Top 10 Common Transformations in SSIS

* **August 22, 2013**
* **By [Anoop Kumar](http://www.developer.com/author/Anoop-Kumar-20848610.htm)**
* [**Bio »**](javascript:void(0))
* [**Send Email »**](http://www.developer.com/feedback/db/top-10-common-transformations-in-ssis.html)
* [**More Articles »**](http://www.developer.com/author/Anoop-Kumar-20848610.htm)

[**Tweet**](https://twitter.com/share)

**Transformation in SSIS**

SQL Server Integration Services (SSIS) has transformations, which are key components to the Data Flow, that transform the data to a desired format as data moves from one step to another step. Transformation in SSIS is all done in-memory; after adding a transformation the data is altered and passed down the path in the Data Flow.

In SSIS, transformations are available in two main categories--Synchronous and Asynchronous. During ETL design it’s recommended to use all Synchronous transformation components.

**http://www.developer.com/images/pdfImage.jpg**

**10 Things You Need to Know About Deploying Microsoft� Office 365**

[**Download Now**](http://o1.qnsr.com/cgi/r?WT.qs_dlk=V0Z9vQrIZ70AAH2aatQAAAAM;;n=203;c=1397070;s=9487;x=7936;f=201309171321110;u=j;z=TIMESTAMP;cfurl=http%3A%2F%2Fwww.developer.com%2Fdb%2Ftop-10-common-transformations-in-ssis.html;k=http://assetform.developer.com/controller?asset=206471410&srvid=95910&vkey=9974310&io=222336&BLUID=2016052521382244016764001&qset=CONTACTFORM_HQB&formHQB=y&domain=www.developer.com)

Synchronous are components like the Conditional Split or Derived Column Transformation where rows flow into memory buffers in the transformation and the same buffers come out. No rows are held and characteristically these transformations perform very quickly with marginal impact to Data Flow.

Asynchronous transformation has two types, fully blocking and partial blocking. Partial blocking transformation is that transformation which creates new memory buffers for the output of the transformation than what come into the transformation, like Union All Transformation; fully blocking transformations also require a new memory buffer similar to partial blocking. Asynchronous transformations additionally cause a full block of the data like Sort and Aggregate transformations.

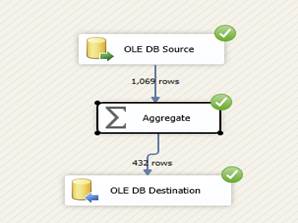
* [**Post a comment**](http://www.developer.com/db/top-10-common-transformations-in-ssis.html#comment_form)
* [**Email Article**](http://www.developer.com/e-mail/db/top-10-common-transformations-in-ssis.html)
* [**Print Article**](http://www.developer.com/print/db/top-10-common-transformations-in-ssis.html)
* [**Share Articles**](http://www.developer.com/db/top-10-common-transformations-in-ssis.html)

Let’s take a quick deep dive with some important SSIS transformations under both categories, Synchronous and Asynchronous, which can be useful to manage and transform data during ETL execution.

**Aggregate**

An Asynchronous full blocking transformation, Aggregate transformation allows to aggregate data from Data Flow to apply certain T-SQL functions that are done in a GROUP BY statement.

Data Flow task design for Aggregate:

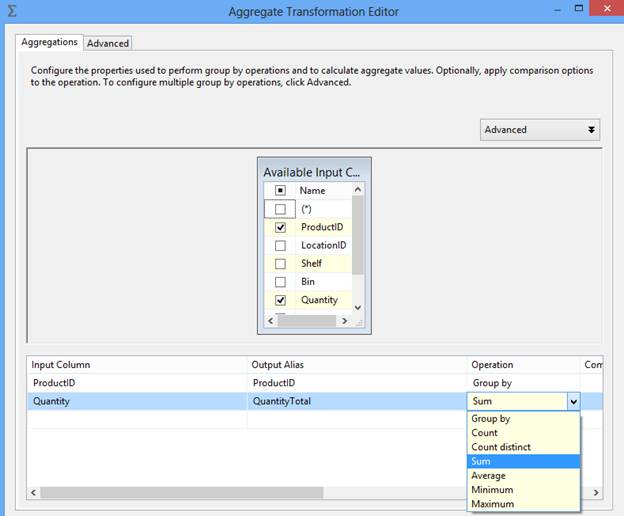
  
Aggregate Data Flow Task Design

— Advertisement —

**Related Articles**

* [Top 10 Methods to Improve ETL Performance Using SSIS](http://www.developer.com/db/top-10-methods-to-improve-etl-performance-using-ssis.html)
* [Best Practices: ETL Development for Data Warehouse Projects](http://www.developer.com/db/best-practices-etl-development-for-data-warehouse-projects.html)

Settings in Aggregate Transformation with all aggregate options:

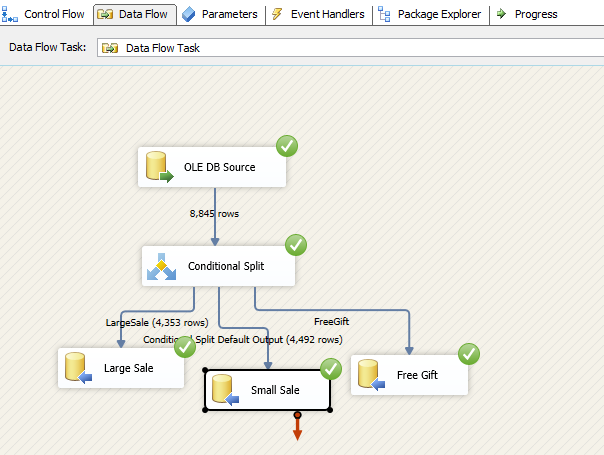
  
Aggregate Transformation Editor

In the above example we have applied SUM aggregation but Aggregation transformation provides other options to aggregate data like Count, Count distinct, Average, Minimum and Maximum.

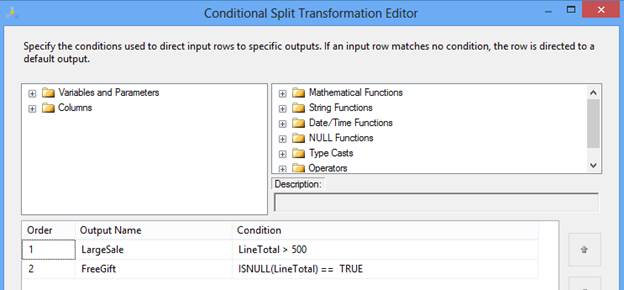
**Conditional Split**

Synchronous transformation, allows you to send the data from a single data path to various outputs or paths based on conditions that use the SSIS expressions.

Data flow task design for Conditional Split:

  
Conditional Split Data Flow Task Design

Conditional Split transformation settings:

  
Conditional Split Transformation Editor

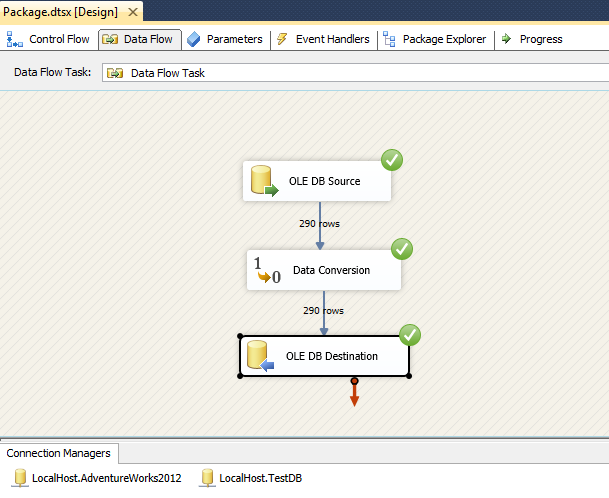
In the above example, we are splitting input records based on total order cost. If cost is more than 500, the record will be considered as part of a large sale. If LineTotal is NULL, we are assuming it's a free gift and no cost is associated with it. The rest we can consider part of small sale, in current implementation it is the default output of Conditional Split transformation.

After execution of DFT the data will move in three different destinations as per ETL design.

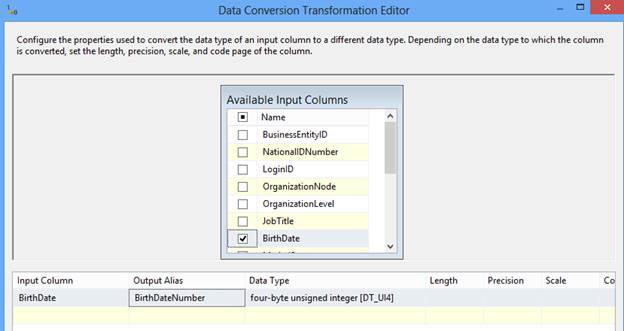
**Data Conversion**

Synchronous transformation is used for data conversion. It is a similar function to the Convert or Cast functions in T-SQL. It is a very useful transformation if we are pulling same data from multiple sources.

Data flow task design for Data conversion:

  
Data Conversion Data Flow Task Design

Data Conversion transformation settings:

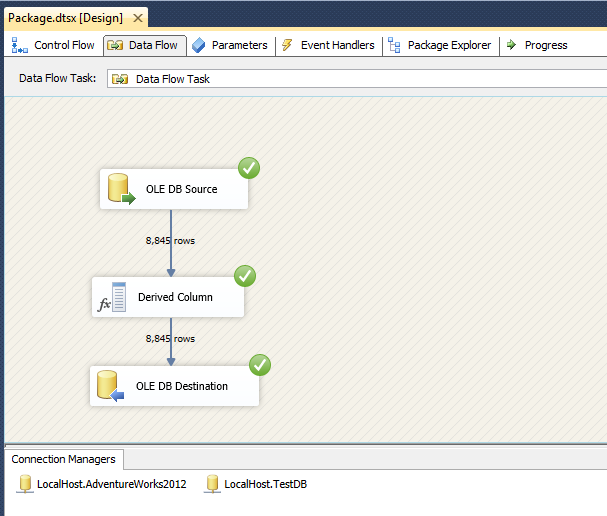
  
Data Conversion Transformation Editor

In this example we converted the BirthDate column of the datetime data type in another column BirthDateNumber of Integer data type.

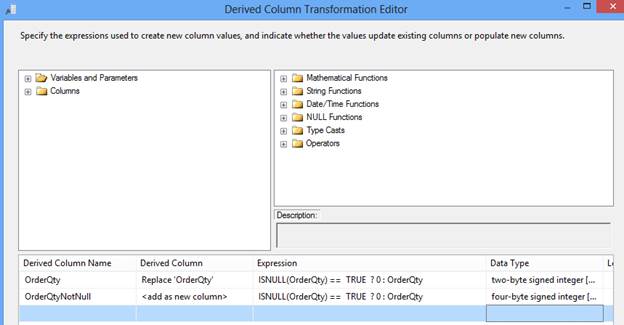
**Derived Column**

Synchronous transformation, this transformation creates a new column that is derived from the output of another column. This transformation provides you two options; either you can create a new column as a derived column or replace the existing column with a new derived column.

Data flow task design for Derived column:

  
Derived Column Data Flow Task Design

Derived column transformation settings:

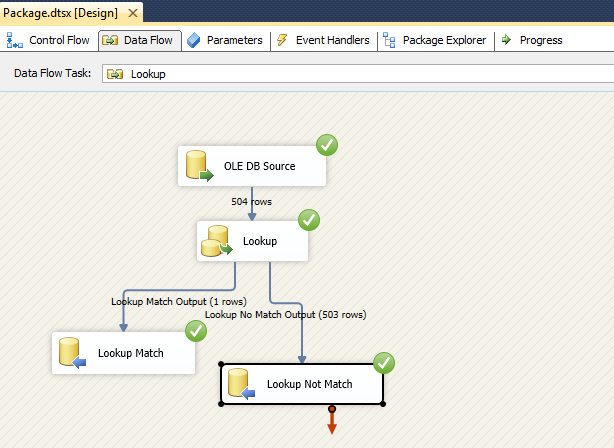
  
Derived Column Transformation Editor

In this example, in the first row, check the if OrderQty value is NULL then update with 0 and in the second row apply the same operation as in the first row; the only difference is it will create one new column OrderQtyNotNull in the output. So, with the help of Derived Column transformation you can either update an existing column value or introduce a new column in the output.

**Lookup**

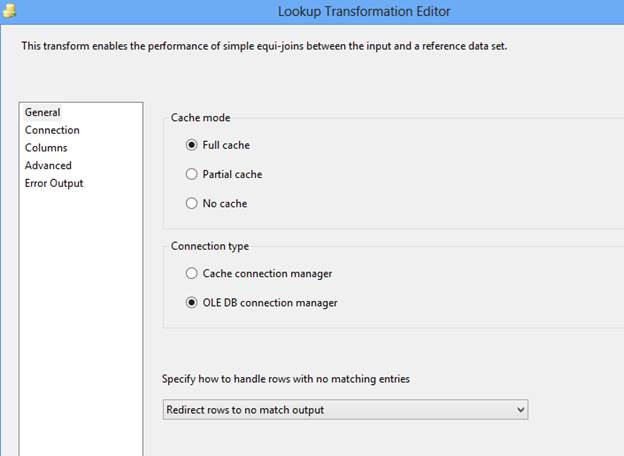
Synchronous transformation, allows you to perform an equi-join between values in the transformation input and values in the reference dataset similar to T-SQL. This transformation is used to join two datasets at a time.  To join more than two datasets we need to put multiple Lookup transformations, similar to a T-SQL join condition.

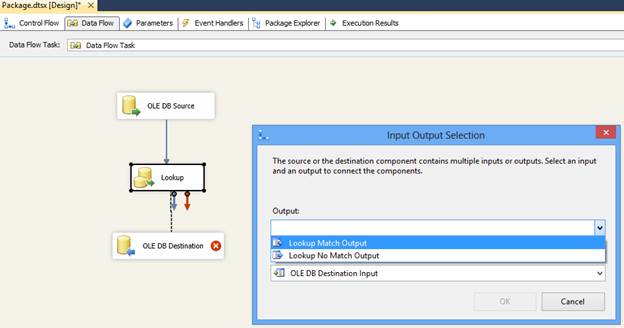
Data Flow task design for Lookup:

  
Lookup Data Flow Task Design

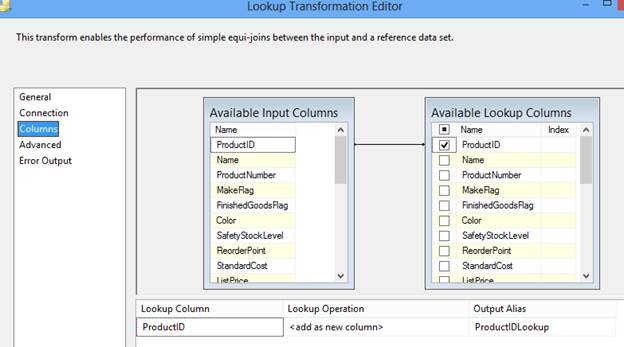
Lookup transformation settings:

If there is no matching entry in the reference dataset, no join occurs. By default, the Lookup transformation treats rows without matching entries as errors. However, it can configure the Lookup transformation to redirect such rows to a no match output as shown in the images below:

  
Lookup Transformation Editor

  
Input Output Selection

The join can be a composite join, which means that multiple columns can be used in the join in the transformation input to columns in the reference dataset; for simplification we used only one column. Refer to the below image:

  
Lookup Transformation Editor

In above image, you can observe in the Lookup Operation that we specified “<add as new column>”; its mean values from the reference dataset are added as a new column to the transformation output. For example, the Lookup transformation can extract the ProductID details from a table using a value from an input column, and then add the ProductIDLookup to the transformation output. The values from the reference table can replace column values or can be added to new columns.

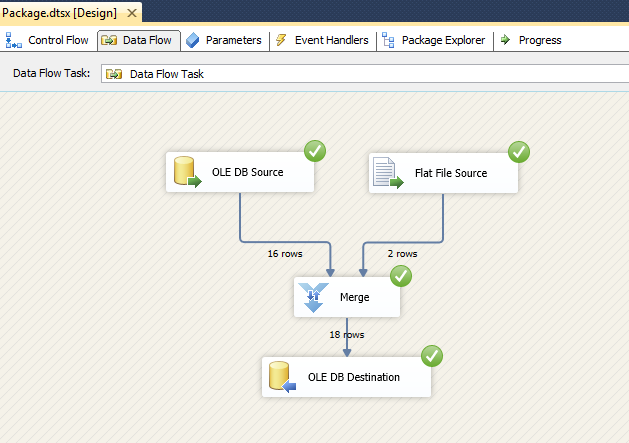
Lookup transformations provides several modes of operations, Full cache, Partial cache or No cache, that allows a trade-off between performance and resource usage.

You can refer to [MSDN](http://technet.microsoft.com/en-us/library/ms141821.aspx) to learn more interesting facts about Lookup transformation.

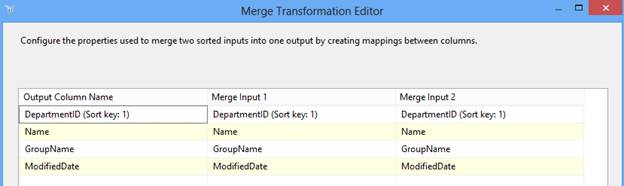
**Merge**

An Asynchronous partial blocking transformation merges two sorted data sets into a single dataset. This transformation is very useful when during ETL its needs to merge data from two different data sources. Merge transformation can’t merge a column that has a numeric data type with a column that has a character data type.

Data Flow task design for Merge:

  
Merge Data Flow Task Design

Merge transformation settings:

  
Merge Transformation Editor

In the above example, we are merging data from two sources; OLEDB and Flat File. The Merge transformation automatically maps columns that have the same metadata. You can then manually map other columns that have compatible data types.

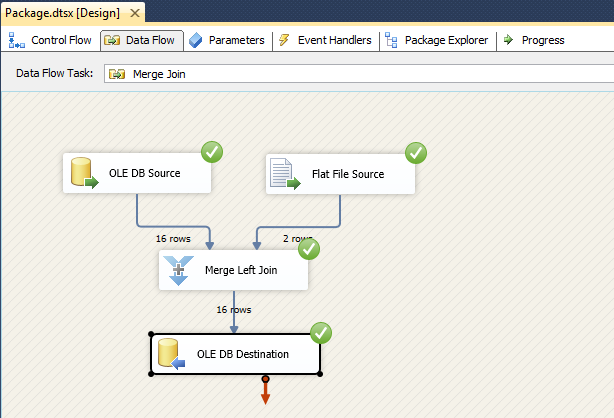
This transformation has two inputs and one output. It does not support an error output.

**Merge Join**

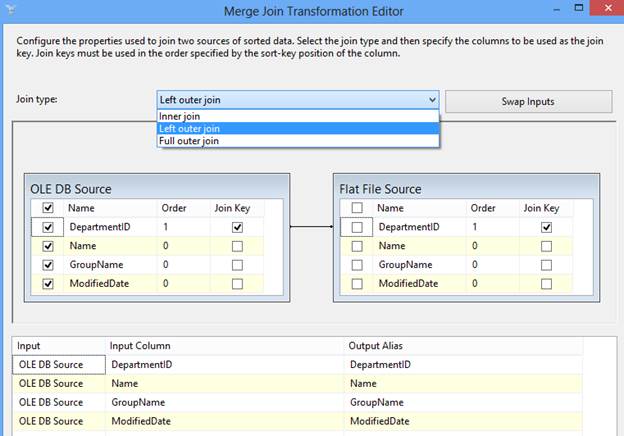
An Asynchronous partial blocking transformation, allows joining data from two sorted datasets using a FULL, LEFT, or INNER join.

It also has two inputs and one output and like Merge transformation, does not support an error output.

Data Flow task design for Merge Join:

  
Merge Join Data Flow Task Design

Merge Join transformation settings:

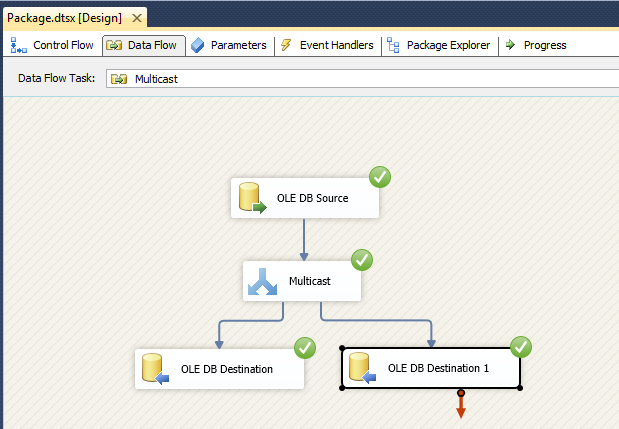
  
Merge Join Transformation Editor

In above example, we merged data from two different sources; OLEDB and Flat File, applying a Left outer join on DepartmentID.

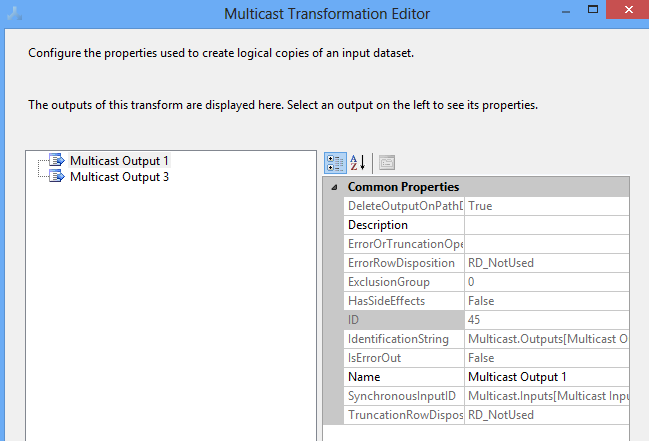
**Multicast**

Synchronous transformation allows you to distribute its input to one or more outputs. This transformation is similar to the Conditional Split transformation. Both transformations direct an input to multiple outputs. The difference between the two is that the Multicast transformation directs every row to every output, and the Conditional Split directs a row to a single output.

Data Flow task design for Multicast:

  
Multicast Data Flow Task Design

Multicast transformation settings:

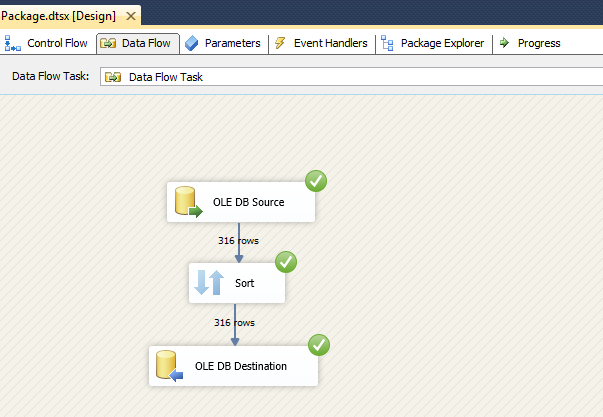
  
Multicast Transformation Editor

In the above example, we are distributing log data to two different destinations.

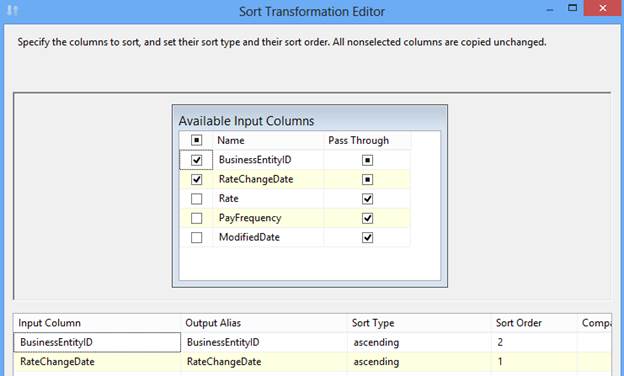
**Sort**

An Asynchronous full blocking transformation allows sort or arrange input data in ascending or descending order and copies the sorted data to the transformation output. You can apply multiple sorts to an input; the column with the lowest number is sorted first, the sort column with the second lowest number is sorted next.

Data Flow task design for Sort:

  
Sort Data Flow Task Design

Sort transformation settings:

  
Sort Transformation Editor

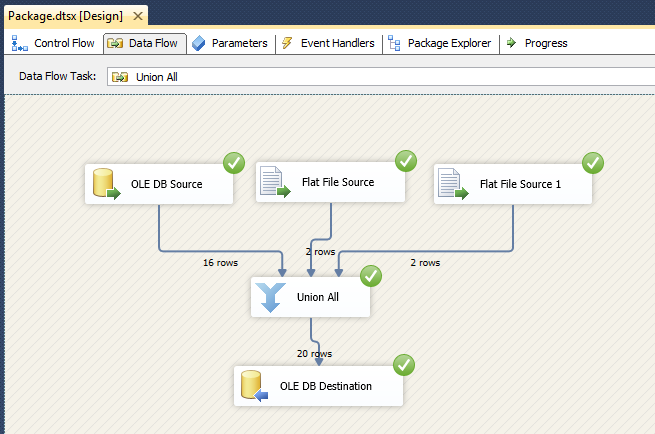
In above example, we arranged input data in ascending order of RateChangeDate first and BusinessEntityID column second.

Sort transformation has one input and one output. It does not support error outputs.

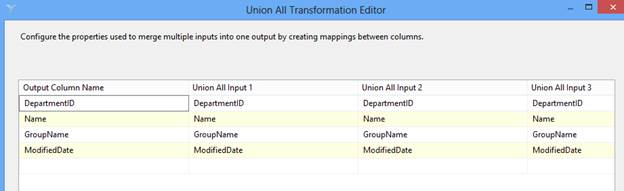
**Union All**

An Asynchronous partial blocking transformation, allows you to combine multiple (more than two) input and produce one output. Its add inputs to transformation output one after the other and doesn’t sort the data.

Data Flow task design of Union All:

  
Union All Data Flow Task Design

Union All transformation settings:

  
Union All Transformation Editor

In above example, we used three sources as input and combine all using the Union All transformation before inserting into  the destination. Here, we took two different type of sources; OLEDB and Flat File.

Below is the list of transformations under both categories, which will help you to design ETL and data warehouse system.

|  |  |  |
| --- | --- | --- |
| Synchronous Transformations | Asynchronous Transformations | |
| Partial blocking | Fully blocking |
| Audit | Data Mining Query | Aggregate |
| Character Map | Merge | Fuzzy Grouping |
| Conditional Split | Merge Join | Fuzzy Lookup |
| Copy Column | Pivot | Row Sampling |
| Data Conversion | Term Lookup | Sort |
| Derived Column | Union All | Term Extraction |
| Export Column | Unpivot |  |
| Import Column |  |  |
| Lookup |  |  |
| Multicast |  |  |
| OLE DB Command |  |  |
| Percent Sampling |  |  |
| Row Count |  |  |
| Script Component |  |  |
| Slowly Changing Dimension |  |  |

**Conclusion**

In this article we explored the 10 most common and important transformations in one place. These are very useful transformations to manage and change data in required format for a data warehouse. Also, we discussed why Asynchronous transformations should be avoided in ETL design.