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 be used 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.

--Page code

--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.

--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.

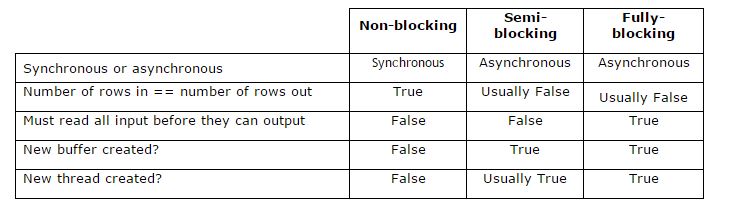
--Auditing Transformations

* + - Audit
    - Row Count

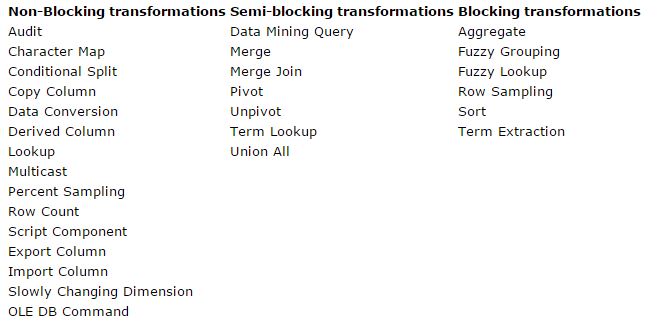
Two types of dataflow component

--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

In the Aggregate transformation, **Keys** and **Keys scale** refer to the number of groups that are expected to result from a **Group by** operation. **Count distinct keys** and **Count distinct scale** refer to the number of distinct values that are expected to result from a **distinct count** operation.

--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.

Fuzzy lookup transformation

--usage: can be used to perform data cleaning tasks to standardizing data and correct data by comparing the original data to a reference data.

-- **Three features for customizing** the lookup it performs:

Maximum number of matches to return per input row, token delimiters, and similarity thresholds.

--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, the confidence score is a measure of **how likely it is that a particular value is the best match among the matches found in the reference table**.

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

--this transformation has one input and one output

--look up column and correction column are two things

Fuzzy Group Transformation

-- Performs data cleaning by identifying potential 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.

--Configuration

a. Columns:

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 domain 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.

Control Flow:

A SSIS control flow contains three type of elements: **containers** that provide structures in packages, **tasks** that provide functionality, and **precedence constraint**s that connect the executable, containers, and tasks into an ordered control flow.

Task:

A task is a job unit in SSIS that provide certain functionality for a package. For example, we have Date flow task which deal with ETL strategy. We have File system Task that work with files and directory. There are various type of tasks and each of them will provide a different functionality.

Precedence Constraints:

--definition

Precedence constraints are used to connect the executable, containers, tasks in a package in order to form an ordered control flow

--there are three type of precedence constraint

a. Success: if we connect a success constrain from task a to task b, this mean task b can only be executed if task a is executed successfully.

b. Failure: If we connect a failure constraint, task b will be executed only if task a fails

c. Completion: For completion constraint, not matter task a succeed or not, it will be executed.

--configure

Specify an evaluation operation (constraint type, expression)🡪choose constraint, provide expression🡪use logical operator to combine two constraint to evaluate together

Bulk Insert Task

--usage

The Bulk Insert task provides an efficient way to copy large amounts of data into a SQL Server table or view. The Source file must be a text file. The destination must be an **existing table** or view in SQL server. The destination table can not be created on the run time.

--notice

If the file is on a remote server, you must specify the file name using the Universal naming convention name in the path

--notice

Transform is not allow for this task.

--configure

Source (file connection manager)🡪destination(OLE DB connection + the table)

Bulk insert task vs. Data flow task.

--transformation

--only support text file source, only support sql server table or view

--destination table must be created first

File System Task

--Usage

Performs operation on files and directories. It can also be used to set attributes on files and directories

--Operations: copy file/dir ; Create/Delete Dir or Dir contents, delete file, move directory/file, Rename file, set attribute .

--configure

Source (file connection manager or variable)🡪operations

Execute Process Task

--usage

Help to run an **application** through SSIS. For example, it can open any standard application such as notepad, excel, word or command prompt.

--configure

Executable name (file path of the application)🡪Arguments(Provide command prompt arguments.)🡪working directory🡪specify any input variable, output variable and the variable to capture error output

Execute Package Task

--usage

Used to run other package in current package.

--benefit

Smaller package: It helps to breaking down complex package workflow and reuse packages, which are easier to test read and maintain

Package reuse: You can build a package that is later referenced by multiple other packages

Work unite: Grouping work units. Units of work can be encapsulated into separate packages and joined as transactional components to the workflow of a parent package.

 Security: By separating a package into multiple packages, you can provide a greater level of security, because you can grant an author access to only the relevant packages.

--configure

Reference type (project reference for packages in the same project, and external reference for packages stored in file system)🡪Select the child package🡪bind variable or parameter in the parent directory to the variable parameter or variable in the child package, in order to pass input to child package. 🡪 **ExecuteOutOfProcess** property, if set to be true, the child package will run in a separate process, this may increase performance since they are running in separate ram

**Execute SQL Task:**

--Usage:

Execute SQL task runs any sort of SQL **statements** or **stored procedures** from a package.

For example, you can this task to create, drop, and alter a table or other database objects. You can truncate a table or view in prepare for data insertion. Run stored procedure. Map a value returned by a query into a variable that can be passed into the dataflow.

--supported connections

It support a range of connections, including Excel, OLEDB, ODBC, SQL mobile and ADO and ADO.net. Notice that isquerystoredprocedure option only work with the ADO and ADO.net

--3 ways to specify SQL statement (**SQLSouceType**)

You have three choices to specify the SQL statement. You can enter the statement directly, you can a variable that contains the query, and you also have the choice to connect to a SQL file using a file connection. These options can be selected at the SQLSouceType button in the General section of the task editor.

--result set option

You have to adjust the result set type depending on the type of SQL statement. None, single row, full result set and xml.

--support parameterized query or procedure (good feature)

You have the choice to a query parameter to a variable as well as map Result Sets to Variables. This can be done in the parameter mapping and result set page of the task editor (Parameter marker,?, and parameter name,0,1,2,3…)

-- GO command can be used to specify multiple statements as a Batch. And only the result of first select statement will be returned.

--configure: As most other tasks, you need to specify the connection. Choose a connection type and connection manager. You will have the choice to specify time out. Once you confirm the connection, you will have choose the SQLSourceType, it can be Direct input, from variable or from file. Make BypassPrepare as True if you want the task to prepare the query before executing it. Adjust the ResultSet option according to your expected result. In parameter Maping, map the variable to parameter. IN Resultset page, map the result to a variable.

Connection🡪input mode🡪result-set🡪parameter mapping and result mapping

Execute family task(process, package, sql, tsql…)

**Execute T-SQL Statement Task:**

--Usage

It is use to execute T-SQL statement. Only T-SQL is allow for this task.

--limitation

Can’t not be used to run parameterized queries. In other words, only allow hard coding. Can’t save result to a variable or using property expressions

--Supported Connection

Only support ADO.NET connection

--configure

Create a stored ADO.NET connection and then input T-sql statement

Create a ADO.NET connection🡪specify the T-sql statement

Execute SQL vs. Execute T-SQL Statement Task:

1. Connection supported
2. Version of SQL supported
3. Working with variable
4. Source of SQL statement
5. Memory, parse time, Performance issue

Script task:

--Usage: perform **functions that are not available** for the **build-in** tasks and transformations

For example: it can be used to access data that are not supported by built-in connections.

--Tools and Language

The script uses **Microsoft Visual Studio Tools for applications** as the environment for writing and executing script. The script can be written in either VB.NET or C#.Net programming language.

--Configuration

To configure the script, you first specify the language, it can either be v or c. Then you input the entry point, which is MAIN usually.( Specify the method in the VSTA project that the Integration Services runtime calls as the entry point into the Script task code.) Optionally, provide a lst of read-only and read/write variable that you want to include in the script.

Choose language, v or c🡪 entry point (main function)🡪provide variables(optional)

**Data Profiling Task**

--usage

This task will generate a profiling of our data and provide detail insight on the natural of data. The profile will be written a XML file which can be presented by another tool called data profile viewer. It can only be used for SQL server table or view

--connection supported

Only work with ADO.NET connection manager.

The table or view can only be a SQL server object.

--Account running data profiling task in SSIS must have read/write permissions on TempDB

--Date profiling function category

-Column based profiling (5 types)

**Column length** **distribution**: Reports all the distinct lengths of string values in the selected column and the **percentage** of rows in the table that each length represents.

**Column Null Ratio**: Reports the percentage of null values in the selected column.

**Column Pattern**: Reports a set of regular expressions that cover the specified percentage of values in a string column. Char data required

**Column Statistics**: Reports statistics, such as minimum, maximum, average, and standard deviation for numeric columns, and minimum and maximum for date time columns.

**Column Value Distribution**: Reports all the distinct values in the selected column and the **percentage** of rows in the table that each value represents.

-Multiple column based (3 types)

**Candidate Key**: Reports whether a column or set of columns is a potential key. It help to find duplicate value in a potential key

**Functional Dependency Profile**: Reports the extent to which the values in one column (the dependent column) depend on the values in another column or set of columns (the determinant column).

**Value Inclusion**: Computes the overlap in the values between two columns or sets of columns. Only one that could be used on **multiple tables.**

--How to configure: a. select destination, either a file or variable 2. Create a .net connection to connect to database 3 .Configure the profiling you want, you have to provide the columns

Create an ADO.net connection 🡪 set destination (file or variable) to save the profile🡪 list the profiling request the columns on which you want to perform the request

Data profile viewer

**Send mail task:**

--usage: sends emails messages from SSIS, basically you can do whatever you can do when sending a regular email. The sender and receiver. Subject. Attachment…

--Connection supported: SMTP Connection Manager

--how to configure

Specify the SMTP connection manager, notice it only support anonymous authentication and windows authentication. 🡪Sender, recipients, cc, subject, massage txt (can be from direct input, file or variable) and attachment

**FTP task:**

--usage: it is used to download and uploads data files and manages directories on servers.

--predefined set of operations

Send and receive files, (basic)

Create, remove local or remote directory (directory)

Delete Local or Remote Files (Files)

--connection supported: FTP connection manager

--How to configure: Create FTP connection manager (specify server Name and Port, Credentials for accessing the FTP server, Time-out, number of retry, Passive or Active , chuck size🡪operation, remote path and local path

--FTP connection manager🡪choose operations

--Wild Cards (‘\*’ and ‘?’) can be used to access multiple files.(similar to the profiling task)

Active server

Passive client

Windows authen not supported

SFTP

More secured version of FTP, it is used to send or receive file using secure connection

In passive, we have to connection

Containers

--definition and usage

Container are used to wrap up different components SSIS. They support repeating control flows, and they help to group tasks into meaningful units of work. More importantly, It helps to maintain

--types

For loop: Repeat tasks until a specified expression evaluates to **false**

Foreach loop: Repeat tasks for each every item in **a specify connection**

Sequence: defines a control flow that is subset of the package control flow

Task Host: The task host container encapsulates a single task. It’s a hidden container

For loop container:

--Usage: for loop container defines a repeating control flow. The task will be repeated until a specified expression is evaluated to be false

--configure

Create a variable🡪specify initialization value🡪specify evaluate condition🡪specify expression that increments or decrements the counter.

Foreach Loop Container

--usage: repeat task for each every task in a specified collection of items, it can be a collection of files in a directory, a collection of table in a database, records in list…

--enumerator types

There mostly used types

**Foreach File enumerator: enumerate files in a folder**. Specify a folder and the files to enumerate, the format of the file name of the retrieved files, and whether to traverse subfolders.

**Foreach Item enumerator: enumerate items that are collections**. For example, you can enumerate the names of executable. Define the items in the Foreach Item collection, including columns and column data types.

**Foreach ADO enumerator:  enumerate rows in tables.** Specify the ADO object source variable and the enumerator mode. The variable must be of Object data type.

Foreach ADO.NET Schema Rowset enumerator to enumerate the schema information about a data source. For example, you can enumerate and get a list of the tables in the AdventureWorks2012 SQL Server database.

Fully qualified vs name

Sequence Container:

--usage: define a control flow that is a subset of the package control flow. In this way, it can disable a groups of tasks together. Managing properties on multiple tasks, more importantly it helps to implement transaction.

Task Host Container: The task host container encapsulates a single task. It’s an invisible task.

In SSIS Designer, the task host is not configured separately, it is configure when you configure the task inside it.

This container extends the use of variable and event handlers to the task level

Error handling

--def: error handling is a practice of dealing with values in run time

--error handling

In SSIS, we can handle error at both control flow level.

--error handling at Control flow level

1. On Failure Precedence Constrain. We can connect a failure precedence constraint to a error work flow. And this error work flow will be executed only and if only the task fails.

2. Event handlers, which is the recommended way of handling error. We can set Event handler on task, container and packages and each executable has 12 different event that can be handled. It is like a trigger when the specified workflow is executed on selected event . By the way, two most used event are onError and OnTaskFaild

--Error vs Failure

An executable may have multiple errors when being executed. However it can only failed once

If a task has ten errors, then the on error handle will be triggered for 10 times

The on task failed handler is triggered when only and if only task false

Errors can lead to failure. For example, a task have a property called **MaximumErrorCount,** which is the maximum number of error that could fail that task

Precedence constraint vs Error handler

It is always recommend to use Error handler instead of Precedence Constraint.

First, for failure precedence constrain, it can only be used to handle failures, it can’t not be used to handle errors.

On the other hand, for each execution, validation will be performed on all control flow statement. If we use precedent constrain, the error work flow will also have to be evaluate, this may drag down the performance

--Error and Failure Escalation

Both errors and failures will be escalated to a component’s immediate parent. For example, a container

--Propagate is a variable that used to control the escalation behavior. For example, in the onerror handler, if you change the Propgate is False, the error for this task will not be escalated to parent level. The parental component will not aware of any error happening at the child level.

--error handling at Data Flow Level

Error handling at data flow level is based on the error output of a dataflow component. You can configure it as ignore failure, Redirect row.

Ignore Failure means when error occur, the component will ignore the error and continue

Redirect row: **Redirect error data to error output**, the component keep executing. Error output pipeline can be used to catch the error output

--FailPackageOnFailure: This property is for containers in a package. If True, a failure in components will fail the package

--FailParentOnFailure: This property is for a child container, If ture, failure in chile contain will at the same time fail the parent container.

Transactions in SSIS

--Transactions in SSIS can be implemented only at control flow level, which mean we can maintain transaction on package, container and task. We can’t maintain transaction at data flow level.

--**TransactionOption property**

TransactionOption is a property for packages, container as well as task. It is used to implement transaction in SSIS. There are three type of transaction options, we supported, not supported and required

Required: if the TransactionOption property is set to Required for a component, whenever the component is executed, it will be treated as a transaction, given that the parental level don’t not have a Required transactionOption. IF the paretal component is also mark as required, then the Transactionoption will just be treated as supported

Supported: this mean a component will be part of the transaction started by its immediate parent

Not supported: this simply means a component is not part of the transaction initiated by its parental component.

Check point:

--usage:

Check point can be used to execute a package from the point of failure of its previous execution.

In other words, it avoid restart the execution from the beginning of the package.

For example, you have a bulking insert task which insert 1 million records into the database, then after that you have some other tasks. If any of these tasks fails, by default, next timey you restart the exec,

It will load the 1 million records again. In this case, you can consider use check point to avoid the loading task and start at where it fails

--implementation:

it can only be implement at control flow level.

Four steps to implement

1. ChectPointFileName Property: you specify where you want to store the checkpoint file along with the filename
2. CheckPointUsage Property: Never, Ifexists and always. In case Never, the execution will never use a checkpoint file. Thus the package will always start from scratch. Ifexist, as its name suggest, the execution use a checkpoint file if there exists one. If a checkpoint file is not exist, then it will start from beginning. Always means the execution will always go with a checkpoint file. Different from Ifexist, this time a checkpoint file is mandatory for execution. If the file is there, then it will start from the point fails. If not, if will fail.
3. SaveCheckPoints Propery: this property determines whether a package saves checkpoints. It must be true if you want to implement a check point.
4. These three properties can actually found at the top three row of package property page.

Beside these properties, you need to set one more property to the tasks or containers you expected to fail. You must set the FailPackageOnFailure property to be true. This will make sure the failure of that component can be signal to package and thus recorded.

Logging

--usage

Used to capture run-time information about a package.

It is helpful for optimizing package performance (how to ??) And trouble shooting a package, it also can be used for auditing specific operations performed by package.

--type of logging

There are 3 major logging options: SSIS built login and Custom Logging as well as SSIS catalog logging

--SSIS building logging

This logging option utilize the build-in log providers.

--how to set up

Add log provider(text file, sql server…)🡪choose containers(task, container, package)🡪select event(onError, Ontaskfailed…)

--checkbox states

Selected: this explicitly indicates yes, the execution of this container will be logged

Cleared: this is an explicit no, no logging will be provided for this container

Unavailable: this means the container will inherit option from parent container

--log providers

Text files🡪.txt, SQL server🡪table, SQL profiler🡪.trc , windows event log, xml files

--custom logging

--usage: custom logging information by defining workflow in event handlers. In this way, it will specified info about the package, container and task, when an event happens and event handler is triggered.

--implementation

Execute SQL task: for example, you can selection some system variable that contained info on execution to a SQL table, what variables

Script Task: for example, you can a vb.net script to store specified system variables in a txt file

--Logging in SSIS catalog

--Usage: will login information on the server.

--4 Logging levels in SSIS catalog

**None:** Logging is turned off. Only the package execution status is logged.

**Basic:** All events are logged, except custom and diagnostic events. This is the default value.

**Performance**: Only performance statistics, and OnError and OnWarning events, are logged.( how long it took to run each task / component, etc)

it captures a lot more events internally to analyze the performance of the data flow components. As a result, there is a bit more overhead during execution – packages run with Basic will actually run a little faster than Performance (in this case Performance means “give me all of the performance details”, not “run my packages as fast as you can”).

**Verbose**: All events are logged, including custom and diagnostic events.

--custom log: special events that are raised with custom / 3rd party SSIS extensions or scripts

None->Basic (everything but custom and diagnose)->Verbose(everything)

Performance (focus on performance statistics, onerror, onwarning events)

--*Catalog.event\_messages : all the captured logs are stored in this view*

Package deployment model:

--usage of deployment: deployment is useful with you want to move you project or package to another server on when the environment may not the same as the current server**. It is all about assigning values to package properties.** For example, you are moving a package from development server to test server, a relevant database name change will cause error in package execution. In this way you can use a configuration file to adjust the corresponding changes.

--What is package deployment model?

--package deployment vs. project deployment

--packages: dtsx ; configurations .dtsConfig

--Types of Configurations

**XML file**: store the configurations in a xml file

**SQL server table**: let a table in server to store the configuration, you can use dml lauguage to update config

**Parent package variable**: in this case, variables in parent package stores configuration items and they are mapped to the properties of child package. This will also be Execute Package Task in the parent package so that we can call the child through parent.

**Environment variable**: map a system environment variable to the property of package. One variable can hold one property.

**Register entry**: use a registry entry to store configuration

--Direct and indirect configuration

Direct: direct link between **package object property** and **configuration item**

Indirect: in this case, a currently configuration item is further mapped to an environment variable. Instead of specifying the configuration setting directly. But how to make change

--configure and deploy

🡪Change the deploy mode to package🡪 package configuration wizard(select type, direct or indirect, map variable and property)🡪 set **CreateDeploymentUtility** True🡪 build the project

Package is deployed in MSDB Database

Project deployment

--What is project deployment?

--Configuration type in project deployment mode

a. Parameterization option: create a parameter, in the parameterize window, map the property of an object to that newly created variable. CAN ONLY use parameter

b. Master-Child packages: utilize a execute package task to run the child package from master package and any the same pass the specify property to child package parameter. You can bind variable or parameter to parameter only in Child package. This make sense.

c. SSIS environments in SSIS catalog. In this case, a property or variable of package object can be mapped to a variable in the SSIS environment. If the variable is connect to an environment in the currently project folder, this is a relative reference. Otherwise if it point to the environment of anther fold, then it is absolute reference

--Steps to deploy a project

Create SSISDB catalog🡪parameterization (optional)🡪build project(will generate project deployment file, .ispac)🡪deploy(follow deployment wizard) to server

--project security

a. file system(eg. Full control or read only)

b. project/package encryption: you can change this under the security(projectprotectionlevel) of package or project

5 options, for example, you can choose to encrypt a package with a password in regarding to the sensitive data. Notice that the Project and Package must have same level of encryption level

Do not save sensitive

Encrypt sensitive/all with user key

Encrypt sensitive/all with password

https://msdn.microsoft.com/en-us/library/ms141747.aspx

c. sensitive variable and parameter

d. package signing

e. security in SSIS catalog

Supports SQL Login based on Windows Authentication

There are two types of roles:

Public Role

SSIS Administrators

Variable:

--definition

Variables **store values** that a **package and its containers, tasks, and event handlers** can use at **run time**.

--usage

a. updating properties of package elements at run time, e.g. foreach loop container, connection string

b. a package can run an Execute SQL task that loads a variable with data values. The variable can be evaluated by expression

c. a for loop container will need variable for iteration

d. in exec sql task, parameter mapping, map variable to sql statement input parameter, map variable to result set

e. Building expressions that include variable values.

vs parameter

--expressions

--def: An expression is a combination of symbols that yields a single data value. Simple expressions can be a single constant, variable, or function. More frequently, expressions are complex, using multiple **operators** and **functions** and **referencing multiple columns and variables**..

--common usage:

a. Conditional Split transformation. Expressions used in a Conditional Split transformation must evaluate to **true** or **false**

b. The Derived Column transformation uses values created by using expressions either to populate new columns in a data flow, or to update existing columns.

c. Variables use an expression to set their value. For example, GETDATE() sets the value of the variable to the current date.

d. Precedence constraints can use expressions to specify the conditions that determine whether the constrained task or container in a package runs. Expressions used in a precedence constraint must evaluate to **true** or **false**.

e. The For Loop container can use expressions to build the initialization, evaluation, and the incrementing statements that the looping structure uses.

f. property of a package element. For example, the connection string can be replaced by expression which contains variable, so as to update the connection when variable changes.

ETL

Extract: extract data from various sources. The source can be oracle, access, db2 as well as sql server, or even from text file or excel. Normally, we will first extract this different sources to a staging database and performing data profiling to get a closer understanding of the data source

Transformation: perform a range of transformation tasks to make sure the data is consistent and uniform and more importantly meet the business rules. Common tasks are data cleaning and data validation, for example, we can used fuzzy lookup to correct misspelling and fuzzy group to remove duplication. We can also filter the data to discard the columns that are not vary useful for analysis. We will also join data from multiple sources by using union, merge or merge join. Finally, we can also business requirement on cleaned data, for example, deriving new calculated columns, performing some aggregations.

Loading: simple means to load the transformed data into a destination, which can be a data warehouse, data mart or repository.

Component in ETL

--OLTP Sources

--Pre-staging DB: data profiling, data mapping Document is prepared

--staging DB: Data transformation

--Date Warehouse/Datemart: Data population tasks place(Initial and Incremental Load)

Initial and Incremental Load:

--initial load: it happens when loading data to destination for the first time. All the data exists will be loaded

--Incremental load: This type of load happens after initial loading, usually happen on a regular basis. Here we give the last extract data and only record after this date are loaded.

Incremental techniques: distinguish between old, new, and updated records

-checksum, CDC, timestamp, triggers, joins, merger, md#2/4/5

Checksum()

--definition

A function will compute a hash value, also called the checksum value over a list of arguments. It helps to uniquely distinguish between different records.

--data type supported: all except text, image, xml

CHECKSUM value is dependent upon the collation

Doesn’t distinguish the trailing zero

Loading using Checksum

--initial load

Populate tracking table for all record, the tracking table will including business key and the hash value calculated based on that record.

--incremental load

Compare business key of tracking, for unmatching, is new records, instert, update tracking table. For matching, compare checksum values, if the check sum value is the same, then this is a duplicate. If not match, then this is a updated record, so update the data mart record and update track table

Pros and cons of checksum

--pros

Fast and easy to implement

Good for small and midsized databases

--cons

Probability of collision for large dateset

Checksum value will be difference if the order of column is different, even the values is the same

Some date type are not supported

Null value

I

Is sort key

Error tolerance index

For a component, failure happen when errors reach to certain number

Error will escalate to direct parent component

Two ways to keep container running: a. change propagate to False (set at child level. prevent error escalation); b. change container property MaximumErrorCount

Package 🡪 container 🡪 task

Services.msc DTC

Performance issue

2 types of deployment models

--Project Deployment model (

--Package Deployment model (since 2005)

Development server🡪test server🡪production server

Move package from one server to another server

Improve the portability of a package, easily distribute, since package has to be moved across servers.

Update the value of the property at run time

For xml config, -🡪 .dts

expressions