Extension of a package is “**.DTSX**”, here ‘DTS’ stands for Data **Transformation Services** and ‘X’ stands for **XML**.

--Connection Manager = what data (Excel, Access, SQL server, Flat file) + where data (location of file). Connection manager just help to connect certain types of data files. A connection can be either use in a source adapter or a destination adapter.

--OLE DB

OLE DB (Object Linking and Embedding, Database), an API designed by Microsoft, allows accessing data from a variety of sources in a uniform manner. In SSIS, we have OLE DB source and destination. OLE DB source extracts data from a variety of OLE DB-compliant relational databases. The OLE DB destination loads data into a variety of databases that support this connection. The OLE DB such as SQL server, Access, Oracle.

--OLE DB source access mode

For OLE DB sources, there are several access mode to extract data. We can extract data from a table or a view, and we can extract data by writing a piece of SQL statement. We can even use variable to represent a table or view, or wrap up a SQL statement.

--OLE DB destination access mode

OLE DB destination allows multiple data access mode. Similar to the OLE DB source, in OLE DB destination, we can use table and view, and also we can use variable to represent this table or view. In addition, we have an option to choose **the fast loading function**. Besides that, it also support SQL statement, but we can’t use SQL statement in variables.

--Fast loading option

In OLE DB destination adapter and ODBC (2012 new) can insert data from data flow through bulk batches of data, instead of one row at a time. Options under fast load: keep identity, key nulls, table lock (default), check constraints (default), rows per batch, maximum insert commit size

--error output

--steps to configure the OLE DB adapter

Whenever to create a data adapter in SSIS, the first step is to create a connection manager. Except XML data the raw data, which doesn’t require in a connection manager. Generally you need to specify the provider and data source. For example, if you are working with access data, you need to the access database engine OLEDB provider and specify the file location. The default one is SQL server native client, which work with SQL server data and will ask you to specify the database name and the authentication mode to login the database. In the next step, you can configure the adapter in OLE DB source editor. You may specify the assess mode, you may view and select the columns to be used to the data flow. You can also specify properties by looking at its properties window.

Create OLE DB Source 🡪 Create connection manager (choose provider e. g. SQL server, Access, Oracle and choose file location, authentication) 🡪Configure OLE DB source adapter (select a table if SQL server, setting up property e. g. access mode, columns, error output)

Create OLE DB Source 🡪 Create connect manager 🡪 Configure OLE DB destination adapter (properties like fast load, and examine mappings, Error output)

Unicode vs ANSI

Flat file source read data from a text file.

--CODE PAGE

a code page is a table of values that describes the character set used for encoding a particular set of glyphs, usually combined with a number of control characters.

--Flat file data connection manager

Need to specify info code page, on data format (delimited, fixed width, or Ragged), test qualifier, header row delimiter {CR}{LF}, header rows to skip, whether columns in first row

--Format of a text file, delimited vs. fixed width vs. ragged right

Delimited format uses column and row delimiters to define columns and rows. For example, a CSV file is delimited by comma. Fixed width format uses width to define columns and rows. This format also includes a character for padding fields to their maximum width. For example, a fixed width file may have something like this. The first 10 characters are first column, the 10-20 characters are the second column. Ragged right format uses width to define all columns, except for the last column, which is delimited by the row delimiter.

--ODBC sources: extract data from ODBC supported database

--Configure ODBC sources: Access Mode (table or SQL), Fetch Mode (by row or batch)

--ODBC destination: bulk load data into ODBC supported database

--ODBC destination configuration: Load option (by row or batch)

--ODBC connection manager: choose ODBC provider, source file location

--ADO.Net Source and Destination: connect data through an ADO.net provider. Destination loads data into a variety of ADO.NET-compliant databases

--ADO.Net connection manager

--Configuration: Access Mode (table or view and SQL command)

--ADO NET source has one regular output and one error output

--Raw File Source: read raw data

--configuration: Access Mode (name of the file, or variable), no connection manager needed

--This source has one output. It does not support an error output.

--Raw File Usage: The Raw File destination is used to write intermediary results of partly processed data between package executions.

--Raw File destination: Written option (Create always, Create once, Append, Truncate and Append)

--XML source: reads xml file and transform xml data into table format

--The XML source supports use of a **XML Schema Definition (XSD) file** or **inline schemas** to translate the XML data into a tabular format.

--XML Schema describes the structure of an XML document.

--Access Mode: file location, file from variable, data from variable.

--SQL server destination: Used to bulk load data into a local SQL server

--use OLE DB connection

--default fast load access mode

--don’t support error output

EXCEL source compactable issues

Used to load data to SQL Server compact databases.

Uses SQL Server Compact connection manager to connect to a data source.

The SQL Server destination has one input. It does not support an error output.

SQL Server Compact 4.0 does not support SQL Server Integration Services (SSIS).

Starting from SQL Server Compact 4.0, SQL Server Compact does not support SQL Server Management Studio.

--Classify transformation: Row transformation, spilt and join transformation, Row set transformation, business intelligence transformation, Auditing Transformation

--Row transformation:

Character Map

Copy Column

Data Conversion

Derived Column,

OLEDB Command, etc.

--Split and Join Transformations

* + - Conditional Split and Multicast
    - Union All
    - Merge and Merge Join
    - Lookup
    - Cache Transform

--Row set Transformations

* + - Aggregate
    - Sort, etc.

--Business Intelligence Transformations

* + - Slowly Changing Dimension
    - Fuzzy Lookup and Fuzzy Grouping, etc.

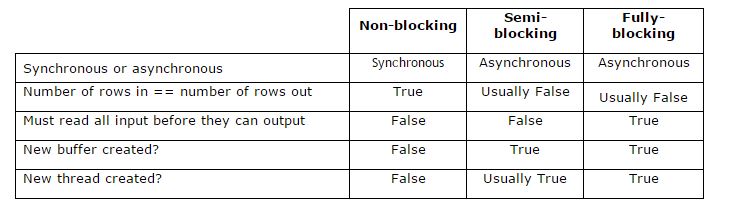
--Auditing Transformations

* + - Audit
    - Row Count

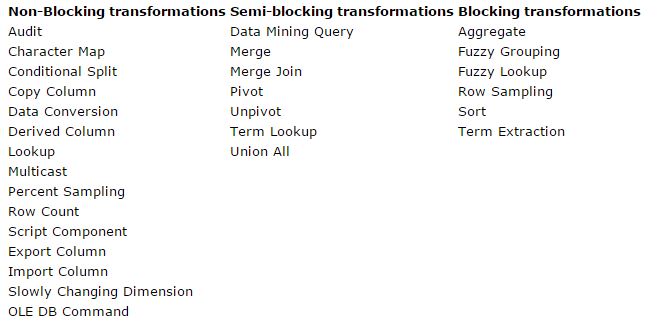
Two types of dataflow component

--Synchronous: The output of this component use the same buffer as input. The number of output record is the same as the number of input record. For example, a row by row based transformation is synchronous 🡪Non-blocking transformation

--Asynchronous: The output of this component uses a new buffer. The output can have more or less records than input. The transformation will have to wait other records. For example, an aggregate transformation will have to wait until all the record are read. 🡪 Semi-blocking and non-blocking transformation



--threshold



-- Explain non-blocking transformations, partially blocking transformation and blocking transformation along with examples.

Behind the scenes, the data flow engine uses a buffer-oriented architecture to efficiently load and manipulate datasets in memory. The benefit of this in-memory processing is that you do not need to physically copy and stage data at each step of the data integration. Rather, the data flow engine manipulates data as it is transferred from source to destination.

As this data flows through the pipeline, SSIS attempts to reuse data from prior buffers as much as possible when additional operations are performed. How buffers are used and reused depend on the type of transformations that you use in a pipeline.

**Row Transformations** - Row transformations either manipulate data or create new fields using the data that is available in that row. Examples of SSIS components that perform row transformations include Derived Column, Data Conversion, Multicast, and Lookup. While these components might create new columns, row transformations do not create any additional records. Because each output row has a 1:1 relationship with an input row, row transformations are also known as synchronous transformations. Row transformations have the advantage of reusing existing buffers and do not require data to be copied to a new buffer to complete the transformation.

**Partially blocking transformations** - Partially blocking transformations are often used to combine datasets. They tend to have multiple data inputs. As a result, their output may have the same, greater, or fewer records than the total number of input records. Since the number of input records will likely not match the number of output records, these transformations are also called asynchronous transformations. Examples of partially blocking transformation components available in SSIS include Merge, Merge Join, and Union All. With partially blocking transformations, the output of the transformation is copied into a new buffer and a new thread may be introduced into the data flow.

**Blocking transformations** - Blocking transformations must read and process all input records before creating any output records. Of all of the transformation types, these transformations perform the most work and can have the greatest impact on available resources. Example components in SSIS include Aggregate and Sort. Like partially blocking transformations, blocking transformations are also considered to be asynchronous. Similarly, when a blocking transformation is encountered in the data flow, a new buffer is created for its output and a new thread is introduced into the data flow.

Transformations are not the only components that can be categorized as synchronous or asynchronous. Sources are a special type of asynchronous component. For example, an RDBMS source component creates two types of buffers: one for the Success output and one for the Error output. By contrast, destinations are a special type of synchronous component. You will see the interactions of source and destinations components when you examine the Execution Trees of a package.

Transformations

--Data conversion: as its name suggest, it convert the data type of a column into another and then make converted column as a new column in output. For example, we can change the numeric data into date data type. Set the column length of string data and the precision and scale on numeric data. Specify a code page. This transformation has one input, one output, and one error output.

--Character Map: it applies string functions on the data and only work with string data type. For example, we can convert a string from lower case to upper case. You can choose to convert data in place or add the converted data in a new column. This transformation has one input, one output, and one error output.

--Copy columns: copies a column and add it as a new column in the transformation output. In other words, it will duplicate a column. It can create multiple copied for a column and copies for multiple column in one operation. This transformation has one input, one output. It does not support an error output.

--Derived columns: it create new column by apply function or expression on the existing columns. For example, we can apply a mathematical function to multiple column to create a derived result. We can concatenate string data of multiple columns. The result can be added as either a new row or a replacement of an original column. This transformation has one input, one regular output, and one error output.

--Sort: order data by a column or multiple columns. Each sort is identified by a numerical that determines the sort. A positive value means sorting by ascending order and negative means descending. The column with the smallest value will be sorted first. You also have option to remove the duplicate rows. This transformation has one input and output, it doesn’t come with error output

--Aggregate: used to perform aggregate operation on data. Common functions are sum, average, count, and max, minimum./ IsBig property can be used to handle very big values/ Settings: Keys or Key Scale for group by operation; CountDistinctKeys or CountDistinctScale for distinct count operation. One input, can be multiple output, not error output

--Union all: used to combine tables vertically. Columns data types must match. Multiple input, one output, not error output

--Merge: Combines two sorted data into one. It can only work on two data input and it requires both tables to be sorted, and the column data type should match. Two input, one output, and no error output

--Merge join: Combines two sorted data set by using joins. This include inner join, left join and full join. Two input one output, no error output.

--Conditional Spilt: it directs data into different output by evaluating again specified conditions. One input, multiple output and one error output

--Multi Cast: it copies data into multiple copies and sent them to different output. One input, multiple output and no error output

--Row count: it counts the number of rows passed through the data flow and store the final count into a predefined variable. The variable should have a scope of data flow task. The row count value will not be sent to the variable until the last row of the input has passed the transformation. One input one out, not error output.

--Audit: will include statistics on the system environment in which the package is running. One input and one out, not error output

--Lookup: It compares the input data with a reference data to see whether there is a match. It only support the cache connection manager and OLEDB connection manager. The input date will be divided into two group, the matched entries and the unmatched entries./ There are three cache mode for the reference table, depending memory usage. Full cached, all reference data are cached before look up. Partial lookup, only the reference data which is matched during the lookup will be cached. No Cached mean we don’t cache the reference date at all. And of course this is gives the worst performance, especially for large tables. Three output: matched, unmatched and error output/ Support all data types except for (DT\_R4, DT\_R8, DT\_Text, DT\_NText, or DT\_Image). /It is very useful in performing incremental load.

--Cache transform: it generate a reference dataset for look up operation. The data will write only unique rows the reference data will be mapped to a cache connection manager. It can be used with full cache mode only. In a single package, a cached connection manager will assess data from only one cache transform component. It is very useful in performing incremental load.

Fuzzy lookup transformation

--can be used to perform data cleaning tasks to standardizing data and correct data

--for each record in input data, it uses fuzzy match to return one or more closely matched rows in a reference table

--the reference data must be a table in SQL server database

--setting: maximum number of matches, token delimiters, similarity threshold

--similarity threshold: \_similarity: mathematics measure that indicates the level of similarity between values in input and output columns. \_confidence, a column that indicate the quality of match

--setting to optimize performance: Exhaustive, WarmCaches, MaxMemoryUsage

--this transformation has one input and one output

Fuzzy Group Transformation

-- Performs data cleaning by identifying duplicate rows and selecting a canonical row of data to use in standardizing the data.

--configuration: 1. Select the columns on which we want to find duplicates. 2. Select type of match, fuzzy or exact match.

--\_key\_in: a column that identifies each records in the input data

--\_key\_out: a column that identifies a group of duplicated rows.

--\_score: a value between 0 and 1 that indicates the similarity of the input row to the canonical row

-- FuzzyComparisonFlags property to specify how the transformation compare string data.

-- **ExactFuzzy** property specifies whether the transformation performs a fuzzy match or an exact match

-- To optimize the performance, **Exhaustive** property can be set to true.

-- Specify **MaxMemoryUsage** value in MB, this transformation can use.

DQS Cleansing:

--Perform data quality tasks including correction, enrichment, standardization and de-duplication.

-- The DQS Cleansing transformation uses Data Quality Services (DQS) to correct data from a connected data source. It applies approved rules that were created for the connected data source or a similar data source.

--Unlike fuzzy lookup and grouping, DQS depends on domain specific knowledge bases

OLE DB command transformation: runs a SQL statement for each row in a dataflow.

--configuration

Provide the SQL statement that the transformation runs for each row

Specify the number of seconds before the SQL statement times out

Specify the default code page

Map the parameters

--One input, one output and one error output

SCD transformation

--The SCD transformation helps to update and insert operations in the dimension tables of a data warehouse.

-- It requires at least one business key column. (References to the primary key of table in database)

-- Can be configured either by Time Stamp or Flag column.

-- Four types of changes

1. Changing attribute: which update or overwrite existing records in dimension.

2. Historical attribute: create new records instead of updating existing ones. The only thing can be changed on historical record the column that indicates when the record is current or expired.

3. Fixed Attribute: this indicates the column value must not change

4. Inferred member indicates that the row is an inferred member record in the dimension table

The Slowly Changing Dimension transformation supports four types of changes: **changing attribute, historical attribute, fixed attribute, and inferred member**.

* Changing attribute changes overwrite existing records. This kind of change is equivalent to a Type 1 change. The Slowly Changing Dimension transformation directs these rows to an output named **Changing Attributes Updates Output**.
* Historical attribute changes create new records instead of updating existing ones. The only change that is permitted in an existing record is an update to a column that indicates whether the record is current or expired. This kind of change is equivalent to a Type 2 change. The Slowly Changing Dimension transformation directs these rows to two outputs: **Historical Attribute Inserts Output** and **New Output**.
* Fixed attribute changes indicate the column value must not change. The Slowly Changing Dimension transformation detects changes and can direct the rows with changes to an output named **Fixed Attribute Output**.
* Inferred member indicates that the row is an inferred member record in the dimension table. An inferred member exists when a fact table references a dimension member that is not yet loaded. A minimal inferred-member record is created in anticipation of relevant dimension data, which is provided in a subsequent loading of the dimension data. The Slowly Changing Dimension transformation directs these rows to an output named **Inferred Member Updates**. When data for the inferred member is loaded, you can update the existing record rather than create a new one.