SQL Server 2005 - SQL Server Integration Services - Part 1

One of the SQL Server 2005 Beta 2 features that has undergone considerable architectural, functional, and interface changes, compared with its earlier versions, deals with extraction, transformation, and loading (also known as simply ETL) of data. (In other words, allowing you to move data between a variety of stores and to modify it in the process, according to your arbitrary requirements). While in SQL Server 7.0 and 2000, such capabilities have been provided by Data Transformation Services (DTS in short), Microsoft decided that the degree of modifications introduced in the new version justified re-branding it, hence its rearchitectured and significantly improved implementation has been introduced under the new name of SQL Server Integration Services (SSIS). For the same reason, rather than pointing out changes that have been implemented since the release of DTS in SQL Server 2000, in the next few articles, we will provide a comprehensive analysis of its SQL Server 2005 equivalent. For review of the SQL Server 2000 version of Data Transformation Services refer to our earlier series covering this topic and published on the Database Journal Web site).

Let's start with an overview of the most essential terms and ideas that are critical to understanding how ETL operations are handled in SQL Server 2005. The concept of the DTS Package has been inherited from the earlier versions of SQL Server and still is considered as the centerpiece of the data transformation object model, representing a unit of work that can be independently saved, retrieved, or executed, and which serves as a container for all other DTS objects. Within a package, you can distinguish two main categories of objects - control flow or data flow. Objects in the first category determine processing sequence within the package and consist of containers, tasks, and precedence constraints. Objects that belong to the second category define the flow of data. They can be further divided into source adapters, transformations, and destination adapters. Control flow dictates architectural structure of a package and can contain any number of data flow units (defined independently of each other and consisting of any number of data flow objects). The following are short descriptions of each of the control and data flow object types:

- Containers - group variety of package components (including other containers), affecting their scope, sequence of execution, and mutual interaction. With graphical utilities (which we will describe shortly), this is done by simply placing components inside them during initial package creation or subsequent modifications. Alternatively, the same goal can be accomplished with programmatic methods or by using wizards (while the former offers the best degree of versatility, the latter is restricted to pre-defined options and its customizability is very limited - although note that it is possible to further edit packages created with wizards). Depending on their characteristics, containers can be divided into the following types - Package (since each package also qualifies as a special type of container), Sequence, For Loop, ForEach Loop, TaskHost, and Event Handler. We will be discussing them in more details later in this series of articles.
- Tasks - are responsible for performing actual work, ranging from retrieval and loading of data across a variety of repositories, through interacting with other entities (including other DTS packages, Message Queues, Windows Instrumentation Management processes, SQL Server or Analysis Services objects, etc.), to performing data transformation and execution of SQL Server administrative tasks. In general, they belong to one of following seven categories - Data Flow, Data Preparation, Workflow, SQL Server, Scripting, Analysis Services, and Database Maintenance. The total number of pre-defined tasks has increased almost twofold compared with SQL Server

2000 (and custom ones can be developed with either legacy COM-capable or .NET programming languages). Tasks will also be covered more thoroughly as part of our series.

- Precedence Constraints - define links among containers and tasks and evaluate conditions that determine the sequence in which they are processed. More specifically, they provide transition from one container (such as For Loop, ForEach Loop, and Sequence) or task to another. Conditions that control whether transitions will occur are evaluated based on constraint properties and may involve execution outcome (success, failure, completion) of the preceding executable, a result of an arbitrary expression, or outcome of other precedence constraints.
- Source Adapters - facilitate the retrieval of data from various data sources (such as flat files or variety of databases) using flat file, OLE DB and .NET Framework data providers (relying for this functionality on their respective connection managers, which we will describe shortly). They also include the script component, which allows implementing and executing custom code, whenever functionality offered by built-in providers do not suffice (for example, when it is necessary to incorporate into a data flow business rules specified in external .NET assemblies). Script component can also be used to implement transformations and destination adapters.
- Transformations - perform modifications to data, through a variety of operations, such as aggregation (e.g. averages or sums), merging (of multiple input data sets), distribution (to different outputs), data type conversion, or reference table lookups (using exact or fuzzy comparisons).
- Destination Adapters - load output of the data flow into target stores, such as flat files, database accessible via various providers, or in-memory ADODB recordsets (similar to source adapters, they rely on appropriate connection managers to accomplish this). As you might expect, most commonly, data flow takes the path from one or more source adapters, through an arbitrary number of transformations, to target data stores defined using destination adapters.

On a more granular level, data and control flow elements include other types of package components, such as connection managers and variables. Connection managers are logical representations of connections to various types of data stores (such as ActiveX data objects and ADO.NET objects, file system folders and files, message queues, OLE DB- or ODBC-compatible databases or SMTP mail servers) along with their properties (for example a connection string, data source identifier, or description). Connections can be defined as part of tasks or transformations, and are used when setting up source and destination adapters. Variables provide temporary storage for parameters whose values can change from one package execution to another, accommodating package reusability (for example, allowing to execute the same T-SQL statement or a script against a different set of objects). They can be incorporated into containers, tasks, event handlers, and precedence constraints. In general, there are two types of variables - system (pre-defined) and custom (created on an as needed basis at the time of package development). We will demonstrate their use in future articles of this series.

Also among new DTS package components introduced in SQL Server 2005 are event handlers and log providers. With the former, you can write custom code that is invoked in response to a specific type of event triggered by executable components, such as ForEach Loop, For Loop, Sequence, and TaskHost containers (as well as packages themselves). For example, by creating a handler for the OnError event, you might be able to gracefully resolve a particular type of error condition or provide notification about its occurrence. Log providers allow the capturing of events taking place when tasks execute within ForEach Loop, For Loop, and Sequence containers and record them in a text or XML file, SQL Profiler log, SQL Server databases, or the Windows Application Event log.

Now, following our short introduction into ETL concepts, let's look next into their implementation, in particular focusing on management lifecycle of DTS packages. In our discussion about the utilities present in SQL Server 2005 Beta 2, we briefly mentioned Business Intelligence Development Studio. This is the primary environment for package development. As with the SQL Server Management Studio, its interface follows the design philosophy derived from Microsoft Visual Studio .NET, reflecting this way its main purpose (as far as Business Intelligence solutions are concerned). Within its scope, besides Analysis Services and Reporting Services-related features, there is also a set of utilities and wizards that provide the ability to create, modify (with proper version control through integration with Visual SourceSafe), debug, and deploy packages. Packages are developed based on the paradigm of a solution consisting of one or more DTS specific projects, which also can be combined with Analysis or Reporting Services projects. Subsequently, such solutions can be managed with SQL Server Management Studio, which includes the ability to store them in the msdb database (in addition to using .dtsx file format - option that is also available from Business Intelligence Development Studio), schedule, and execute them (but not modify, deploy, debug, or provide version control).

The main ETL development utility included with SQL Server Business Intelligence Development Studio is DTS Designer. DTS Designer is a sophisticated and highly versatile tool, giving access to an overwhelming majority of DTS functionality. In addition, SQL Server Management Studio offers DTS Import/Export Wizard (also accessible by invoking the DTSWizard executable, which resides in C:\Program Files\Microsoft SQLServer\90\Tools\Binn\VSShell\Common7\IDE folder, from the Command Prompt). It provides a limited ability to create packages outside of the primary development environment (but at the same rate, simplifying the development process by delivering an intuitive, step-by-step guide). It is also possible to fully manage packages with purely programmatic methods. Package execution is triggered with the Start entry from the Debug menu (or equivalent toolbar button) in the DTS Designer interface (corresponding to F5 function key shortcut). The same can be accomplished with DTExec or DTExecUI command line utilities (both located in the C:\Program Files\Microsoft SQL Server\90\DTS\Binn folder). Besides running packages interactively, you also have an option of scheduling them for unattended execution with SQL Server Management Studio. The DTUtil command line utility (also residing in C:\Program Files\Microsoft SQL Server\90\DTS\Binn folder) offers the ability to copy packages between files, msdb SQL Server database, or DTS Package Store, as well as signing them in order to protect their content. To deploy packages, use DTS Package Deployment Wizard, which installs packages either to the file system or msdb database. Package Configurations, new in SQL Server 2005, facilitate the changing of properties of package components at run time, by applying values stored outside of it (it is possible to use environment variables, registry entries, XML-formatted files, or variables contained in a parent package for this purpose).

While each edition of SQL Server 2005 includes the ability to execute packages, the level of the remaining ETL related functionality varies. In particular, no additional features are available in the Express and Workgroup versions. Starting with the Standard edition, you have access to Business Intelligence Development Studio, which allows you to develop packages with DTS Designer. More advanced transformations and tasks, dealing with data and text mining or fuzzy operations (such as Analysis Services Partition Processing Destination, Analysis Services Dimension Processing Destination, Data Mining Training Destination, Data Mining Query Component and Task, Term Extraction and Lookup, as well as Fuzzy

Grouping and Lookup) are available only when you switch to the Enterprise product. Fuzzy operations, new in the SQL Server product line, provide the ability to clean up data, either at the time of its loading (Lookup) or already existing in a data store (Grouping), by comparing the content of "candidates" rows against the ones already established as valid, and finding the closest matches in case of inconsistencies. (This eliminates the potential for misspellings, spelling variations of the same name, which yield to duplications and other typical data input-related issues).

Another ETL-related change in SQL Server 2005 is the introduction of Integration Services Service, running in the security context of NT AUTHORITY\NETWORK account, which takes part in managing the operation of DTS packages. It is controlled from the SQL Server Management Studio from where you have an option of starting or stopping it (pausing and restarting states are not available). Its main purpose is to monitor status of running
packages, but it is also required in order to launch packages from Integration Services node in SQL Server Management Studio (although its status does not affect the ability to invoke them via SQL Server Import and Export Wizard or via a scheduled SQL Server Agent job). Stopping it will also trigger the stopping of all running packages.

In our next article, we will present the process of creating a simple package, explaining at the same time all relevant ETL functionality in more detail.