**There are three (3) different types of SSIS transformations available in the SSIS Toolbox** for the DataFlow of our SSIS packages. For highest performance, it’s recommended to use as few semi-blocking and fully blocking SSIS transformations as possible. These transformations require SSIS to cache some or all the results before they are passed on to the destination(s).

As an SSIS developer, your job is to create packages using as few blocking or semi-blocking SSIS transformations as possible. This can be accomplished by leaning on data sources to do some of the transformation work, such as pulling data from the source pre-sorted. This will always outperform using the blocking Sort transformation, for example. If this is not possible, you could also move the data to a temporary landing destination before moving it into your data warehouses.

Non-Blocking Synchronous Streaming Transformations (Fastest)

Non-Blocking Synchronous Row-Based Transformations (Fast)

Semi-Blocking Asynchronous Transformations (Medium)

Fully Blocking Asynchronous Transformations (Slowest)

**Synchronous (non-blocking) transformations** always offer the highest performance. They do not change the shape of data. You can think of it as one row in and then one row out. Note that most transformation are non-blocking.

**Synchronous transformations** are either stream-based or row-based. Streaming transformations are calculated in memory and do not require any data from outside resources to transform the data. These are the fastest transformations around. Row-based transformations run a little bit slower because they require calling a service or looking up data from another source to calculate their values. They still move data one row at a time but the call to the resource outside of the SSIS package costs time.

**Semi-Blocking Asynchronous Transformations** require a subset of the data to be collected before they can be sent to the destination(s). The shape of the data can change. A subtotal or sampling of data may be extracted from the source(s).

**Fully Blocking Asynchronous Transformations** are the slowest transformations. They require all the data to be pulled from the source(s) before they can be sent to the destination(s). All source data must be loaded into memory first – ouch! As much as we should try to avoid these, they can sometimes still be required, such as sorting data pulled from a flat file source.

If there is more data than memory available, it will use the %TEMP% directory to cache some of the data. If you want to use a different location (such as a fast drive), you can set the BufferTempStoragePath property of the Data Flow Task to point to a different folder location. Be sure packages running these transformations are running on x64 platforms with plenty of memory so it won’t be cached to the hard drive.

Note that some of the SSIS transformations (such as the Business Intelligence transformations) require the Enterprise edition of SQL Server to be used. However, many can be replaced with transformations available from the community or from third parties. Here’s one source: http://ssisctc.codeplex.com/

**Here is a list of the different types of transformations – enjoy!**

**Non-Blocking Synchronous Streaming Transformations (Fastest)**

Audit

Cache Transform

Character Map

Conditional Split

Copy Column

Data Conversion

Derived Column

Lookup, if configured to Full Cache is enabled to store search results

Multicast

Percent Sampling

Row Count

Non-Blocking Synchronous Row-Based Transformations (Fast)

DQS Cleansing

Export Column

Import Column

Lookup, if configured to “Partial Cache” or “No Cache” to store search results

OLE DB Command

Script Component, if configured to use an outside resource

Slowly Changing Dimension (SCD)

**Semi-Blocking Asynchronous Transformations (Medium)**

Data Mining Query

Merge

Merge Join

Pivot

Term Lookup

Unpivot

Union All

Fully Blocking Asynchronous Transformations (Slowest)

Aggregate

Fuzzy Grouping

Fuzzy Lookup

Row Sampling

Sort

Term Extraction

Script Component, if configured to collect all data before send it to a destination. Set the “SynchronousInputID” property to “None” on the Output column to make it asynchronous.