SQL Server 2008 Integration Services

Utilizar a ferramenta certa para cada tipo de trabalho é uma decisão importante para projetos dentro de qualquer empresa. Dentro do *Microsoft SQL Server 2008* o problema descrito anteriormente é endereçado a princípio pelo *Integration Services* (ou SSIS, como ficou conhecido na versão *SQL Server 2005*), capaz de criar soluções robustas e complexas de extração, transformação e carga de dados (ou ETL, do inglês *Extract, Transform and Load*). Posteriormente, outros serviços do *Microsoft SQL Server 2008* propiciarão os benefícios sugeridos no início desse artigo.

Focaremos daqui por diante na construção de um processo automatizado de carga utilizando o *SQL Server Integration Services*, na versão 2008, com o intuito de fornecer uma solução flexível e de fácil manutenção.

Integration Services é para mim?

Sim, é! A Microsoft foi paulatinamente direcionando todas as consoles e IDE´s de desenvolvimento de seus produtos para o Visual Studio. O que é bastante justificável, pois mais do que um ambiente de desenvolvimento para .NET, o Visual Studio suporta diversos tipos de projetos, como por exemplo, BizTalk, Office, Silverlight, e é claro, o SQL Server através do chamado ***Business Intelligence Development Studio Environment***.

Outra razão para o direcionamento dos ambientes de desenvolvimento para o Visual Studio é a separação clara dos papeis dentro ciclo de criação de uma solução. Para equipes que pretendem desenvolver soluções baseadas nos serviços da plataforma SQL Server são sugeridos os papeis de ***arquiteto***, ***desenvolvedor***, ***analista*** e ***usuário***, embora muito comumente os papeis de ***arquiteto*** e ***desenvolvedor*** sejam desempenhado pela mesma pessoa, às vezes acumulando também o papel de analista de negócio.

Como desenvolvedor de soluções baseadas em SSIS (*SQL Server Integration Services*), existem duas abordagens fundamentais na programação:

- Estender pacotes no SSIS codificando a partir dos componentes existentes para prover funcionalidades customizadas;
- Criar, configurar e executar os pacotes programaticamente através de suas aplicações.

Ambiente necessário

Para desenvolver o exemplo deste artigo será necessário instalarmos o *Business Intelligence Development Studio Environment* e o *SQL Server Management Studio* disponíveis na instalação do *Microsoft SQL Server*. Confira a seção de links para mais informações sobre o processo de instalação. Uma vez instalado o *Business Intelligence Development Studio* teremos disponível no *Visual Studio* um novo tipo de projeto chamado *Business Intelligence Projects*, conforme podemos conferir na **Figura 1**.

Será necessária a instalação do *Database Engine Services* e do *Integration Services*, caso não existam no seu ambiente de desenvolvimento e, é claro, o *Business Intelligence Development Studio*. A Microsoft disponibiliza uma edição de avaliação especial do SQL Server 2008 que expira após 6 meses de uso (confira o link para download na seção de links). Após efetuar o download, extraia os arquivos e execute o *setup*. A instalação é bastante intuitiva, basta seguir os passos indicados. Lembre-se de selecionar *Database Engine Services* (dentro deste item selecione também *Full-Text Search*, para que o banco de dados de exemplo funcione corretamente) e *Integration Services* no item *Feature Selection*.

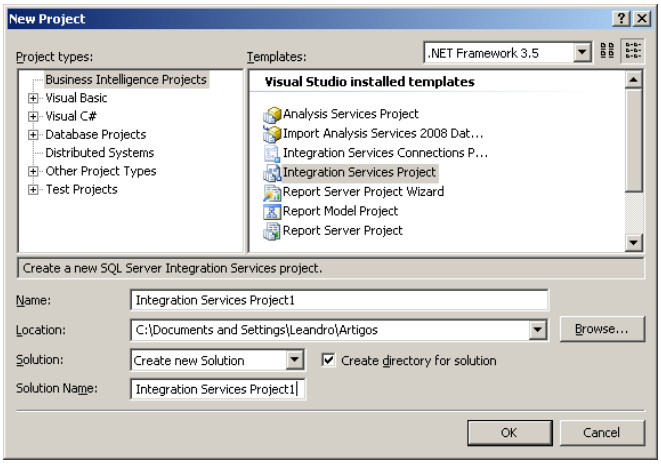

Figura 1. Templates disponibilizados com a instalação do Business Intelligence Development Studio

Cenário

Para este artigo estudaremos a implementação de um projeto SSIS baseado no método *ETL*, que consiste em realizar a consolidação de informações oriundas de fontes de dados de diferentes tecnologias por meio de:

- **Extract (extração):** Extração dos dados de uma ou mais fontes;
- **Transform (transformação):** Aplicação de regras de negócio e transformações nos dados, quando necessário;
- **Load (carga):** Carga dos dados para um destino (geralmente um *database* ou *data warehouse*).

Como mencionado no início deste artigo, o SSIS é um importante componente para integração dos demais serviços do *SQL Server* com o objetivo de prover consultas estratégicas baseados em banco de dados OLAP, mineração de dados, relatórios, dentre outras possibilidades.

Faremos a carga das informações contidas em planilhas do *Excel* para uma base de dados *SQL Server* e, por meio deste exemplo, veremos os principais conceitos do SSIS e conheceremos alguns dos seus componentes. Utilizaremos o banco de dados de exemplo *AdventureWorks* disponível para download no CodePlex (veja sessão Links) e faremos a carga das informações para a tabela *ProductReview*, cujo modelo conferimos na **Figura 2**.

Vamos agora criar algumas planilhas do Excel para o nosso cenário. Uma das novidades do SQL Server 2008 é o suporte para planilhas criadas na versão 2007 do Microsoft Office, que possuem a extensão “xlsx”. Seguindo o modelo descrito nas **Tabelas 1** e **2**, crie os arquivos utilizando a versão do *Excel* que você tiver disponível (neste arquivo serão utilizadas planilhas da versão *Microsoft Office Excel 2007*).

O primeiro arquivo deverá ser renomeado para ***Reviewer1.xlsx*** e o segundo para ***Reviewer2.xlsx***. Com o banco de dados de exemplo e as planilhas criadas, podemos prosseguir no Visual Studio 2008.

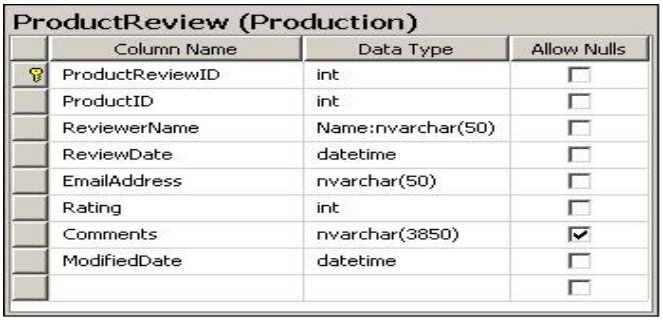

Figura 2. Tabela ProductReview do banco de dados AdventureWorks

ID	Reviewer	Date	Email	Rating	Comments
1	Leandro Daniel	04/05/2009	contato@leandrodaniel.com	5	Approved
2	Leandro Daniel	05/05/2009	contato@leandrodaniel.com	3	Good
3	Leandro Daniel	08/05/2009	contato@leandrodaniel.com	4	Very good
4	Leandro Daniel	10/05/2009	contato@leandrodaniel.com	1	Terrible

Tabela 1. Informações da planilha Reviewer1.xlsx

ID	Reviewer	Date	Email	Rating	Comments
1	Bruno Daniel	08/05/2009	bruno@email.com	5	Cool
4	Bruno Daniel	14/05/2009	bruno@email.com	1	No comments

Tabela 2. Informações da planilha Reviewer2.xlsx

Com o *SQL Server* já devidamente instalado, crie no *Visual Studio 2008* um novo projeto a partir do menu *File / New Project*. Na janela *New Project* escolha no painel *Project Types* o tipo *Business Intelligence Projects* e no painel *Templates* selecione *Integration Services Project*. Digite **NetMag.ETL** no campo *Name* e **NetMag.Solution** no campo *Solution name,* clicando em *Ok* em seguida. O projeto será criado apresentando a estrutura ilustrada na **Figura 3**. Note que como padrão consta um pacote chamado *Package1.dtsx* dentro do diretório *SSIS Packages* (esses diretórios fazem parte da estrutura de organização do *Integration Services* e não podem ser excluídos). Perceba também que a janela do *SSIS Designer* é apresentada (caso isso não ocorra, dê um duplo clique no item *Package1.dtsx* na janela *Solution Explorer*).

Principais conceitos do SSIS Designer

Conforme podemos conferir na **Figura 4**, o SSIS Designer possui quatro guias: ***Control Flow***, ***Data Flow***, ***Event Handlers*** e ***Package Explorer***.

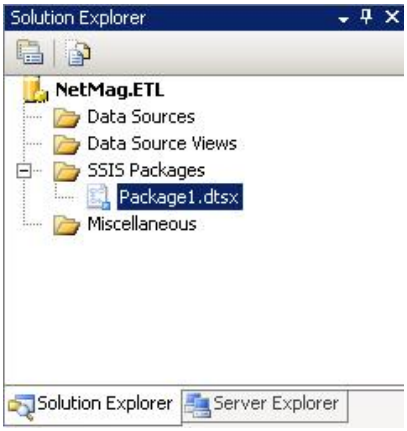

Figura 3. Projeto do Integration Services na janela Solution Explorer do Visual Studio

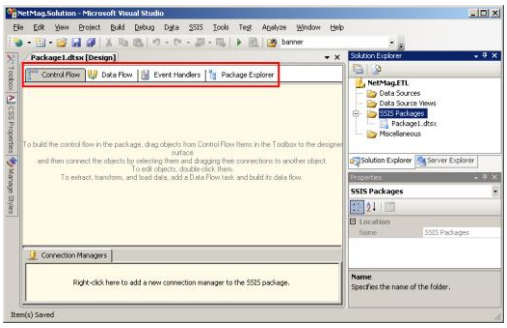

Figura 4. Janela do SSIS Designer com destaque nas guias que

A seguir temos a descrição de cada uma das guias presentes no SSIS Designer:

- **Control Flow:** Serve para definir macro atividades que o pacote executará, onde o fluxo e a precedência de cada atividade são desenhadas e configuradas;
- **Data Flow:** Nesta guia construímos as funcionalidades de extração das fontes de dados, transformação e carga para o destino;
- **Event Handlers:** Permite definir ações em decorrência de determinados eventos durante a execução do pacote, como por exemplo falhas na execução de uma determinada tarefa;
- **Package Explorer:** Enumera a estrutura de objetos que compõem o pacote de forma hierárquica.

Além dos conceitos descritos acima o *SSIS Designer* possui outros objetos importantes, descritos a seguir:

- Connection Managers: objetos responsáveis pelo acesso às fontes de dados e os destinos;

- **Variables:** Objetos que podem ser utilizados em vários contextos do pacote para atualizar dinamicamente valores de uma coluna, expressões das propriedades dos controles além de definir condições de precedência entre as atividades. A janela ***Variables*** está disponível no menu ***View*** / ***Other Windows*** / ***Variables***.

Desenvolvendo o pacote ETL

Iremos agora renomear o pacote padrão *Package1.dtsx*, clicando com o botão direito na janela *Solution Explorer* e escolhendo *Rename*. Altere o nome do arquivo para ***LoadReviewer.dtsx***. Utilizando a guia *Control Flow* do *SSIS Designer*, escolha na janela *Toolbox* o controle *Foreach Loop Container* e arraste-o para dentro do painel *Control Flow*. Este controle permitirá varrermos uma estrutura de diretórios indicada em busca dos arquivos para carga, que no nosso exemplo serão planilhas do Excel.

Da janela *Toolbox* escolha agora o controle *Data Flow Task* e arraste-o para dentro do controle *Foreach Loop Container*. Clique com o botão direito no controle *Foreach Loop Container* na janela do *SSIS Designer* e escolha *Edit*. Na janela *Foreach Loop Editor* altere as propriedades da opção *Collection* conforme a **Figura 5** demonstra. Ao escolher a opção *Foreach File Enumerator* para a propriedade *Enumerator* serão exibidas novas propriedades nas quais informaremos o caminho do diretório que contém os arquivos para carga, a máscara de nome para os arquivos e a opção de procura em subdiretórios.

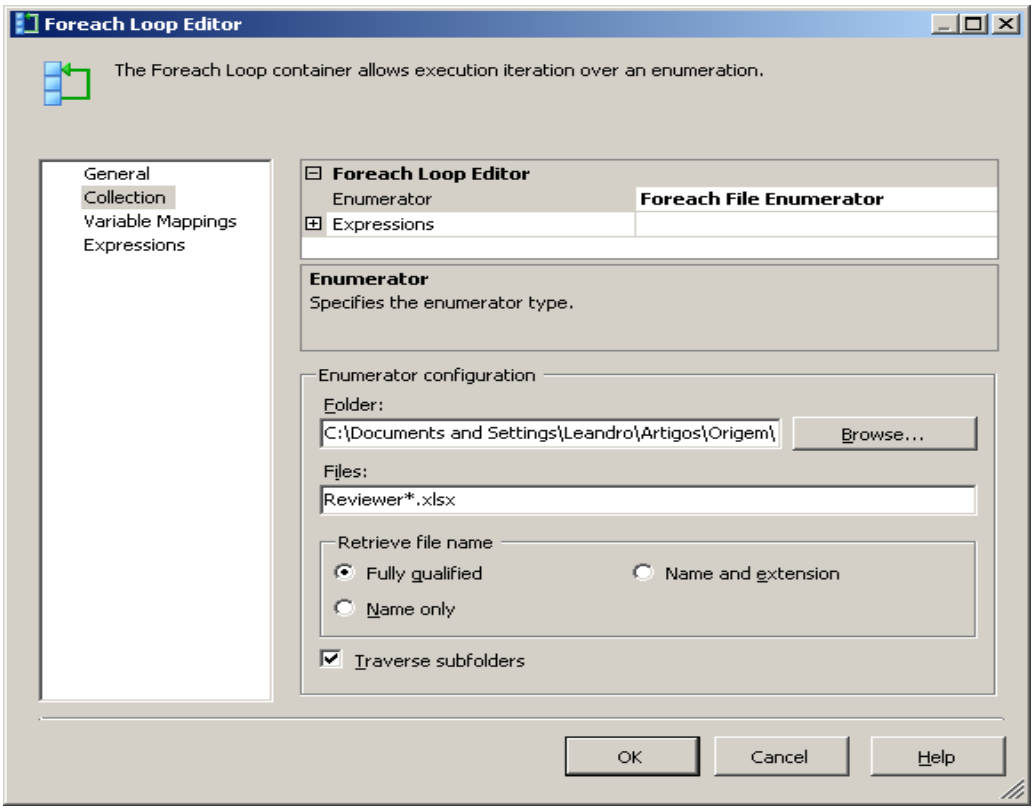

Figura 5. Configurações da opção Collection do objeto Foreach Loop Container

Vamos criar agora uma variável que conterá o nome do arquivo sendo acessado pelo laço. Certifique-se de que o objeto selecionado é o próprio pacote (para isso confira na janela *Properties* se o objeto que aparece é o próprio *LoadReviewer*). Caso não seja, apenas clique em uma área em branco da aba *Control Flow*. Isso é importante para que a variável seja criada no escopo correto. Em seguida, utilizando a janela *Variables* (disponível no menu *View / Other Windows / Variables*), clique no primeiro botão localizado na parte superior esquerda (*Add Variable*) para criar uma variável chamada *Planilha* e utilize a janela *Properties* para configurá-la conforme a **Tabela 3**.

Voltaremos agora no controle *Foreach Loop Editor*, clicando com o botão direito nele e escolhendo a opção *Edit*. Configuraremos agora a opção *Variable Mappings* indicando para o controle que cada arquivo que ele localizar será atribuído a variável *Planilha*, conforme ilustrado na **Figura 6**.

O valor **0** (zero) da propriedade *Index* retorna o caminho completo com o nome do arquivo encontrado pelo controle. Clique em *Ok* para confirmar as alterações e fechar a janela *Foreach Loop Editor*.
Na guia *Control Flow* do *SSIS Designer*, clique com o botão direito no controle *Data Flow Task* e escolha a opção *Edit*. Note que a guia *Data Flow* será exibida, pois é nela que fazemos todas as configurações e definições da atividade *Data Flow Task* selecionada. Usando a janela *Toolbox*, arraste o controle *Excel Source* do grupo de controles *Data Flow Sources* para dentro do painel. Em seguida, arraste o controle *Data Conversion* do grupo de controles *Data Flow Transformation* para o painel *Data Flow*. E por último arraste um controle *SQL Server Destination* do grupo de controle *Data Flow Destinations*. Esses três controles farão respectivamente a extração, transformação e carga do nosso processo de ETL.

Propriedade	Valor
ValueType	String
Value	C:\Origem\Reviewer1.xlsx Nota importante: Informe aqui o caminho onde as planilhas foram armazenadas na sua máquina. Será necessário fornecer também o nome do arquivo, pois o SSIS o tomará como base para modelar o mapeamento entre as fontes de dados, conforme veremos mais adiante.

Tabela 3. Configuração das propriedades da variável Planilha

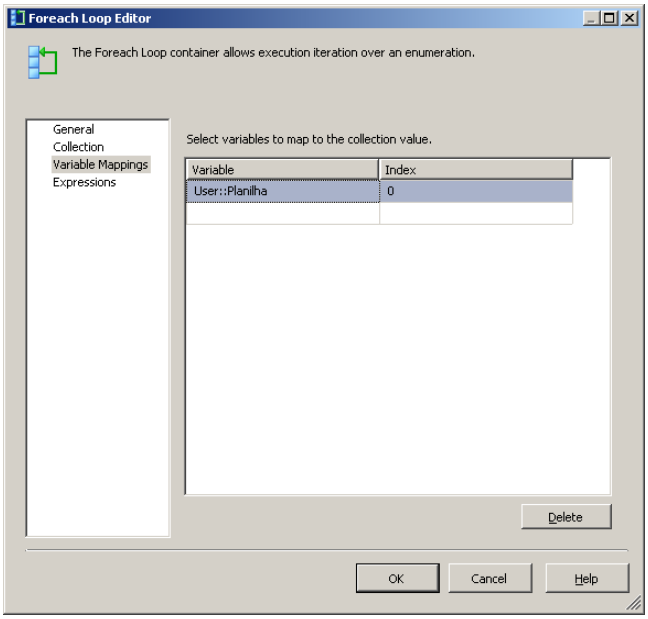

Figura 6. Configurações da opção Variable Mappings do objeto Foreach

Faremos agora a conexão entre os controles. Primeiro, clique no controle *Excel Source* e repare que ele exibirá duas setas, uma verde e outra vermelha. Clique na seta verde e arraste até o controle *Data Conversion*, de forma a conectá-los. Clique agora no controle *Data Conversion* e arraste a seta verde até o controle *SQL Server Destination*. Com esses procedimentos estamos definindo tanto a ordem de execução como também em que condição cada tarefa será executada, pois a seta verde indica que a tarefa sucessora somente será executada se a tarefa anterior finalizou com sucesso e, sendo assim, se quiséssemos tratar algum fluxo diferenciado no caso de falhas na execução de uma tarefa qualquer usaríamos a seta vermelha para a conexão dos controles. Neste ponto a guia *Data Flow* estará com o aspecto ilustrado na **Figura 7**. Perceba que os controles *Excel Source* e *SQL Server Destination* estão com uma indicação de erro, isso porque não definimos ainda onde eles se conectarão.

Essa será nossa próxima atividade.

Clique com o botão direito no controle *Excel Source* e escolha *Edit*. Na janela *Excel Source Editor* clique no botão *New* e em seguida informe o caminho do arquivo *Reviewer1.xlsx* e escolha a opção *Microsoft Excel 2007* para a propriedade *Excel Version* (caso você tenha optado por usar uma versão anterior do Excel basta escolher a versão apropriada). Certifique-se que a opção *First row has column names* esteja marcada. Para confirmar e fechar a janela clique em *Ok*. Será criado um novo *Connection Manager* chamado Excel Connection Manager. Ainda na janela *Excel Source Editor* escolha a planilha que contenha as informações de carga para a propriedade *Name of the Excel sheet* que no caso deste exemplo, como foi usada a versão em português do Excel, será *Plan1$* (mas tenha em mente que isso pode variar).

Caso você tivesse optado por dar um nome diferente para a *Sheet* do Excel bastaria informá-la nesta tela. Clique em *Ok* para confirmar e fechar a janela *Excel Source Editor*.

Faremos agora a configuração para o controle *Data Conversion*. Neste exemplo ele será utilizado para fazer a transformação dos tipos do Excel de forma que fiquem compatíveis com os tipos de dados do SQL Server. Clique com o botão direito sobre o controle *Data Conversion* e escolha a opção *Edit*. Na janela *Data Conversion Transformation Editor* configure as transformações conforme a **Tabela 4**. Clique em *Ok* para confirmar e fechar a janela.

[ampliar] [abrir em janela]

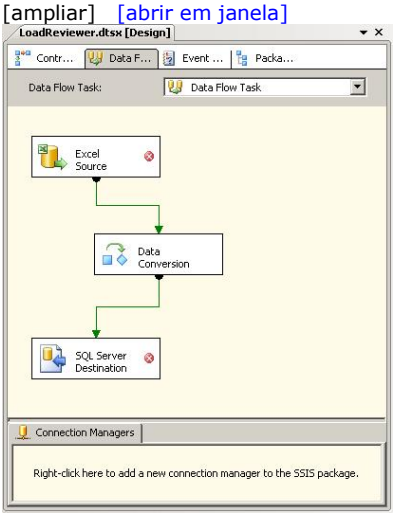

Figura 7. Guia Data Flow com os controles para execução do processo de ETL

Input Column	Output Alias	Data Type	Length
ID	Copy of ID	four-byte signed integer [DT_I4]	
Date	Copy of Date	database timestamp [DT_DBTIMESTAMP]	
Reviewer	Copy of Reviewer	Unicode string [DT_WSTR]	50
Email	Copy of Email	Unicode string [DT_WSTR]	50
Rating	Copy of Rating	four-byte signed integer [DT_I4]	

Tabela 4. Configuração do controle Data Conversion

Por fim iremos configurar o controle *SQL Server Destination* clicando com o botão direito e escolhendo *Edit*. Na janela *SQL Destination Editor* clique no botão *New* da propriedade *Connection Manager*. Na janela seguinte clique no botão *New* e informe as propriedades de conexão para a sua base de dados *SQL Server* e selecione o banco *AdventureWorks*.

Ao finalizar clique em *Ok* e em seguida novamente em *Ok*. Ainda na janela *SQL Destination Editor* escolha a tabela *[Production].[ProductionReview]* para a propriedade *Use a table or view*, pois assim definimos a tabela que receberá os dados da carga das planilhas do *Excel*. Por último iremos clicar na opção *Mappings* do painel esquerdo da janela *SQL Destination Editor* e fazer o mapeamento entre os campos da data source e a tabela de destino, conforme ilustrado na **Figura 8**.

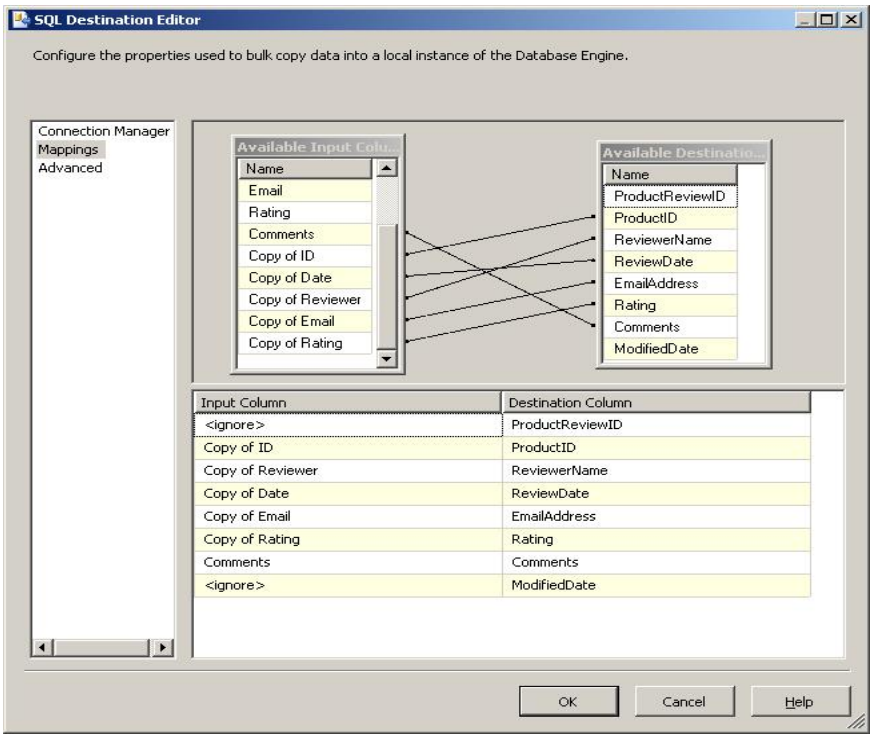

Figura 8. Mapeamento entre os

Note que o mapeamento entre os campos *Comments* e *Rating* é feito automaticamente, isso porque o SSIS assume que esse mapeamento está correto já que os campos possuem o mesmo nome. Os demais mapeamentos serão feitos manualmente, tomando o cuidado de

escolher os campos criados pelo controle de transformação *Data Conversion*. Outro detalhe é que deixaremos de fora o mapeamento para a coluna *ProductReviewID*, por ser esse um campo de autoincremento do SQL Server e o campo *ModifiedDate*, por possuir uma regra de preenchimento default.

Utilizando as variáveis para empregar dinamismo ao pacote

A variável *Planilha* foi criada para que pudéssemos aplicar dinamismo ao pacote, pois como estamos usando o controle *Foreach Loop Container* precisamos informar ao *Connection Manager* do Excel a string de conexão dinamicamente.

Para isso, na guia *Data Flow* clique no objeto *Excel Connection Manager* localizado na guia *Connections Managers* no canto inferior. Utilize a janela *Properties* para configurar a propriedade *Expressions* clicando no botão com os três pontos destacado na **Figura 9**.

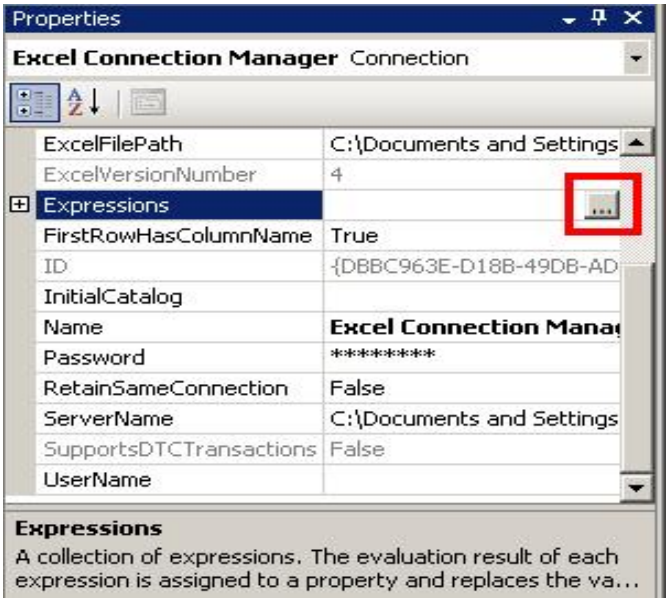

Figura 9. Configuração da propriedade Expressions do Excel Connection Manager

Na janela Property Expressions Editor escolha na coluna Property o valor ExcelFilePath e clique no botão da propriedade Expression para abrir a janela do Expression Builder. Na área Expression digite o seguinte:

@[User::Planilha]

Dessa forma indicaremos para o controle Excel Connection Manager que o arquivo a ser acessado será o contido na variável Planilha, que é preenchida a cada iteração do controle Foreach Loop Container. É possível testar a configuração clicando no botão Evaluate Expression da janela Expression Builder. Clique em Ok para confirmar e depois em Ok novamente. O aspecto final da guia Data Flow deverá ser similar ao ilustrado na **Figura 10**.

O pacote LoadReviewer.dtsx está agora totalmente finalizado e funcional. No próximo passo iremos realizar um teste executando o pacote localmente.

Testando localmente o pacote

Para executar o pacote, testando as configurações realizadas, pressione F5 no Visual Studio ou escolha no menu Debug a opção Start Debugging. O pacote será executado e todas as tarefas definidas serão executadas uma a uma, sendo que as cores identificam a situação de cada tarefa. Se o controle estiver com a cor amarela significa que aquela

atividade está sendo executada naquele instante.

Já a cor verde indica que a tarefa foi executada com sucesso e a cor vermelha indica que houve alguma falha na execução. Depois de finalizado o pacote deverá apresentar os resultados descritos nas **Figuras 11, 12 e 13**.

O SSIS Designer exibe uma nova guia chamada Progress, onde é possível acompanhar todos os passos executados e ajudá-lo na identificação dos erros em caso de falhas. Esse tipo de visão também está disponível para o operador do Integration Services através do SQL Server Management Studio.

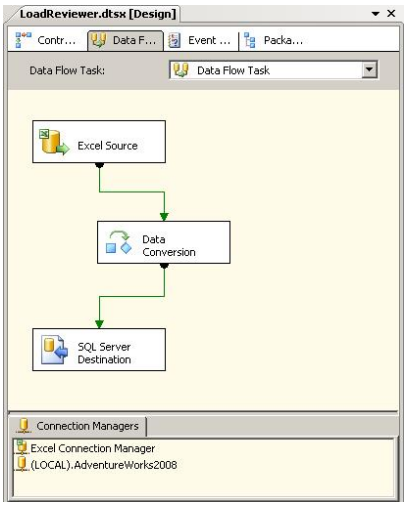

Figura 10. Guia Data Flow com todos os controles e connection managers configurados

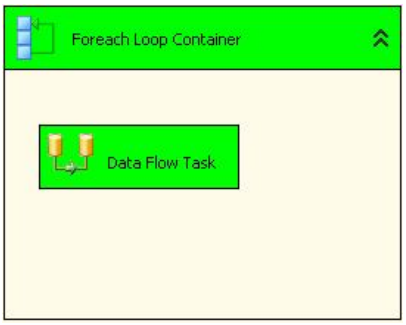

Figura 11. Resultados da execução do pacote na guia Control Flow

Figura 12. Resultados da execução do pacote na guia Data Flow

[ampliar] [abrir em janela]

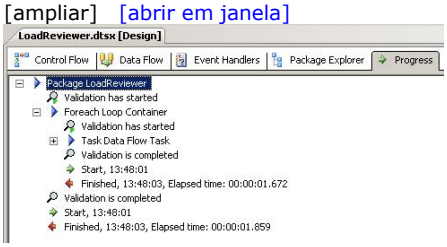

Figura 13. Resultados da execução do pacote na guia Progress

Deployment do pacote no SSIS

Para realizar a instalação do pacote criado no Integration Services será necessário executar a opção Build NetMag.ETL do menu Build. Isso criará o arquivo LoadReviewer.dtsx no diretório bin da aplicação.

Iremos agora utilizar o aplicativo SQL Server Management Studio, disponível na instalação do SQL Server. Se você tiver realizado a instalação padrão, ele estará disponível no menu Iniciar / Todos os Programas / Microsoft SQL Server 2008 / SQL Server Management Studio.

Na janela de logon escolha o serviço Integration Services para o campo Server type e clique em Connect. Na janela Object Explorer do Microsoft SQL Server Management Studio você verá a instância do Integration Services com dois diretórios. Expanda o diretório Stored Packages, clique com o botão direito no diretório File System e escolha a opção Import Package. Na janela Import Package escolha a opção File System para o Location e informe o caminho para o arquivo LoadReviewer.dtsx gerado após o build do Visual Studio. Clique em Ok para confirmar e fechar a janela Import Package.

Para executar o pacote vá à janela Object Explorer, expanda o diretório Stored Packages e em seguida expanda a opção File System. Clique com o botão direito no item LoadReviewer e escolha a opção Run Package. Na janela Execute Package Utility clique no botão Execute. A **Figura 14** mostra a janela de progresso de execução do pacote.

[ampliar] [abrir em janela]

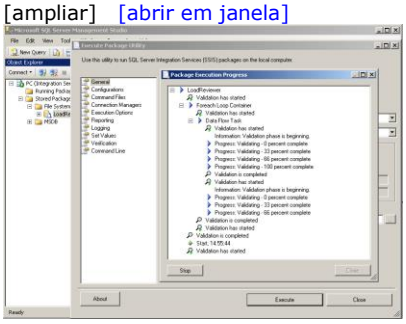

Figura 14. Progresso da execução do pacote instalado no Integration Services

Uma vez criado, um pacote pode ser utilizado sempre que necessário. No nosso cenário, toda vez que fossem disponibilizados novos arquivos Excel bastaria colocá-los no diretório de carga e executar o pacote do Integration Services. Para adicionar uma nova transformação utilizaríamos o Visual Studio, aplicando os controles disponíveis, repetindo o processo de criação e teste e, por fim, de deploy do pacote no ambiente de produção.

Em situações reais o Integration Services é capaz de realizar cargas de informações com elevado volume de dados e processamento, com grande desempenho e flexibilidade de customizações. Em alguns casos, a execução de um pacote do SSIS através de aplicativos pode ser necessária. O tópico a seguir mostra como podemos criar um job para ser chamado posteriormente por uma aplicação.

Criando um job para execução do pacote

Na maioria das situações um pacote instalado no servidor é bem mais útil quando podemos agendar sua execução, pois como vimos no tópico anterior, mesmo depois de instalado no Integration Services, seria necessária a figura de um operador para executar manualmente o pacote. O SQL Server Agent é o serviço capaz de executar automaticamente as atividades criadas (conhecidos como jobs) depois de agendadas.

Vamos criar um *job* para o nosso pacote, pois o utilizaremos no tópico a seguir para realizar execuções de pacote programaticamente por meio de uma aplicação. Para isso, siga os passos descritos a seguir:

1 - Abra o SQL Server Management Studio informando na tela de logon o valor Database Engine para o campo Server type. Informe os demais valores para autenticação e clique em Connect;
2 - Na janela Object Explorer expanda o nó que contém o serviço SQL Server Agent. Em seguida clique com o botão direito no nó Jobs e escolha New Job;
3 - Na janela New Job, informe o valor LoadReviewerJob para o campo Name;
4 - No painel Select a page, escolha a opção Steps clicando em seguida no botão New;
5 - Na janela New Step Job (**Figura 15**) informe o valor “Executar pacote LoadReviewer” para o campo Step Name e para o campo Type escolha a opção SQL Server Integration Services Package;

6 - Serão exibidos novos campos para escolha do pacote a ser executado. Na guia General escolha a opção SSIS Package Store para o campo Package source e informe o nome do servidor do Integration Services no campo Server. Para o campo Package clique no botão ao lado do campo e localize o pacote LoadReviewer na janela Select an SSIS Package;
7 - Feche todas as janelas confirmando as alterações.

Seria possível configurar diversas opções de agendamento através da opção Schedule da janela New Job exibida no passo 3, contudo para nosso exemplo será suficiente termos o job criado apenas. Veremos no tópico a seguir como podemos executar o pacote programaticamente através de uma aplicação.

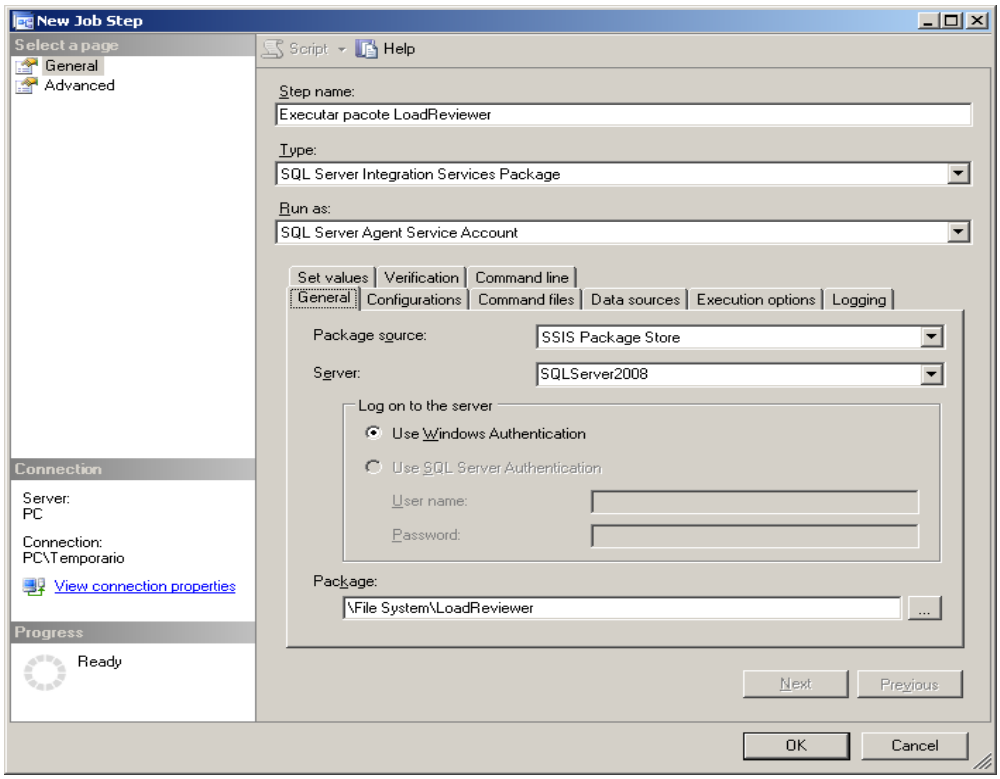

Figura 15. Janela de criação de novo passo para o job

Executando o pacote SSIS programaticamente

Vamos criar um novo projeto na solução NetMag.Solution, escolhendo a opção Add / New Project do menu File. Vamos adicionar uma aplicação do tipo Console Application com o nome NetMag.Console. Em seguida vamos adicionar uma nova classe a esse projeto chamada DTSX.cs. Adicione também ao projeto uma referência ao assembly Microsoft.SQLServer.ManagedDTS.dll. A classe DTSX deve ser codificada como mostra a **Listagem 1**.

Vamos entender agora o código. O método ExecutarPacoteLocal (linhas 10 a 17) possui um único parâmetro que receberá o caminho físico do pacote (no nosso exemplo, o arquivo LoadReviewer.dtsx do diretório bin do projeto NetMag.ETL).

Na linha 12 uma variável do tipo Application (classe contida no namespace Microsoft.SqlServer.Dts.RuntimeNI) se encarregará de acessar o pacote local que será passado para outra variável chamada Package capaz de executar o pacote informado. Na linha 14 o resultado da execução é atribuído a uma variável do tipo DTSExecResult e esta retornada pelo método.

O método ExecutarPacoteRemoto (linhas 19 a 65), por sua vez, será capaz de executar um pacote do Integration Services instalado no servidor na forma de um job, isso porque não podemos amarrar a execução do aplicativo ao término da execução do pacote, já que essa operação poderia onerar negativamente a performance da aplicação cliente. A solução é trabalhar de forma assíncrona, apenas indicando para o SQL Server Agent que um job deve ser iniciado. Para isso ele receberá dois parâmetros, o primeiro contendo a string de conexão com o banco de dados msdb do SQL Server e outro contendo o nome do job.

O propósito do método é realizar uma conexão com o banco de dados msdb do SQL Server e executar a stored procedure chamada sp_start_job, passando o nome do job desejado e obtendo como retorno um número inteiro que indica se o pacote foi iniciado com sucesso ou não, conforme vemos nas linhas 23 a 36. Nas linhas 40 a 44 a conexão é aberta e o comando disparado contra o banco de dados. O resultado é armazenado na variável jobResultado e analisado nas linhas 45 a 53.

Listagem 1

Listagem 1. Código da nossa classe, responsável por executar o pacote SSIS dentro de uma aplicação

```
1 using System;
2 using System.Data;
3 using System.Data.SqlClient;
4 using Microsoft.SqlServer.Dts.Runtime;
5
6 namespace NetMag.Console
7 {
8    public class DTSX
9    {
10       public string ExecutarPacoteLocal
         (string caminhoPacote)
11       {
12          Application app = new Application();
13          Package pacote = app.LoadPackage
            (caminhoPacote, null);
14          DTSExecResult resultado =
            pacote.Execute();
15
16          return resultado.ToString();
17       }
18
19       public string ExecutarPacoteRemoto
         (string connString, string nomeJob)
20       {
21          string resultado;
22
23          SqlConnection conn =
            new SqlConnection(connString);
24          SqlCommand command =
            new SqlCommand
            ("sp_start_job", conn);
25          command.CommandType =
            CommandType.StoredProcedure;
26
27          SqlParameter jobReturnValue =
            new SqlParameter(
28             "@RETURN_VALUE", SqlDbType.Int);
29          jobReturnValue.Direction =
            ParameterDirection.ReturnValue;
30          command.Parameters.Add
            (jobReturnValue);
31
32          SqlParameter jobParameter =
            new SqlParameter(
33             "@job_name", SqlDbType.VarChar);
34          jobParameter.Direction =
            ParameterDirection.Input;
35          command.Parameters.Add(jobParameter);
36          jobParameter.Value = nomeJob;
```

```
37
38          try
39          {
40             conn.Open();
41             command.ExecuteNonQuery();
42             int jobResultado =
43                (Int32)command.Parameters
                  ["@RETURN_VALUE"].Value;
44
45             switch (jobResultado)
46             {
47                case 0:
48                   resultado = "Pacote
                      iniciado com sucesso.";
49                   break;
50                default:
51                   resultado = "Pacote
                     falhou ao iniciar.";
52                   break;
53             }
54          }
55          catch (Exception exception)
56          {
57             resultado = exception.ToString();
58          }
59          finally
60          {
61             conn.Close();
62          }
63
64          return resultado;
65       }
66}
```

Antes de executar o aplicativo de console certifique-se de que a instância do SQL Server Database Engine esteja iniciada e também que os serviços do Integration Services e do SQL Server Agent estejam rodando. A **Listagem 2** exemplifica a chamada para os métodos criados. A linha 15 deve ser alterada com o caminho onde o arquivo LoadReviewer.dtsx foi criado na sua máquina.

Ao executarmos a aplicação console veremos que o primeiro método executa o pacote localmente, o que obriga a thread de execução da aplicação aguardar o seu término. Já na execução remota a aplicação apenas indica ao SQL Server Agent que o job LoadReviewerJob deve ser executado, contudo a operação é assíncrona sendo que a aplicação apenas aguarda um retorno da execução da stored procedure sp_start_job indicando se foi possível realizar o início da tarefa ou não.

Listagem 2. Código da classe Program, que realiza chamadas à classe DTSX

```
   1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace NetMag.Console
7 {
8    class Program
9    {
10       static void Main(string[] args)
11       {
12          DTSX client = new DTSX();
13          string result;
14
15          result = client.ExecutarPacoteLocal
               (@"C:\Documents and Settings\
               Leandro\Artigos \NetMag.Solution\
               NetMag.ETL\bin\LoadReviewer.dtsx");
16
17          System.Console.Write(result);
18          System.Console.ReadKey();
19
```

```
20          result = client.ExecutarPacoteRemoto
              (@"Data Source= (local);
              Initial Catalog=
              msdb;Integrated Security=SSPI",
              "LoadReviewerJob");
21          System.Console.Write(result);
22          System.Console.ReadKey();
23        }
24    }
25 }
```

Conclusão

Este artigo exemplificou através de um cenário simples como podemos utilizar as ferramentas de desenvolvimento do *SQL Server 2008* para elaborar sofisticadas soluções de *ETL*. É claro que os recursos do *Integration Services* não ficam limitados ao que foi apresentado aqui, você pode exercitar a partir deste exemplo outras situações de transformação de dados, adicionar novas tarefas e utilizar outros tipos de fonte de dados, tudo isso através do *SSIS Designer*.

Cabe relembrar que o Integration Services é apenas uma parte para chegarmos aos reais ganhos sugeridos no texto de introdução. Outros serviços do SQL Server devem ser utilizados para completar a solução criada nesse artigo.

Espero que a leitura tenha sido útil e que o contato com o ambiente de desenvolvimento do SQL Server 2008 possa direcionar novas possibilidades de solução para os problemas propostos no início desse artigo. Até o próximo!