a Data Conversion transformation was created between the source and destination.

1. Open the editor of the Data Conversion. You can see the conversions were created below.



1. Click Cancel button to close the editor.

In summary, Data Conversion transformation is used to convert a data type to another one in order to match its counterpart in the pipeline of the Data Flow Task.

# SSIS Data Flow Advanced Editor

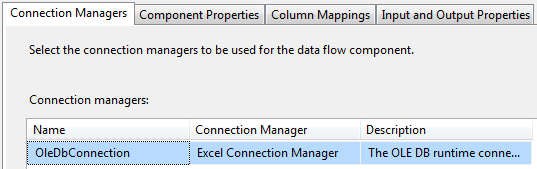
Nearly all the tasks or data flow components (transforms) in SSIS packages have their own editors or custom user interfaces to set their specific properties or fields. This editor can be opened by right-clicking the item and select "Edit...". Most of the transforms also have advanced editors which can be opened by right-clicking the item and select "Show Advanced Editor...". The Advanced Editor is standardized as following 5 tabs and one transform can have up to 4 of them.

* Connection Managers
* Component Properties
* Column Mappings
* Input Columns
* Input and Output Properties

To understand all of them, we'll check the editor of the transforms which were created in the Data Conversion section.

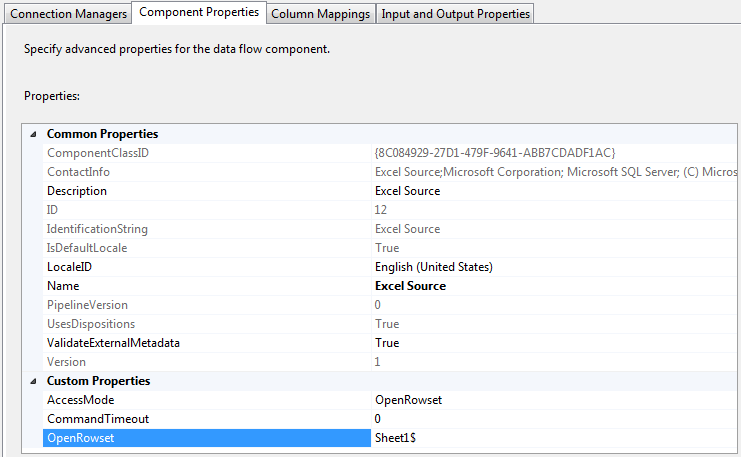
1. Open the package DataConversion.dtsx in the project LeanSSIS2 in Visual Studio.
2. Click Data Flow tab and right-click "Excel Source" and choose "Show Advanced Editor..." to open the editor.

The first Connection Managers will be shown up.



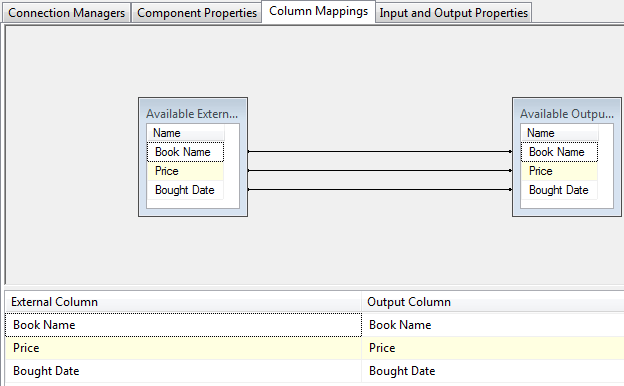
In this tab, the connection managers which will be used in the transform are set here. In this example, it was set as Excel Connection Manager.

1. Click "Component Properties" tab.



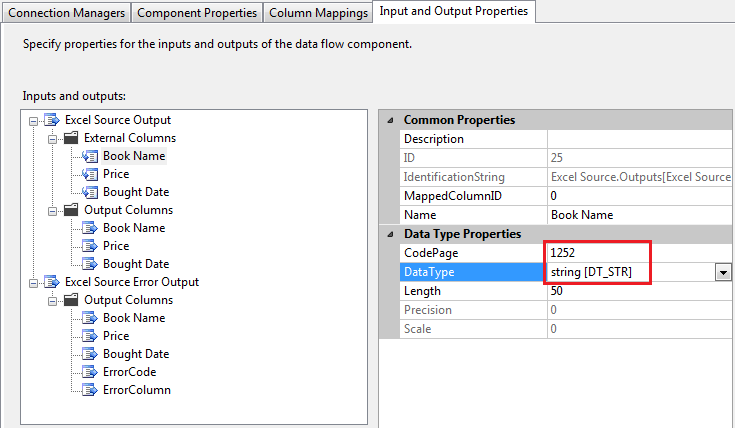
As you see, the Component Properties are the same as that in the properties window outside the editor. The OpenRowset property is the table or view level setting and it was set as Sheet1$ in this example.

1. Click "Column Mappings" tab.



The column mappings between the input columns and output columns can be set here.

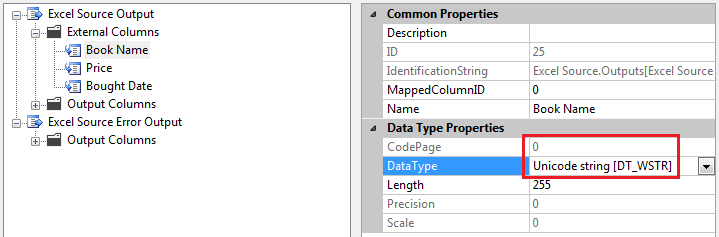
1. Click "Input and Output Properties" tab. Then click "Book Name" in External Columns in Excel Source Output and change the Data Type to string on the right side.



There are 2 outputs in this source transform. One is called Excel Source Output which is normal output and the other is Excel Source Error Output which is the error output if an error occurs. In the Excel Source Output, there are 2 sets of columns. The External Columns stand for the columns definition in the data source and Output Columns stand for the output of the transform. The schema data of the External Columns cannot be changed actually if the data source has not been changed. In this example, we changed the Data Type of "Book Name" column on purpose and see what will be happened.

Please be noted, 2 columns ErrorCode and ErrorColumn are added in the Error Output Columns.

1. Click "OK" to save the change and reopen the editor and go to the last tab.

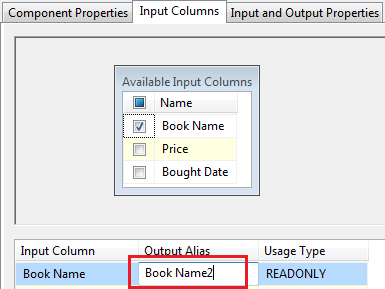


You will find the data type was changed back automatically.

1. Click "Cancel" to close the editor. Right-click "Data Conversion" to open its Advanced Editor.

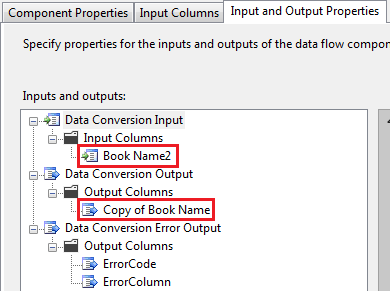
The default tab is Component Properties which was already be mentioned above.

1. Click "Input Columns" and change the "Output Alias" to "Book Name2".



In this tab, the converted input column was selected and shown up. The output alias of the input column was changed on purpose.

1. Click "Input and Output Properties" tab.



You can see the alias name you just typed is actually input alias name which is different from that of output alias name.

1. Click "Cancel" button to close the advanced editor.

In summary, The advanced Editor provides a way to access the low-level properties of data flow components and a detailed data structure in both input and output columns.

# SSIS Data Flow Error Output

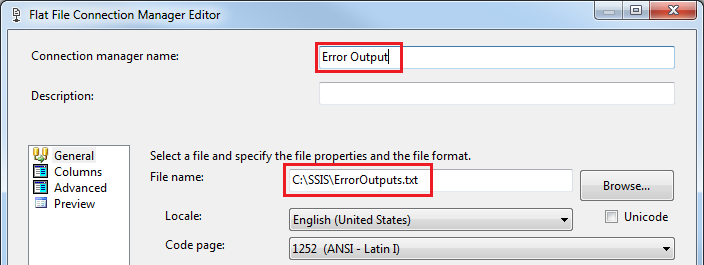
Error outputs are the output stream of a data flow component whenever an error occurs on the component. It is well supported by the most components and 2 columns Errorcode and ErrorColumn are added automatically in the output.

In the following example, we'll add 2 records with wrong date format and wrong number in the source Excel file and save the error outputs to a flat file.

Let's add 2 wrong records in the Excel file C:\SSIS\Books.xlsx first.

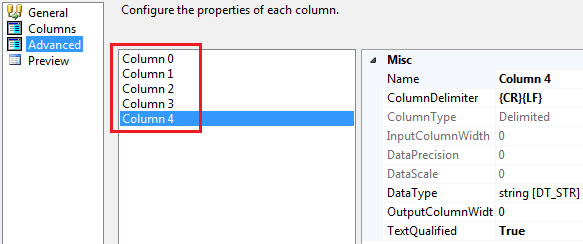
Then start to create the example by the following step-by-step.

1. Create a copy of the package DataConversion.dtsx and rename it to ErrorOutput.dtsx. Then open it.
2. Create a Flat File Connection Manager "Error Output" as shown below.



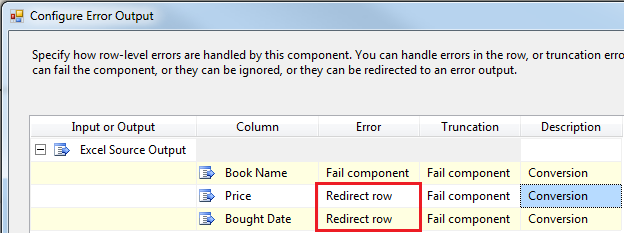
The outputs will be saved to the file C:\SSIS\ErrorOutputs.txt after the package executes.

1. Click "Advanced" tab and add 5 columns, then click OK to finish the settings on the Flat file connection manager.



1. Click Data Flow tab and drag & drop Flat File Destination into the data flow edit panel and link the Error Outputs (Red Arrow) of the Excel Source to it.

The Configure Error Output dialogue box will be popped up.



The Error column saves the action when an error occurs and the Truncation column saves the action when a truncation occurs, for example, trying to save a string with 20 length to a field with 10 length. The table below will explain the 3 action options of which you will have to choose one.

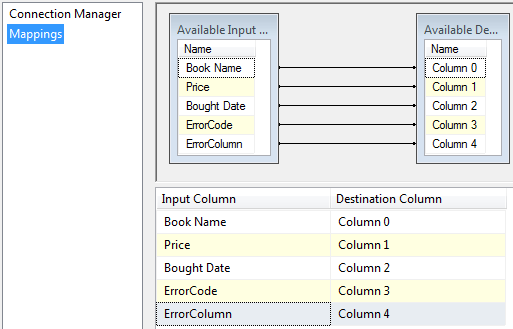
|  |  |  |
| --- | --- | --- |
| **Option** | **Explanation** | **Default Option** |
| Fail Component | The whole Data Flow task fails when an error or a truncation occurs. | Yes |
| Ignore Failure | Ignore the error or the truncation and direct the row to the normal output. | No |
| Redirect Row | Direct the row to the error output when an error or a truncation occurs. | No |

In this example, we just change the Error column of Price and Bought Date row to "Redirect row".

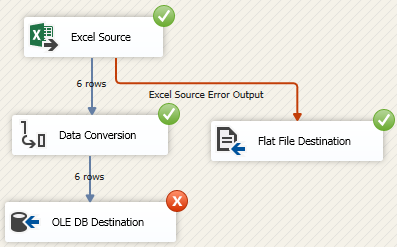
1. Click OK button. Then double-click the Flat File Destination component to open the Editor.

The connection manager is automatically assigned to the manager we just created.

1. Click Mappings tab and set the Input Columns to match Destination Columns as below.



1. Click OK button to close the editor. Then run the package and you'll get the following result.

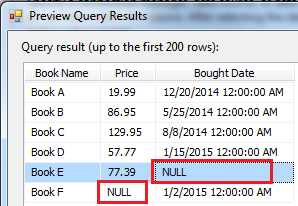


This result is not the expected one. What's happened? Why was the Excel Source returned nothing in Error Outputs?

Based on the Microsoft explanation, the Excel driver gets the first several rows to evaluate the data type of each column. By default, they are the first 8 rows of the Excel file. Once the driver decides in favour of the majority data type, the cells with the other types will be regarded as NULL value when the data is being loaded.

In this case, we'll check the data in Excel Source.

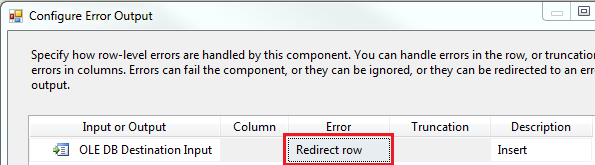
1. Open Excel Source Editor and Click "Preview..." button in the Connection Manager tab.



As you can see, the cells with invalid data were changed to NULL values. There is no records going to Error Outputs.

Let's capture the destination error outputs and see what will be happened.

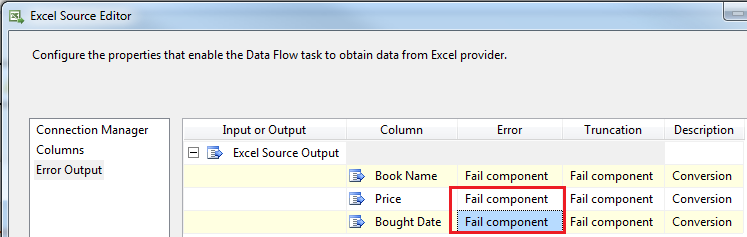
1. Remove the Error Output from Excel Source and put the error output of OLE DB Destination to Flat File Destination. And change the Error column to "Redirect row" in the pop-up dialogue.



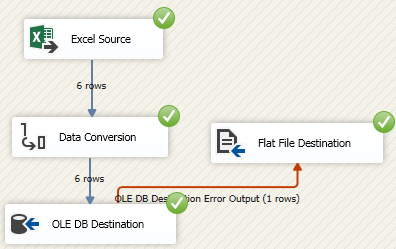
1. Click OK button.

There is a warning sign in the Excel Source. We'll have to change back the setting which we did on step 4 .

1. Open the Excel Source editor, click the "Error Output" tag and change back the configuration to its default settings.

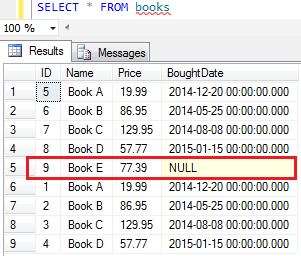


1. Click OK and run the package again.



Only 1 record went to Error Outputs. Why?

Here is the reason.



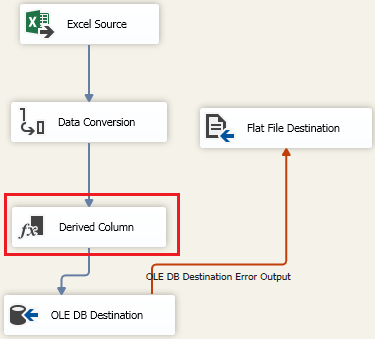
Another record with NULL value in the BoughtDate column was already added in the books table because NULL is allowed in BoughtDate.

# SSIS Derived Column Transformation

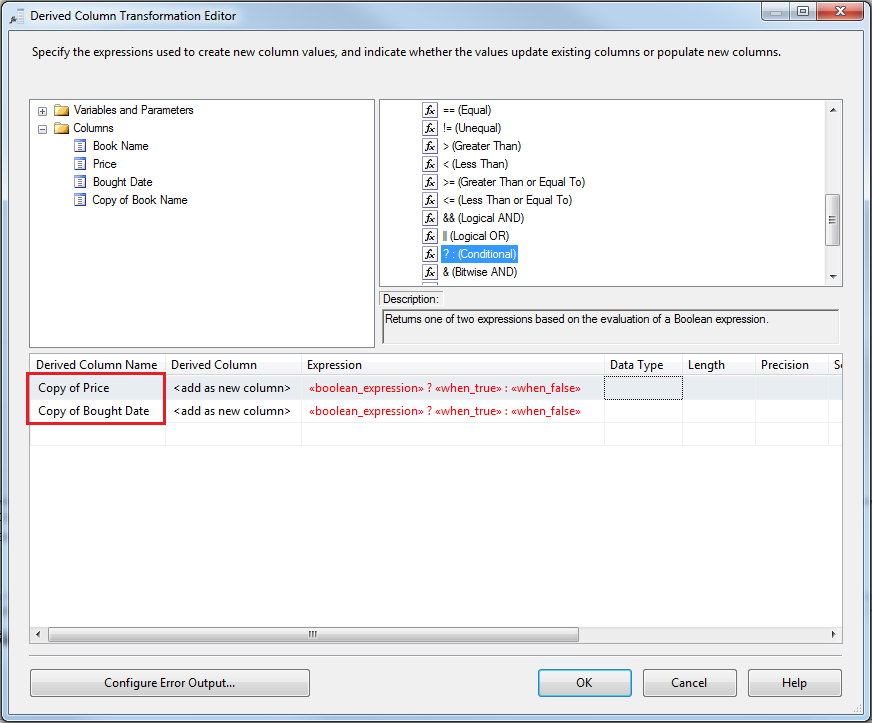
The Derived Column transformation provides a way to create new columns by using expressions on the input columns. Like Expression Task, The expressions can be created in the Derived Column Transformation Editor which is similar to the Expression Builder. The new columns will be added to the output columns of the transformation.

In the following example, we'll create a new package from the one we created in Error Output section and add 2 derived columns in the Derived Column transformation to handle NULL value situations. The source Excel file Book.xlsx is kept unchanged. Let's get started to build the new package.

1. Copy & paste the package ErrorOutput.dtsx in the project LearnSSIS2 to create a new one and rename it to DerivedColumn.dtsx.
2. Click Data Flow tab and drag & drop the Derived Column from SSIS Toolbox to the data flow and link them as below.

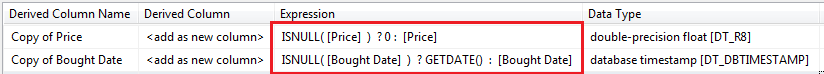


1. Right-click "Derived Column" transform and choose "Edit..." to open the Derived Column Transformation Editor. Then add 2 derived columns "Copy of Price" and "Copy of Brought Date" and drag & drop Conditional operator to the Expressions of the 2 columns.



Apparently, the expressions are not finished so they are displayed in red. Keey the default setting "" in Derived Column because we don't want to replace any columns. If you want to configure the error output of this transform, you can click the button "Configure Error Output...".

1. Finish the expressions.



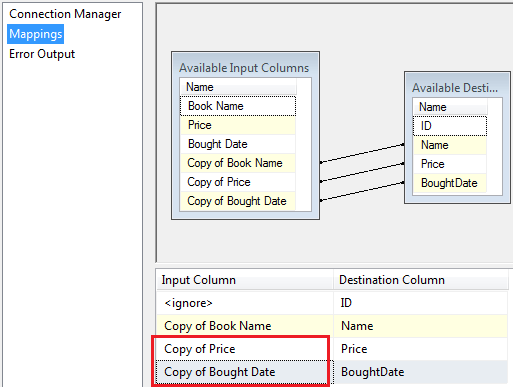
The price is set to 0 if it is NULL and the bought date is set to the current date if it is NULL.

1. Click OK button.

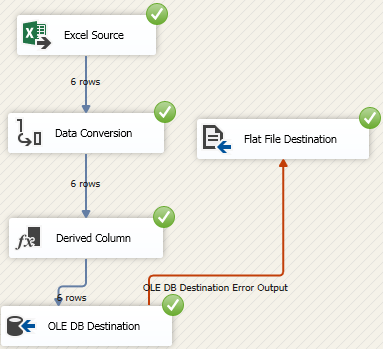
You can see there is no error icons. Is it done? Actually not.

The new derived columns have not been added in the data pipeline and the old mapping is still defined in the destination transform.

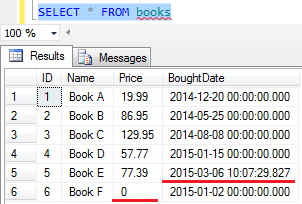
1. Right-click the "OLE DB Destination" to choose "Edit..." to open its editor, then click Mappings tab and change the Input Column as shown below.



1. Click OK button, remove all the records in Books table in Test database and run the package.



There is no error column any more. Let's check the database table.

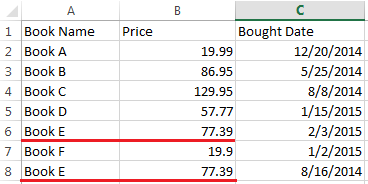


The 2 records were added as expected.

# SSIS Data Viewer

A data viewer is used to display data between two data flow transformations in SSIS packages.

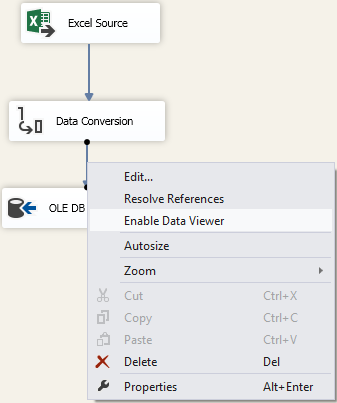
Before creating the package, let's change the content of the source Excel file C:\SSIS\Books.xlsx as follows.



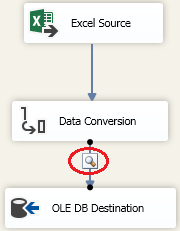
Data Viewer

Now we'll follow the steps below to create a data viewer on an output and see the result.

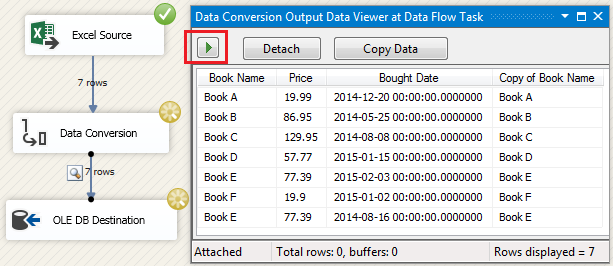
1. Make a copy of the package DataConversion.dtsx and rename it to Sort.dtsx. Then open it and click Data Flow tab.
2. Right-click the path between Data Conversion and OLE DB Destination and click "Enable Data Viewer".



A magnifier icon is shown up on the path.



1. Run the package and the data will be shown up in the data viewer.



In Data Flow task, data is transferred buffer by buffer. That means the data will be saved to a buffer before going to the next transformation. The buffer exists in memory or disk with limited size. The buffer size can be set in the property of the task. Now we can explain the above buttons in the data viewer.

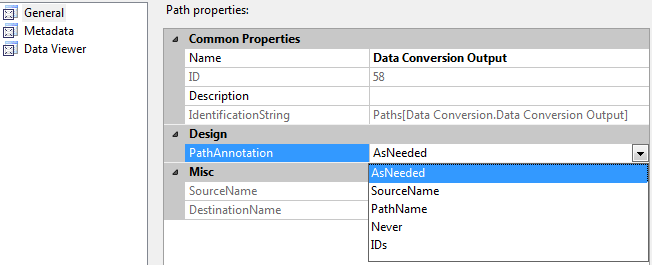
|  |  |
| --- | --- |
| **Option** | **Description** |
| Green arrow | Displays the next buffer's data if we still have. |
| Detach | Data continues to flow through the path without stops and the data in the viewer will be kept. |
| Attach | Displays the data in the current buffer and pause until Green arrow or Detach button is pressed. |
| Copy Data | The data in the current buffer is copied to Clipboard. |

1. Click Green arrow button to get over the execution.

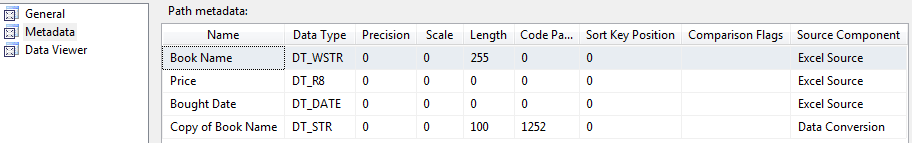
Data Flow Path Editor

This is the editor to set data flow path settings. Let's continue the work.

1. Right-click the path again and choose "Edit..." to open the editor.
2. In the General tab, click on PathAnnotation and you can choose the items below to show them up on the path or nothing.

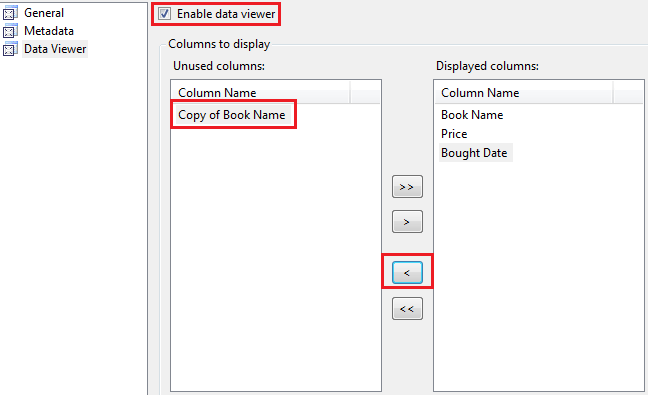


1. Click Metadata tab, you can see the meta data which flows through the path.



You can click "Copy to Clipboard" to copy the meta data to clipboard.

1. Click Data Viewer tab, you can see the data viewer is enabled and all the columns are set to be viewed as default.
2. Choose "Copy of Book Name" column and click "<" button to move the column to Unused columns. At last, click OK button.



1. Run the package again and you will see the column "Copy of Book Name" will not be shown up in the data viewer.

# SSIS Aggregate Transformation

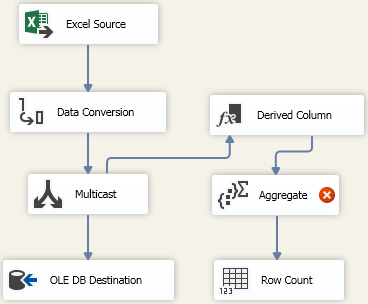
Like the aggregate functions in SQL SELECT statement, the Aggregate transformation is used to apply the aggregate functions to the input columns and put the result in the output columns. The GROUP By clause is provided to specify groups to aggregate across.

The operations supported in the transformation are listed below.

|  |  |  |
| --- | --- | --- |
| **Operation** | **Description** | **Transact-SQL Keywords** |
| Group by | Divides datasets into groups | GROUP BY |
| Count | The number of rows in a group | COUNT |
| Count distinct | The number of unique non-null values in a group | N/A |
| Sum | The total of the values in a column | SUM |
| Average | The average of the values in a column | AVG |
| Minimum | The minimum value in a group | MIN |
| Maximum | The maximum value in a group | MAX |

In the following example, I'm going to use the Aggregate transformation to get the number of books and its total cost by the year of the bought date. We'll use the the same data source as that in the previous section.

1. Copy & past to a copy of the package Multicast.dtsx and rename the copy to Aggregate.dtsx.
2. Open the package and click Data Flow tab, then remove the output from the Multicast to the Row Count transform.
3. Drag and drop Derived Column and Aggregate transformation to the editor area and link them together as shown below.

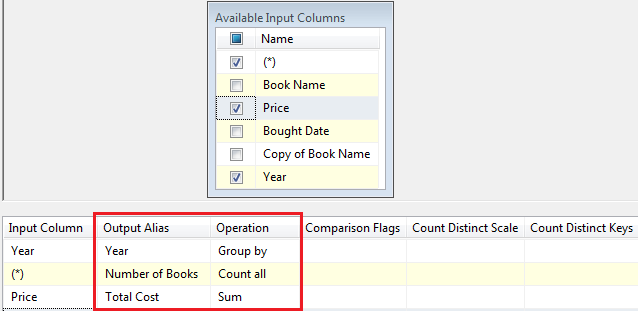


1. Right-click Derived Column transform and choose "Edit..." to open its editor. Add a new column Year and configure its expression like below then click OK button.

[New Derived Column]

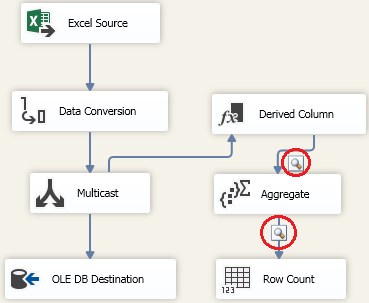
We'll group by the new column "Year" in the next step.

1. Right-click the Aggregate transform and choose "Edit..." to open the Aggregate Transformation Editor. Then choose Year, (\*) and Price and make their settings.

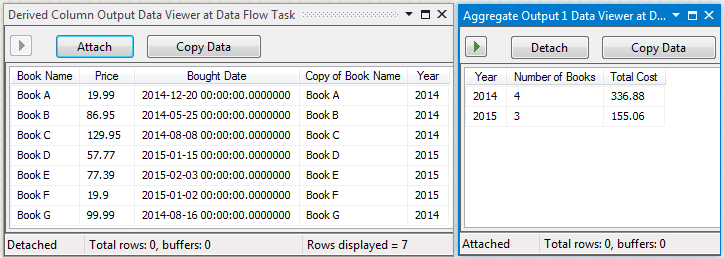


We chose "Group by" operation for Year, "Count all" for all the records with its output alias changed to "Number of Books", and "Sum" for Price with Output Alias changed to "Total Cost". CountDistinctScale and CountDistinctKeys are properties for Distinct count operation and their settings will improve running performance.

1. Click OK button. Then add Data Viewer before and after the Aggregate transform.



1. Run the package.



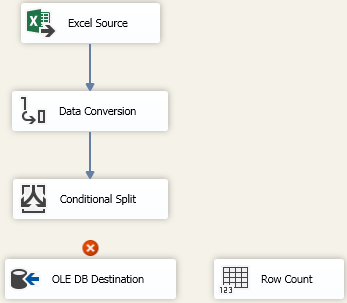
The left side is the Input data of the Aggregate transform and the right side is the output data. As you can see the Aggregate transformation generates aggregation data based on its input data.

# SSIS Conditional Split Transformation

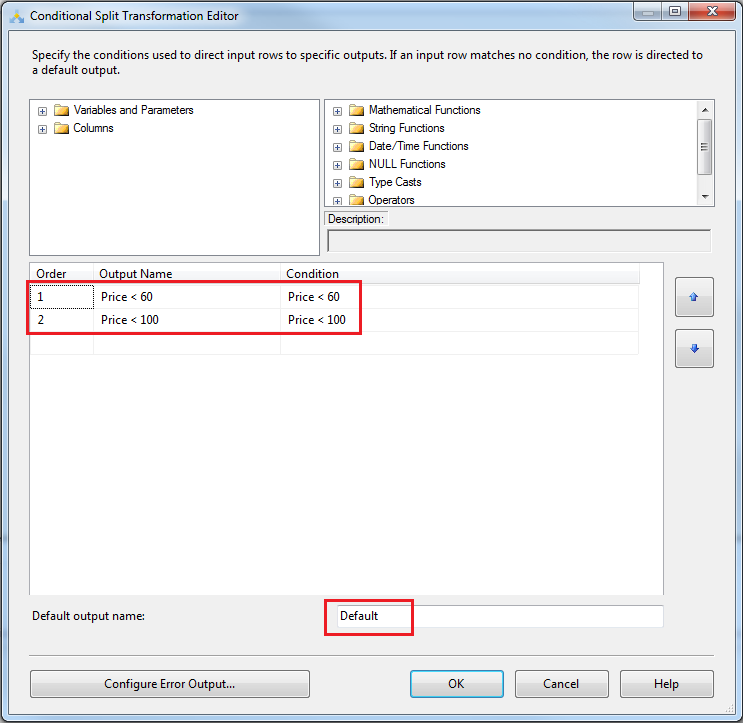
Like Multicast transform, the Conditional Split transformation also can be defined with multiple outputs. But instead of copying the input data as an output, it evaluates the defined expressions and based on the result to direct output. It mostly likes the Switch Statement in C#. At run-time, each row in the data flow will be evaluated the condition of the first output, if it is true, the row will go to the output otherwise start to evaluate the next condition of the second output. If the row cannot meet any conditions we defined, it will go to the default output. A default output must be defined for each Conditional Split transformation.

Let's do some testing on this transform.

1. Make a copy of Multicast.dtsx in the project LearnSSIS2 and rename the copy package to ConditionalSplit.dtsx.
2. Open the package and click Data Flow tab at the top to go to design panel. Then remove the Multicast transform.
3. Drag & drop the Conditional Split transform to the panel and link them as shown below.

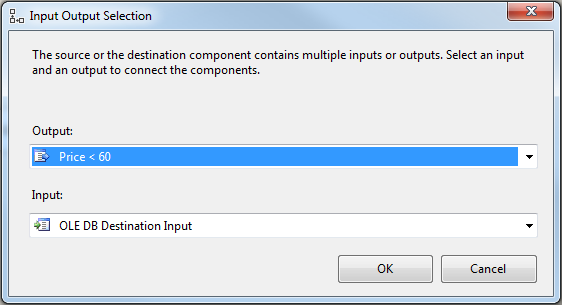


1. Right-click Conditional Split transformation and choose "Edit..." to open its editor and make the settings as follows.

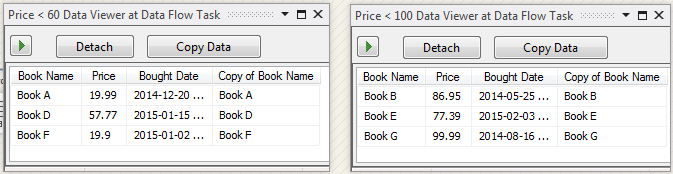


The default output is defined as "Default" and you can configure the error output on this transform as well.

1. Connect the output "Price < 60" to OLE DB Destination. A dialogue box will be popped up as below. You then choose "Price < 60" and click OK button.



1. Repeat step 5 to connect the output "Price < 100" to the Row Count transform.
2. Enable the data viewer on these 2 outputs. Then run the package.



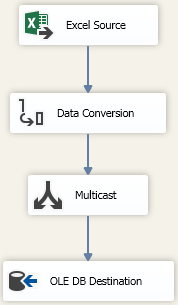
The result is the same as that we expected. Only one record whose price is greater than 100 is going to Default output which is not defined to accept the data in the data flow.

# SSIS Multicast Transformation

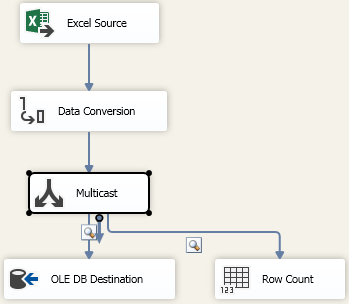
The Multicast transformation has one input and multiple normal outputs without an error output. It just directs its input to each output it had. That means each output data owns a copy of its input data.

We will use the same source data of the previous section to create a package to test the transformation.

1. Make a copy of DataConversion.dtsx in the project LearnSSIS2 and rename the copy package to Multicast.dtsx.
2. Open the package and click Data Flow tab at the top to go to design panel.
3. Drag & drop Multicast transform to the panel and link them as shown below.

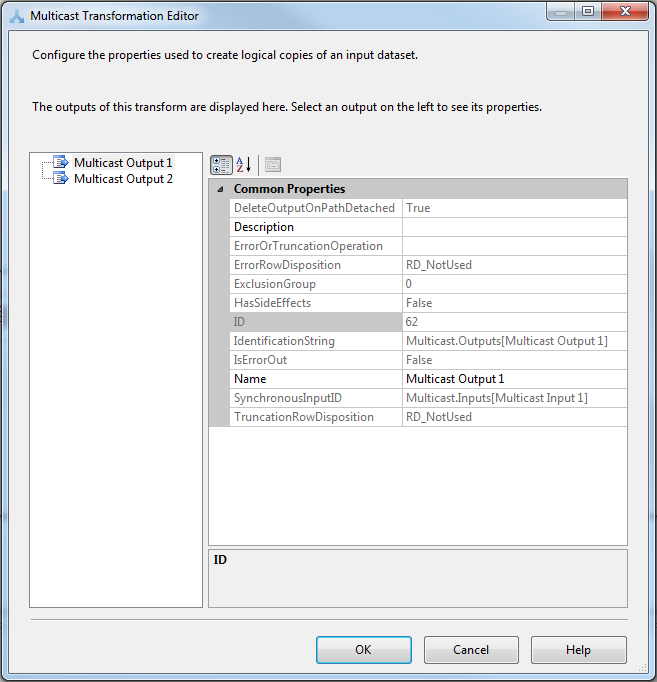


1. Create a variable Count with Int32 type and drag & drop Row Count into the panel and link it to another output of Multicast.
2. Select the Variable User::Count in the Row Count and add data viewer on each output of the Multicast transform.



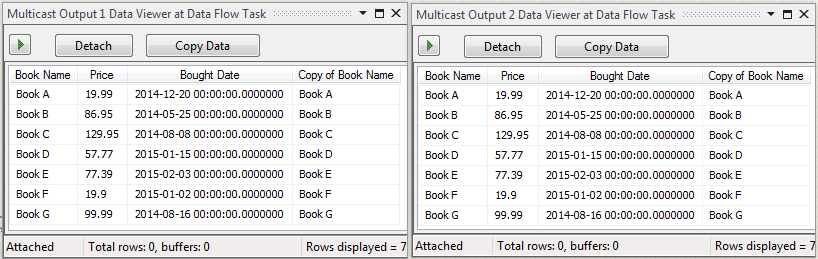
You can repeat the step 4-5 to get more outputs of Multicast. By now, we have not seen the editor of this transform.

1. Right-click the Multicast and choose "Edit..." to open its editor.



Everything is read-only except the Name property.

1. Click Cancel button and run the package.



You can see the data in the 2 viewers are exactly the same.

# SSIS Row Count Transformation

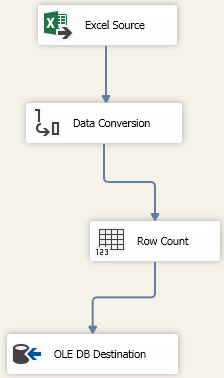
The Row Count might be the simplest one of all the transformations in Data Flow Task. It just counts the rows of the data which pass through the transform in a data flow and save the number in a variable. The transform has one input and one output without any error output.

We prepared the source Excel file C:\SSIS\Books.xlsx with its contents below. The total number of records is 7.



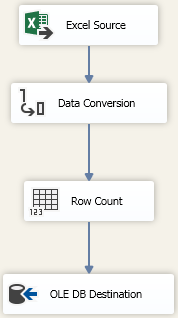
Just follow the steps below to see how to create and use the row count transformation.

1. Copy and paste the package DataConversion.dtsx in project LearnSSIS2 and rename the copy to RowCount.dtsx.
2. Open the package and go to Data Flow Task design panel by clicking Data Flow tab at the top. Then drag & drop Row Count from Toolbox into the panel.



The layout looks ugly. Let's adjust it.

1. Use your mouse to select all the transforms or just click CTRL-A, then click the menu FORMAT -> Align -> Centers, and then click the menu FORMAT -> Vertical Spacing -> Decrease several times.

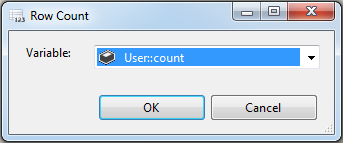


Now the layout looks good.

1. Create an Int32 variable count.

If you don't know how to create it, please check the Variable section.

1. Right-click the "Row Count" and select "Edit...". Then choose the variable "User::count" in the drop-down list.

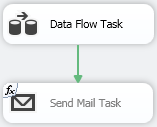


1. Click OK to get over the setting.

To verify the result, we'll use a Send Mail Task to get the count of the inserted records.

1. Click Control Flow tab and add a Send Mail Task and replace the MessageSource with the following in the Expressions of the editor.

"There are " + (DT\_WSTR, 20) @[User::count] + " records added to the database."



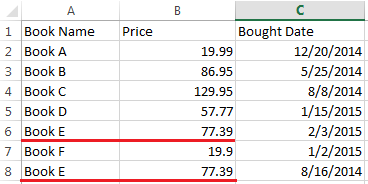
For detail settings, please check Send Mail Task section.

1. Run the package and you will get an email indicating 7 records added to the database.

# SSIS Sort Transformation

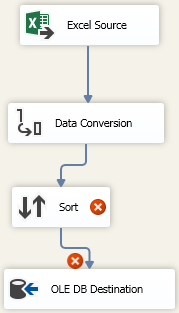
The Sort transformation is mostly like ORDER BY clause in SQL statement to sort data. It has one input as data source and one output to export sorted data. There is no error outputs for this transformation. You can sort data by one column or a couple of them and the sort type is either ascending or descending for each column. The sorted columns have orders to sort the data. Only when the first sort columns of the data are the same, the second sorted column starts to be used to compare data.

The data source is still the Excel file C:\SSIS\Books.xlsx which was changed in the previous section.

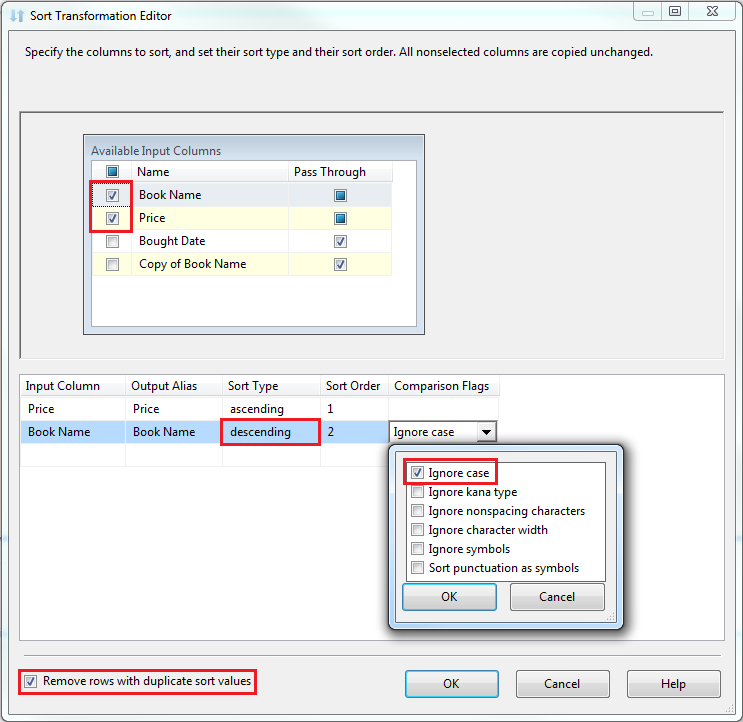


We'll sort the data by price and book name and the record with the same sort key can be removed if we make a proper setting.

1. Open the package Sort.dtsx if it is closed and click Data Flow button.
2. Delete the path of Data conversion output, drag & drop the Sort transform under the "Data Conversion" and link them as shown below.



1. Right-click the Sort transform and choose "Edit..." to open the Sort Transformation Editor and set it as shown below.



We choose "Price" as the primary sort column and "Book Name" as secondary one with descending type. Because the Book Name is string-typed, "Ignore case" is chosen in the Comparison Flags. At last, we have "Remove rows with duplicate sort values" checked.

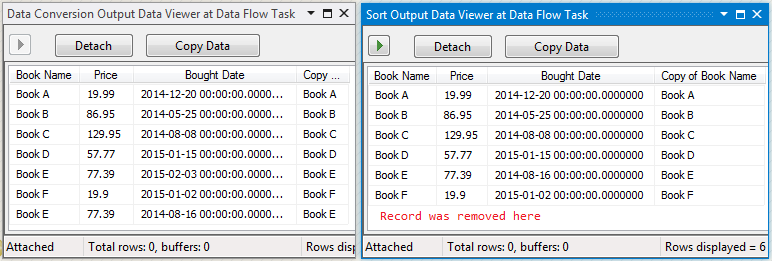
1. Click OK button to close the editor.

The red error icons in the data flow will be removed.

1. Add a data viewer above the Sort transform and another below it. Then run the package.

The data in first data viewer will be shown up first with original sequence.

1. Click the Green arrow in the viewer and second viewer will be shown up.



The record with the same sort columns was removed in the second viewer.

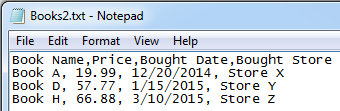
1. Click the Green arrow in the second one and the execution is over.

In summary, the Sort transform is not only sorting the data by the selected columns but also removing the records with the same value of the sorting columns

# SSIS Union All Transformation

The Union All transformation is used to combine all the inputs into one output and the columns in the inputs must be mapped to columns in the output. Ideally all the inputs have the same number of columns with the same data type. If not you will have to make sure at least one column in the inputs mapping to the column in the output and there is no conflict among the data types of the mapping columns.

In the upcoming example, I'm going to get a copy of the data Conversion package, use the same Excel file as a data source and add a new data source from a flat file. Then we'll use the Union All transformation to combine the 2 inputs into one output. Let's check the Flat file first.

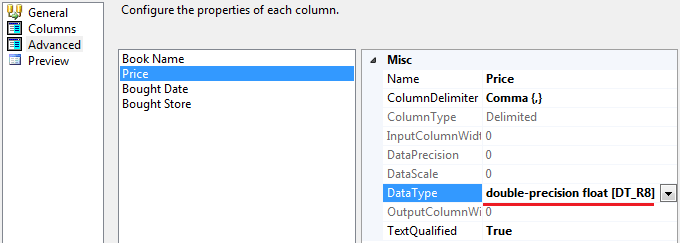


The file Books2.txt was saved in C:\SSIS folder and we added additional column "Bought Store" for testing purpose. Take the following steps to test the transform.

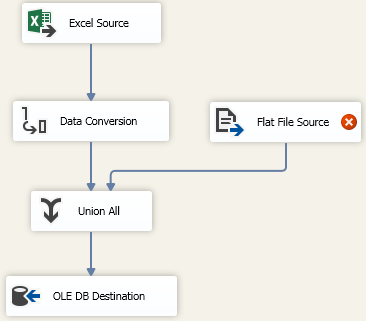
1. Make a copy of the package DataConversion.dtsx and rename the copy to UnionAll.dtsx in the project LearnSSIS2 which created in the previous section.
2. Double click the new package to open it and create a Flat File Connection Manager to point to the flat file we displayed above.

The default data type of all the columns are string type.

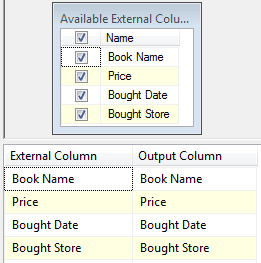
1. Change the data type of the Price column to DT\_R8 and the type of Bought Date to DT\_DATE in the Advanced tab in Flat File Connection Manager Editor.



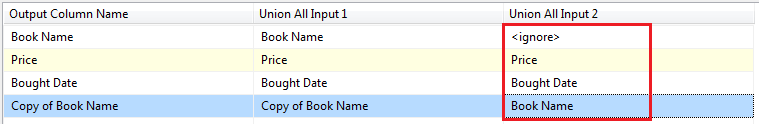
1. Click OK button to close the editor and then click Data Flow tab to go to data flow edit area.
2. Remove the output from the "Data Conversion" transform and drag & drop the "Union All" transformation and Flat File Source to the data flow edit area and link them with the other transforms as shown below.



1. Right-click the Flat File Source and choose "Edit..." to open the editor and select the Flat File Connection Manager as its connection manager. Click Columns tab, the columns looks like below.

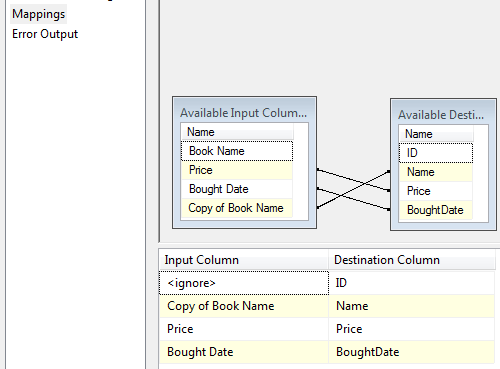


1. Click OK button. Then right-click Union All transform and choose "Edit" to open the editor. Then change the setting in Union All Input 2 as shown below.

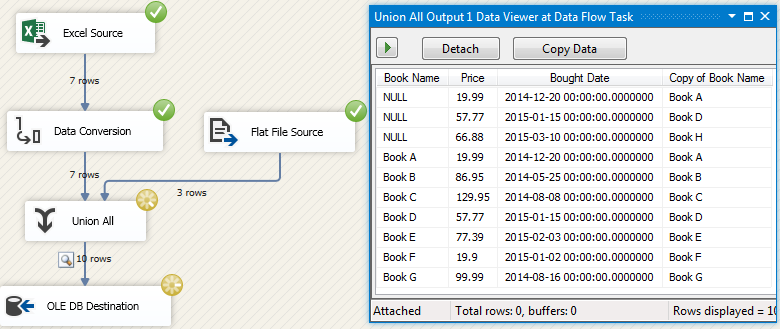


There is no column in Input 2 to map to the output column "Book Name" because it was converted to Copy of Book Name with string type. You can add a "store" column at the end of the mapping list if you want.

1. Click OK button and Right-click OLE DB Destination to open its editor and click Mappings tab to make sure the mappings of columns are correctly.



1. Click OK and add data viewer between Union All and OLE DB Destination and then run the package.



The first 3 records in the data viewer are the data from the flat file and the others are from the Excel file. No record was removed even if the record had the same data as the other's.

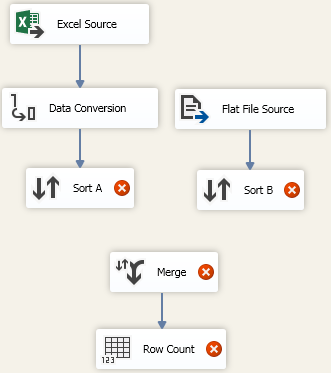
# SSIS Merge Transformation

The Merge transformation is used to combine 2 sorted inputs into a single sorted output. The key columns of the output are defined by the same keys of the inputs. Like Union All transformation, The data types of the mapping columns in input datasets must be the same. The following will demonstrate the difference between the Merge transformation and the Union All transformation.

|  |  |  |
| --- | --- | --- |
| **Items** | **Merge** | **Union All** |
| How many input datasets are there in the transformation? | 2 | Multiple |
| Must the input datasets be sorted in the transformation? | Yes | No |
| Is the output dataset sorted? | Yes | No |

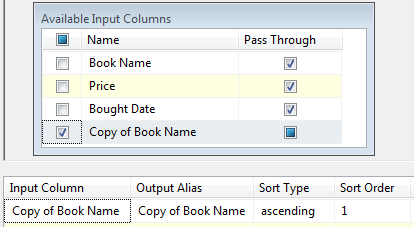
In the following example, I'll make a copy of the Union All package and change the Union All to Merge transformation. Also we'll have to add Sort transform before each inputs. Let's get started.

1. Open the project LearnSSIS2 which was created in a previous section in Visual Studio 2012 and make a copy of the package UnionAll.dtsx and rename it to Merge.dtsx.
2. Open the Merge.dtsx package and click Data Flow to prepare editing the data flow.
3. Remove the "Union All", the OLE DB Destination and its connection manager. Then drag & drop Sort, Row Count and Merge transforms into the package and link them as shown below.

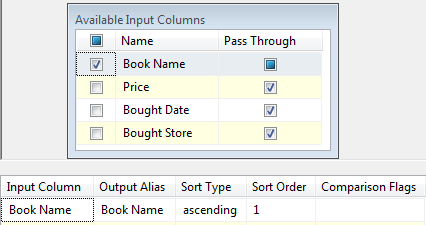


The two sort transforms were renamed as "Sort A" and "Sort B".

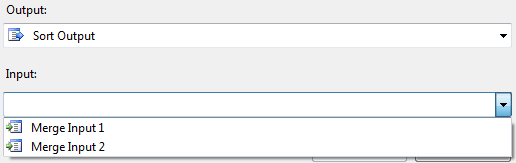
1. Right-click ""Sort A" and choose "Edit..." to open its editor and click on "Copy of Book Name" to choose it as sort column.



1. Click OK button and open the editor of "Sort B" and choose "Book Name" as the sort column.



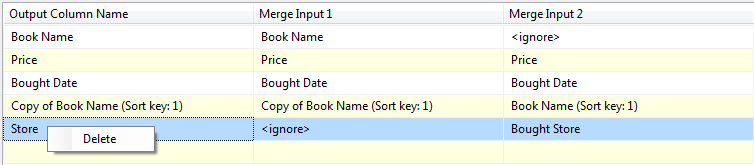
1. Click OK button and drag the output of "Sort A" to Merge transform. An Input Output Selection dialogue will be popped up.



1. Choose "Merge Input 1" and click OK. Then drag the output of "Sort B" to Merge transform.

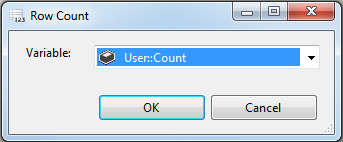
This time there is no pop-up dialogue any more because the last option "Merge Input 2" was chosen automatically.

1. Right-click the Merge transform and choose "Edit" to open Merge Transformation Editor and make the changes as below.

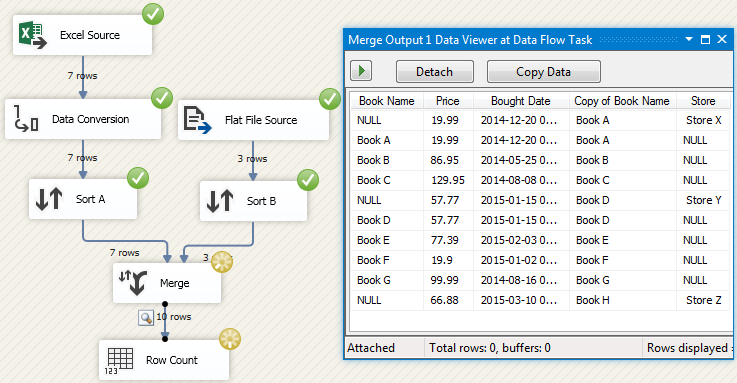


Choose "Book Name" for the output column "Copy of Book Name" in the "Merge Input 2" column and add a new output column "Store". If you want to delete one line, you just right-click on the line and choose "Delete".

1. Click OK button. Then create a variable "Count" with data type "Int32" and default value 0.
2. Double click the "Row Count" transform and select the variable and click OK.



1. At last, add a data viewer between Merge and Row Count transform and run the package.



The data from 2 data sources are merged and sorted. The new column "Store" was added.

1. Click "Stop Debugging" to terminate the execution.

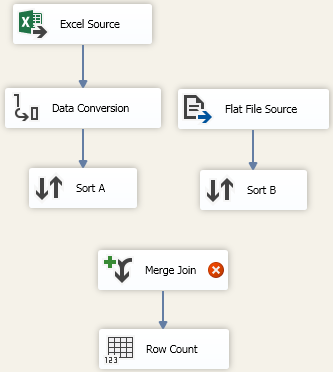
# SSIS Merge Join Transformation

Like INNER JOIN, LEFT JOIN AND FULL OUTER JOIN in SQL statement, the Merge Join transformation provides INNER, LEFT and FULL join between two sorted datasets and output the join result. In the following table, The red colour stands for the Join result of the dataset A and B.

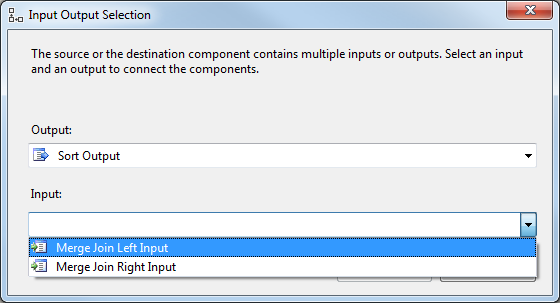
|  |  |  |
| --- | --- | --- |
| **Left Join** | **Full Join** | **Inner Join** |
| [Left Join] | [Full Join] | [Inner Join] |

In the following example, I'm going to replace the Merge transformation with the Merge Join transformation in the copy of Merge.dtsx package and trying to use 3 types of Join to see what difference is among them.

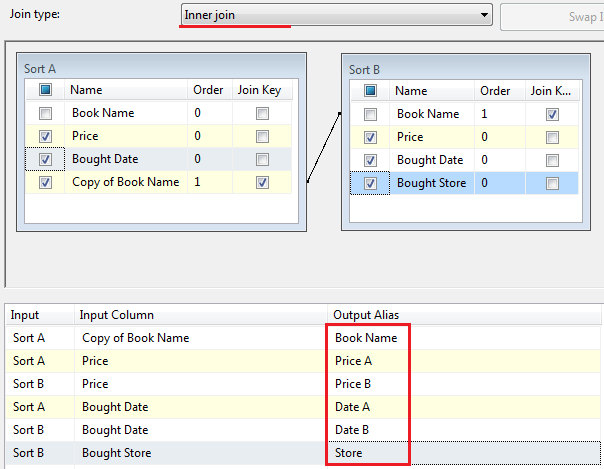
1. Open the project LearnSSIS2, make a copy of Merge.dtsx and rename it to MergeJoin.dtsx.
2. Open the package MergeJoin.dtsx and click Data Flow to go to the data flow edit area of the task.
3. Remove the Merge transform and add a new Merge Join from Toolbox and link them as below.



1. Put the output of "Sort A" to Merge Join transform to open the Input Output Selection dialogue box, then select "Merge Join Left Input" in Input drop-down list and click OK button.

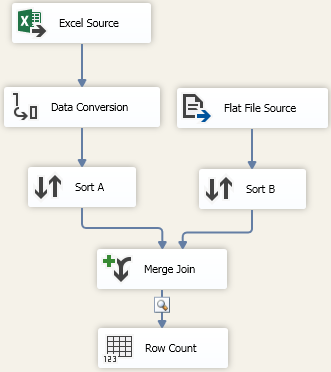


1. Put the output of "Sort B" to Merge Join transform and this input is set as "Merge Join Right Input" automatically.
2. Right-click the Merge Join and choose "Edit..." to open the editor. Leave the default Join type "Inner join" unchanged and add the output columns as below.



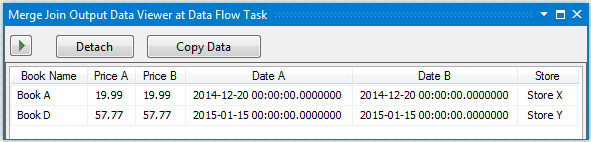
The output columns were renamed in the Output Alias column.

1. Click OK button and add the data viewer on the output of the Merge Join transform.



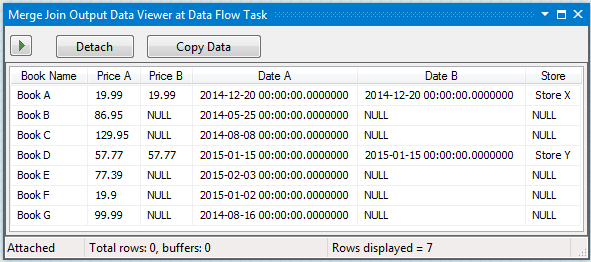
That's the data flow looking like after finishing all the above settings.

1. Run the package.



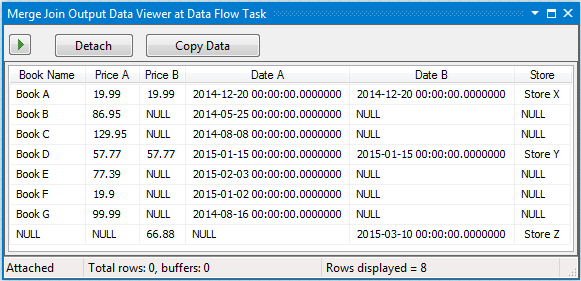
Two records were returned because both of the inputs have these records.

1. Terminate the running and repeat the step 6 to change Join type to "Left Outer Join" and run it again.



All the columns from "Sort A" input were outputted and the columns from "Sort B" were left joined to the output.

1. Terminate the running again and repeat the step 6 to change Join type to "Left Outer Join" and run it again.



This time all the records were in the output.

# SSIS Lookup Transformation

The lookup transformation is used to create a lookup dataset and join the selected columns to the major input columns based the connected key columns and it can only use an OLE DB connection manager or a Cache connection manager to connect to the lookup dataset. Also only SQL Server, Oracle and DB2 databases are supported as an OLE DB connection manager by the transform. I'll describe the transform in details in the following example.

Before starting to create the sample package, a lookup table like below must be created in the SQL Server database Test.

CREATE TABLE lendout(

ID INT IDENTITY(1,1) NOT NULL,

BookName VARCHAR(100) NOT NULL,

StudentID INT NOT NULL

)

INSERT INTO lendout(BookName, studentID) VALUES('Book A', 1)

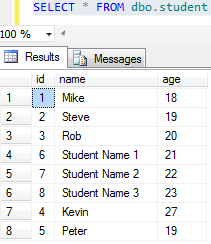
INSERT INTO lendout(BookName, studentID) VALUES('Book d', 2)

INSERT INTO lendout(BookName, studentID) VALUES('Book H', 3)

INSERT INTO lendout(BookName, studentID) VALUES('Book A', 4)

INSERT INTO lendout(BookName, studentID) VALUES('Book D', 5)

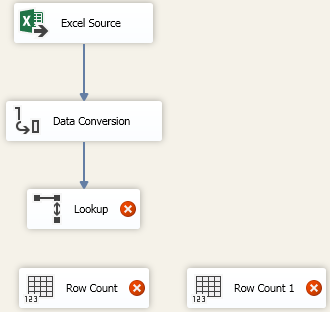
The studentID is the foreign key which is used to link to the student table defined in SQL Task None Result Set. The current records in the table are listed below.



The main data comes from Books.xlsx which is the exactly same as the data source in Data Conversion section.

Now let's get started.

1. Open the LearnSSIS2 project in Visual studio 2012 and make a copy of the package DataConversion.dtsx and rename it to Lookup.dtsx.
2. Open the package and double click the Data Flow task to go to the data flow design area.
3. Remove the OLE DB Destination transform and drag & drop a Lookup, 2 data rows transforms to the design area and have them linked as below.



1. Create 2 variables count1 and count2 with Int32 data types and assign the 2 variables to the 2 Row Count transforms.

For details, you can check the Row Count section.

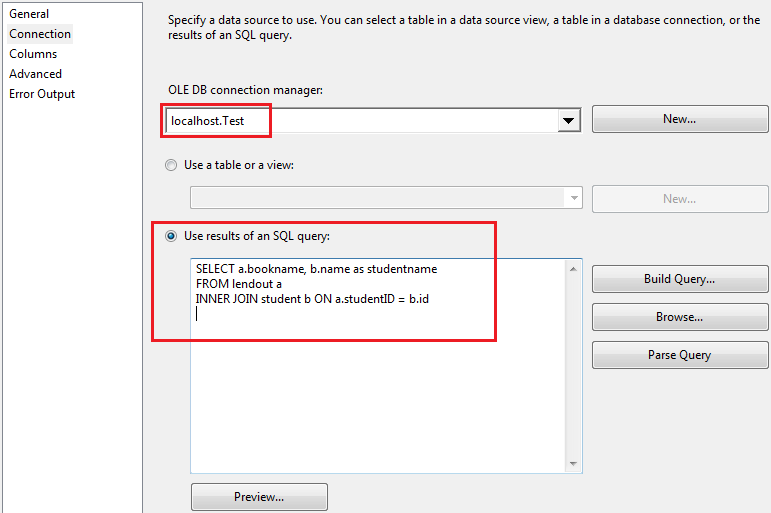
1. Right-click Lookup transform and choose "Edit..." to open its editor. Then click Connection tab, and click "Use results of an SQL query" and input the following as the query.

SELECT a.bookname, b.name as studentname

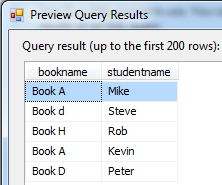
FROM lendout a

INNER JOIN student b ON a.studentID = b.id

After the setting, the editor looks like below.



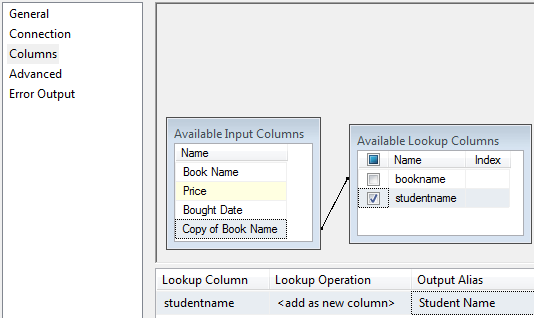
1. Click "Preview..." button the check the query results.



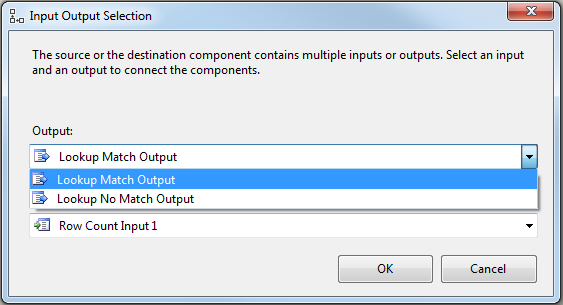
You can see "Book A" has 2 records, "Book d" and "Book D" are the same? We'll see it at the running result.

Next we'll use the "bookname" as the connect key and add the "studentname" column to the outputs.

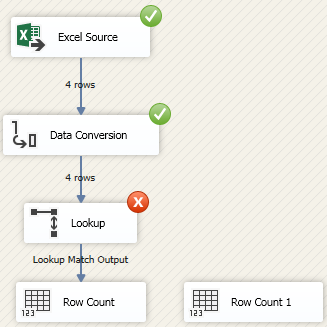
1. Click Close button and then click Columns tab, drag the key "Copy of Book Name" in Available Input Columns to the lookup key "bookname" and then click "studentname" to create a new output column, then change its Output Alias to "Student Name".



1. Click OK to close the Lookup editor. Then drag the Lookup output to the first Row count and choose "Lookup Match Output" in the pop-up dialogue box.



1. Click OK and run the package.

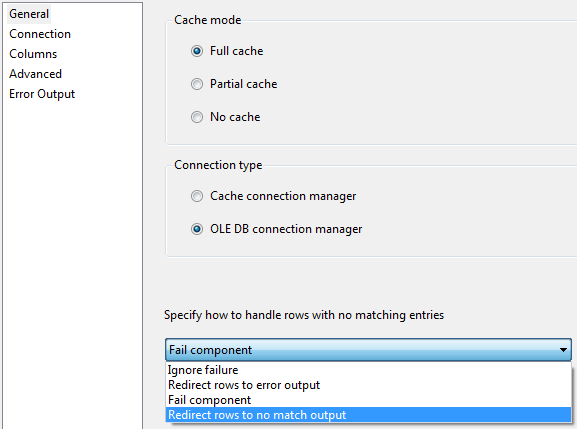


You will get the above error result. Why?

Because the book B and C cannot find its corresponding record in the lookup table and by default this situation is set as errors.

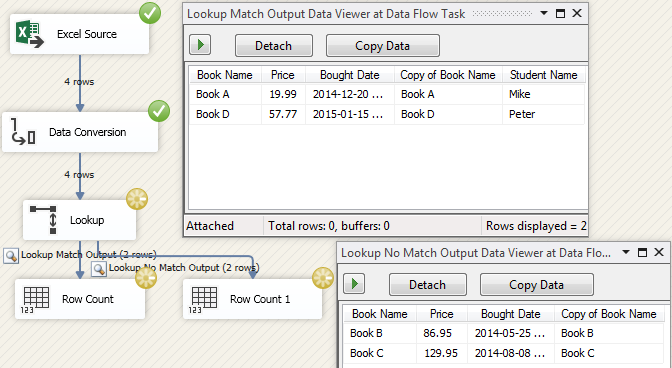
Let's change the setting.

1. Stop the execution and right-click the Lookup and open its editor again. Then change the "Fail component" to "Redirect rows to no match output".



We'll keep the Cash mode as its default "Full cache". This setting will cache the lookup query result in memory to make it run faster on the next call.

1. Click OK button and drag the "Lookup No Match Output" to the second Row Count. At last, enable the Data Viewer on the 2 outputs and run the package again.

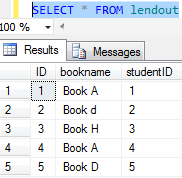
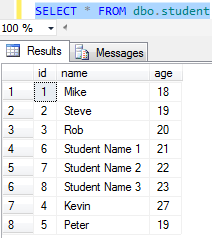


The first record of "Book A" was chosen as the matched result because the lookup transform always gets the first matched record in the lookup query. "Book d" was not chosen because the Lookup transformation are case sensitive. That means "Book d" and "Book D" are different in the eyes of the transform.

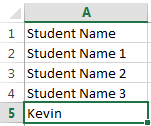
# SSIS OLE DB Command Transformation

The OLE DB Command transformation is used to run an SQL statement for each row of its input data flow to insert, update or delete records in a database table. The transform has one input, one output and one error output. Generally speaking, its input data flow is always defined as arguments to feed the parameters defined in the SQL statement of the transform.

Let's see an example, assume several students were graduated from the college and their borrowed books were returned. It's time to remove their records in lendout table and student table in Test database. The contents of the 2 tables look like below.

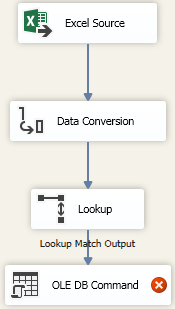
 

Normally we'll use student ID as the key to remove records but in this example to make things clear, I'll use student name as a key and assume there is no student with the same name. So an excel file graduate.xlsx in folder C:\SSIS was created below.

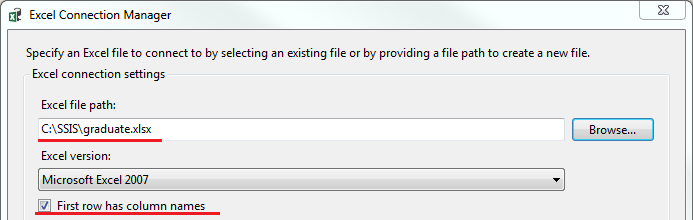


Now follow the step by step to the create a new package to use the OLE DB Command transformation.

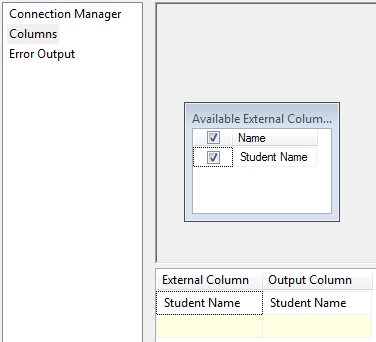
1. Open the project LearnSSIS2 which was created in the previous section in Visual Studio 2012.
2. Make a copy of the package Lookup.dtsx and rename the copy to OLEDBCommand.dtsx. Then open the package and click "Data Flow" to open it editing surface.
3. The Input Output Selection dialogue box will be popped up, we'll choose "Lookup Match Output" to link the OLE DB Command Input. Then Click OK.



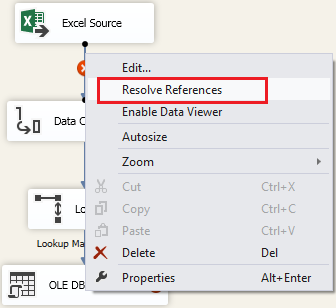
1. Right-click Excel Connection Manager and choose "Edit..." to open its editor and choose the excel fil to C:\SSIS\graduate.xlsx, then click OK button.



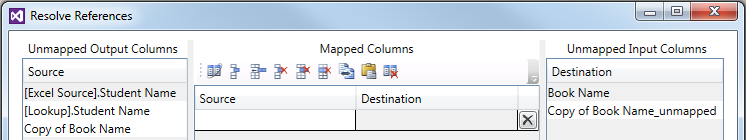
1. Right-click Excel Source and choose "Edit..." to open its editor and click "Columns" to see the columns were changed already.



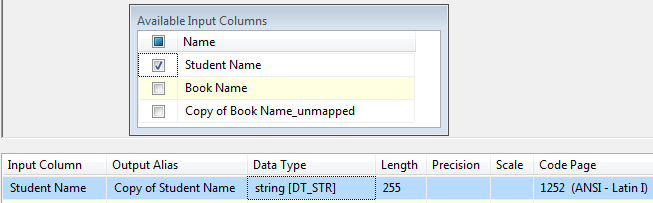
1. Click OK button. You will see a red cross shown up on the output of Excel Source. Then right-click the output and choose "Resolve References".



1. Remove all the rows in the middle and click OK.

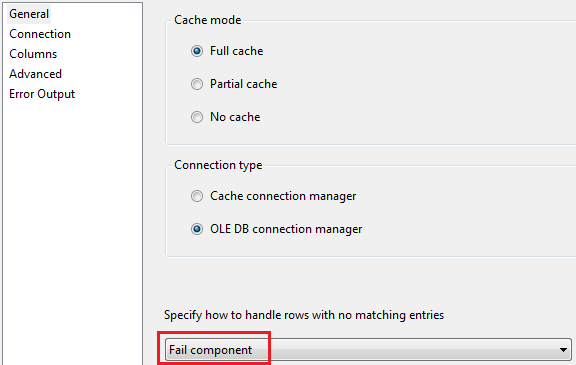


1. Right-click "Data Conversion" and choose "Edit..." to open its editor and change it as follows.

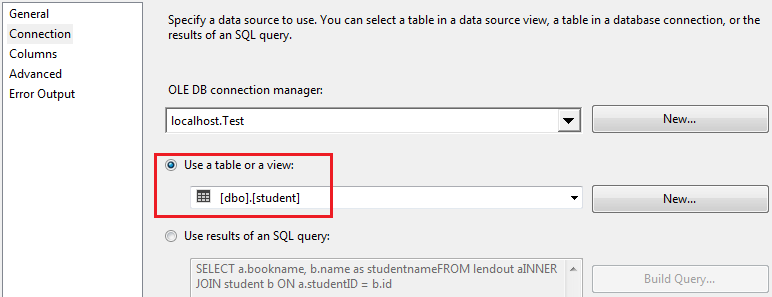


The "Student Name" was converted to a new column "Copy of Student Name" with string type.

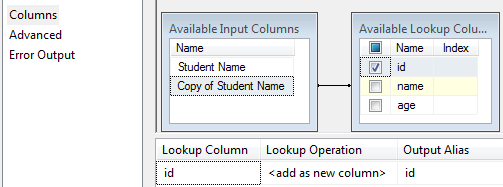
1. Click OK. Right-click Lookup transform and open it editor and change the "Redirect rows to no match output" to "Fail component" in the General tab.



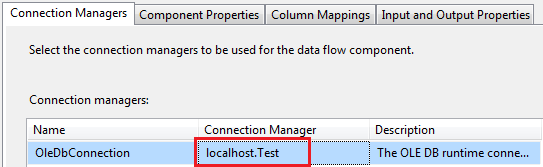
1. Click Connection tab and click "Use a table or a view" and choose the student table.



1. Click Columns tab, Link the "Copy of Student Name" to the "name" in the lookup dataset and add id as a new output column.



1. Click OK. Right-click OLE DB Command and choose "Edit..." to open its advanced editor. Then select localhost.Test as the Connection Manager.



1. Click Component Properties tab and select SqlCommand and click "..." to input the SQL command like below.

DELETE FROM lendout

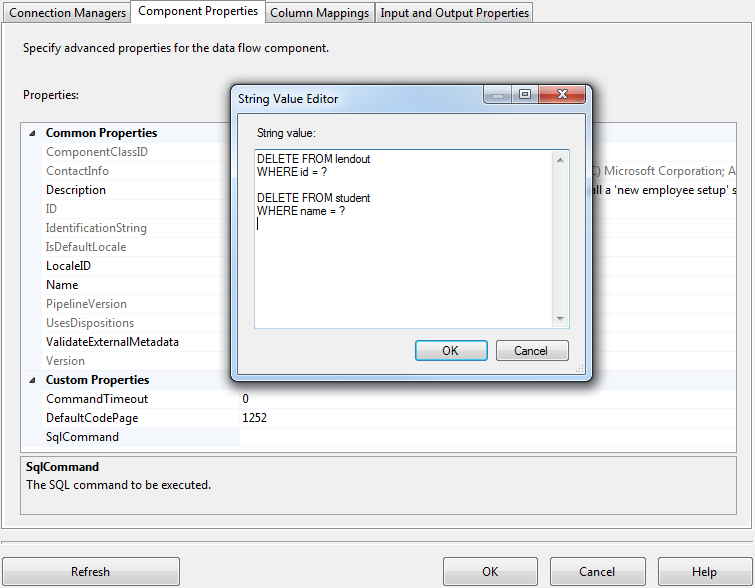
WHERE id = ?

DELETE FROM student

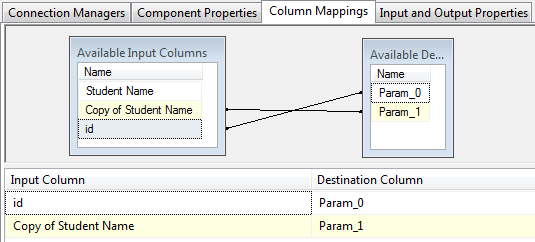
WHERE name = ?

The question marks stand for parameters and their names are Param\_0, Param\_1 and so on. In the following steps, we'll map them to the input columns.

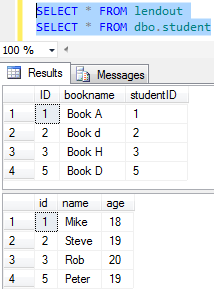
1. Click OK button. Then click refresh button to let system know how many parameters are defined.



1. Click "Column Mappings" tab and in the Input Column, choose id for Param\_0 and "Copy of Student Name" for Param\_1.



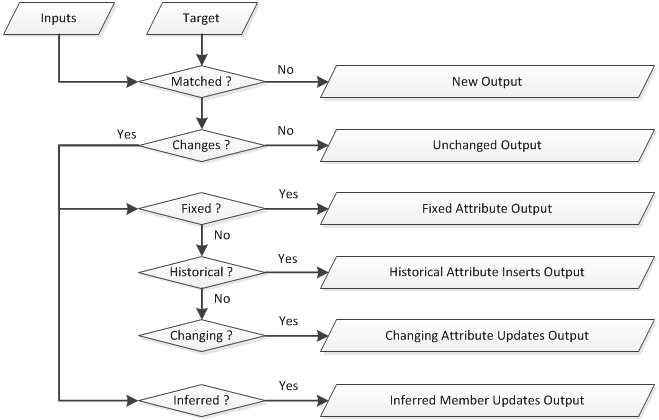
1. Click OK button and run the package. After execution, click "Stop Debugging" and check the database Test.



The graduate students' records were removed from the 2 tables.

# SSIS Slowly Changing Dimension Transformation

The Slowly Changing Dimension transformation is used to insert or update records in a table based on the business keys defined in the transform. It is the most powerful and complicated transform in a Data Flow Task and broadly used to change records in tables, especially in data warehouse dimension tables. Right now it only supports to connect to SQL Server and it has one input and up to six outputs without an error output. Let's see the diagram below to understand how it works.

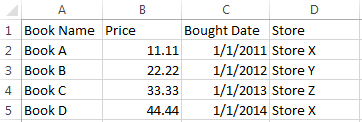


1. At run time, the Slowly Changing Dimension transformation will check if the key columns of its inputs match the key columns of the target table.
2. If not matched, this means the row of the input is a new row and will go to the New Output.
3. If matched, then check if at least one column is changed or not.
4. If not changed, the row of the input will go to Unchanged Output.
5. If changed, then check the change type of the column. There are 3 change types of each column and each type has different action and output. The types are summarized in a table below.

|  |  |  |
| --- | --- | --- |
| **Change Type** | **Description** | **Output** |
| Fixed attribute | Indicate the column of the target table cannot be changed. | Fixed Attribute Output |
| Changing attribute | Indicate the column of the target table will be changed. | Changing Attribute Updates Output |
| Historical attribute | Create a new row in the target table instead of changing the column. | Historical Attribute Inserts Output |

1. If the inferred member support is enabled, the Inferred Member Updates Output will be generated and is used to change the inferred member record in the dimension table.

Before create a sample package, let's see what the source excel file C:\SSIS\Books\_scd.xlsx looks like.



And we'll create a table to received the source data in database Test.

DELETE FROM lendout

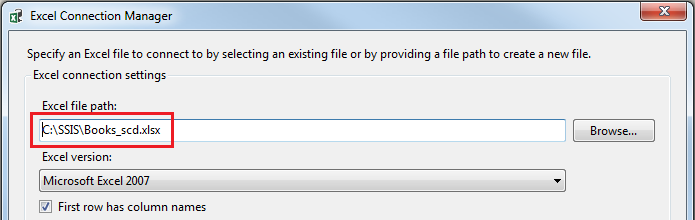
WHERE id = ?

DELETE FROM student

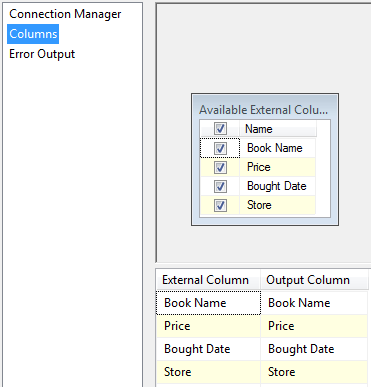
WHERE name = ?

Now follow the steps below to create the package.

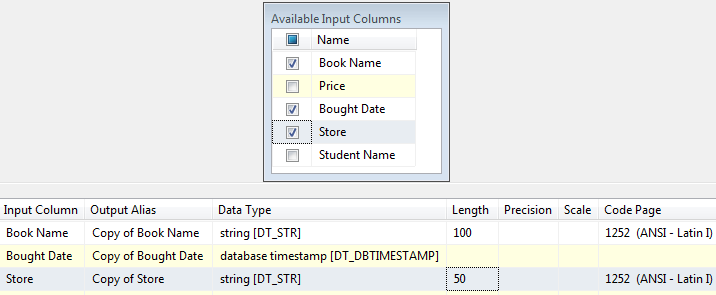
1. Open the project LearnSSIS2 which we created before and copy the package OLEDBCommand.dtsx and paste in the SSIS Packages folder in Solution Explorer. Then rename the package to SlowlyChangingDimension.dtsx.
2. Open the new package and click Data Flow tab, then remove the Lookup and OLE DB Command transforms.
3. Double-click the "Excel Connection Manager" to open its setting window and change the Excel file path to C:\SSIS\Books\_scd.xlsx and click OK button.



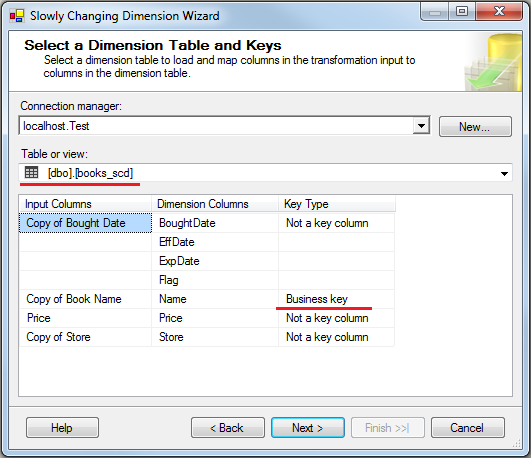
1. Double-click Excel Source to open the editor and click "Columns" tab to check the source data columns.



1. Click OK. Then double-click the Data Conversion transform to open its editor and make the settings as shown below.

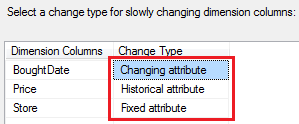


1. Click OK. Drag & drop the Slowly Changing Dimension (SCD) from SSIS Toolbox to the edit surface and link the Data Conversion output to it.
2. Double-click the SCD transform to open the Slowly Changing Dimension Wizard and click Next button to skip the welcome page.
3. Make the setting below and then click Next.



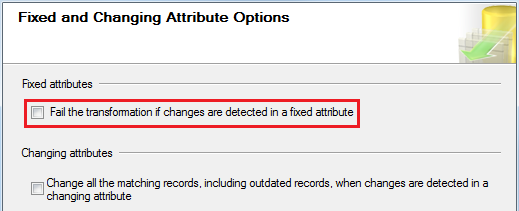
Pay attention: The "Copy of Book Name" is chosen as the Business key.

1. Set the "BoughtDate" to "Changing attribute", "Price" to "Historical attribute" and "Store" to "Fixed attribute".

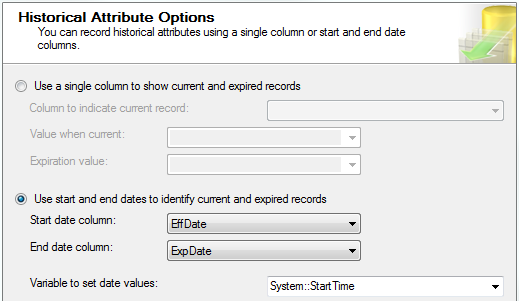


We set the 3 columns in different types to see what will be happened later in the result.

1. click Next. Uncheck the "Fail the transformation if changes are detected in a fixed attribute".

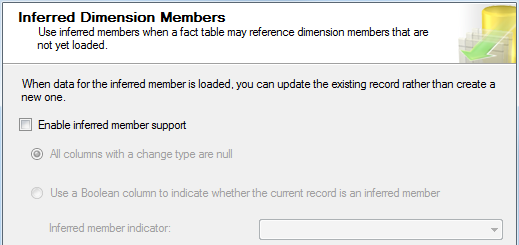


1. Click Next. We'll set the Historical Attribute Options as shown below.



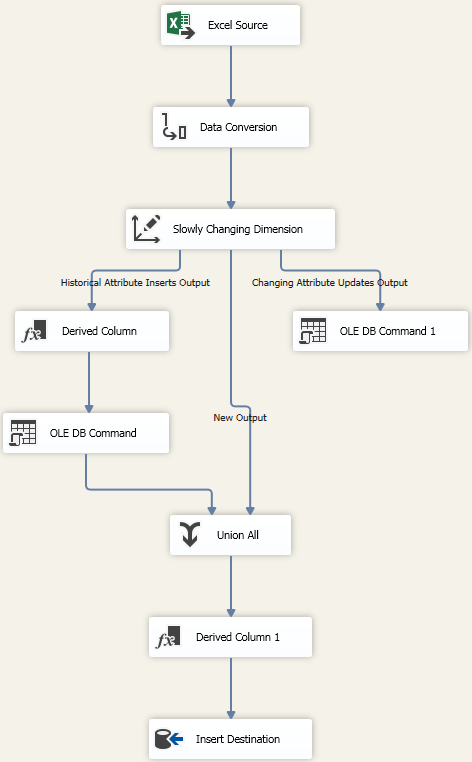
This option is displayed because we set the "Price" to "Historical attribute" in step 9.

1. Click Next. Uncheck "Enable inferred member support".

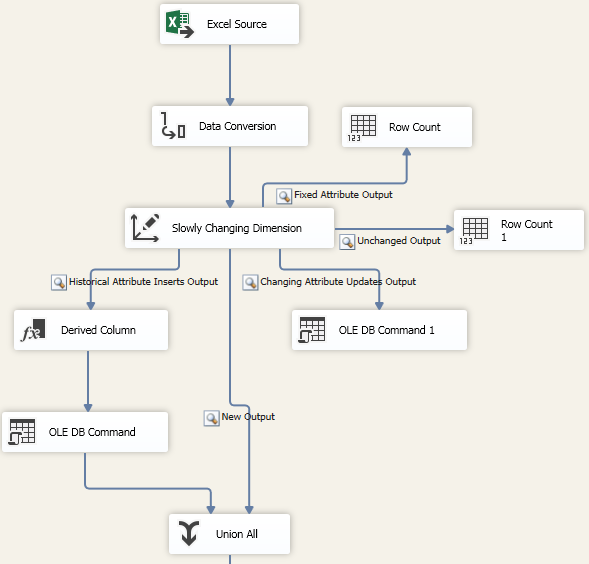


We'll not test the Inferred Member Updates Output in this example, so we just unchecked it here.

1. Click Next. Then click finish button. Then the New Output, Historical Attribute Inserts Output and Changing Attribute Updates Output are created automatically.



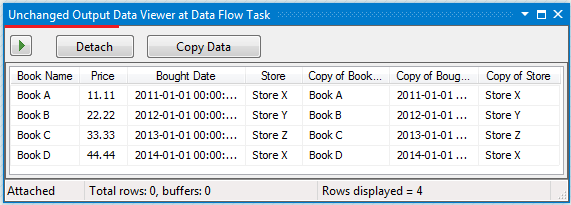
1. Drag & drop the 2 Row Counts to the edit surface and create & assign 2 variables to them.
2. Link the Fixed Attribute Output and Unchanged Output of the SCD transform to the 2 Row Counts. Then enable the data viewers on all the outputs.



1. Run the package.

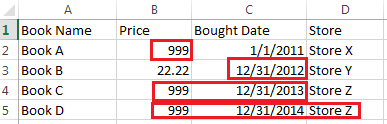
You will see all the 4 records go to the New Output. Because there is nothing in the table.

1. Run the package at the second time.



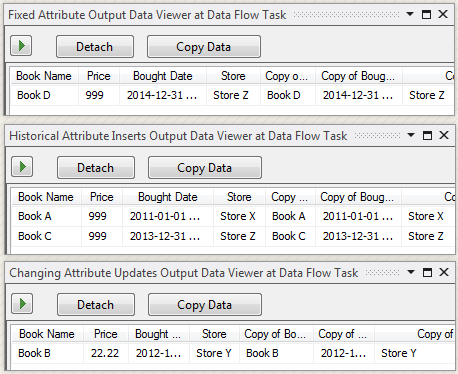
The 4 records are going to Unchanged Output.

1. Change the source data as follows.



The red squares indicates the changed data.

1. Run the package again.



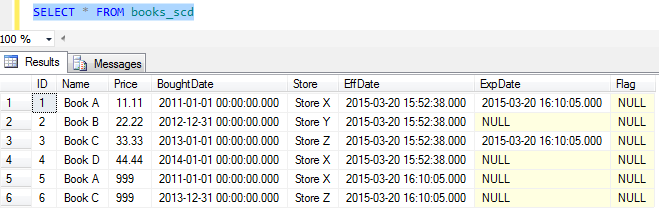
Book D goes to Fixed Attribute Output because the data in Store column was changed.

Book A, C go to Historical Attribute Output because the Price columns were changed.

Book B goes to Changing Attribute Updates Output because only the data in BoughtDate column was changed.

|  |
| --- |
| **Note Note** |
| Fixed Attribute Output has the highest priority to have the output data, then Historical Attribute Output and at last Changing Attribute Updates Output. |

Let's check the data in Test database.



Apparently, the record 1 and 3 are expired records and the expired dates are set up.

I'm going to leave the reader to change the step 11 to "Use a single column to show current and expired records" by using the Flag field in the table.

The following SQL statement can be used to remove all the records in the table books\_scd and restart running the package.

TRUNCATE TABLE dbo.books\_scd