SSIS 专题三：

SSIS 进阶技巧指南

SSIS 进阶技巧指南

SQL Server 2005 Integration Services（SSIS）提供一系列支持业务应用程序开发的内置任务、容器、转换和数据适配器。您无需编写一行代码，就可以创建 SSIS 解决方案来使用 ETL 和商业智能解决复杂的业务问题，管理 SQL Server 数据库以及在 SQL Server 实例之间复制 SQL Server 对象。在前两次 SSIS 技术专题中，我们介绍了关于 SSIS 的基础知识，大家已经有所了解。而在本次技术手册中，我们将介绍包括扩展 SSIS 包在内的进阶技巧，相信对那些已有 SSIS 使用经验的朋友也会有所帮助。

SSIS 包的扩展

SQL Server 整合服务（SSIS）是微软在 SQL Server 2005 中引入的数据转换服务（DTS）的替代物，它加载了大量不同的组件来导入数据和将数据转换为实际数据，而不仅仅是被动地导入。微软提供了两个基本的方法来扩展 SSIS 的功能：编写脚本和自定义对象编程。

- **通过编写脚本和程序来扩展 SSIS 包**

SSIS 与 VB.NET

SQL Server Integration Services（SSIS）提供了一个强大的提取、转换和加载数据的环境，同时，大多数时候我们都可以发现 SSIS 中的内置组件更胜任管理数据。然而，在某些情况下，我们可能需要扩展 SSIS 来满足特定需求。使用 Script 组件，我们可以添加自定义 Visual Basic.Net 代码到数据流上，从而在任意 SSIS 包中使用.Net 全部功能。

- 在 SSIS 中自定义 VB.Net 脚本（上）
- 在 SSIS 中自定义 VB.Net 脚本（下）

导出 SQL Server 2005 数据到微软 Excel

本部分介绍了从 SQL Server 2005 数据库单个表中提取数据导入 Excel 文件所使用数据流组件的配置方法和字符映射数据转换组件的用法。通过对本部分内容的掌握，您还可以实现选择 Excel 作为源，SQL Server 数据库作为目标的反方向的数据转移。

- 用 SSIS 和 Visual Studio 导出 SQL Server 数据到 Excel 文件（上）
- 用 SSIS 和 Visual Studio 导出 SQL Server 数据到 Excel 文件（下）

不验证而打开 SSIS 包

当我们在 SSIS Designer 中打开 SQL Server Integration Services（SSIS）包或者添加组件到一个包时，默认情况下 SSIS 会检查不同组件使用的数据源。这个对 SSIS 包验证过程可以确保外部元数据是有效的。如果元数据是无效的，那么我们将接收到警报或者错误消息。有时，我们可能也想不经验证就重写默认的行为和打开 SSIS 包。

- 不验证而使用 SQL 属性打开 SSIS 包（上）
- 不验证而使用 SQL 属性打开 SSIS 包（下）

SSIS 中的事务处理标签页

有很多关于创建 SQL Server 集成服务（SSIS）包的信息，都关注怎样开发控制流元素和数据流元素。众所周知，你在 SSIS 设计器的“控制流”标签页开发控制流元素，在“数据流”标签页开发数据流元素。SSIS 设计器还提供了“事件处理程序”标签页，它支持你设计基于包可执行体（包括文件，任务等）和它们生成事件的事件处理程序。

- SSIS 中使用事件处理程序的五个步骤（上）
- SSIS 中使用事件处理程序的五个步骤（下）

SSIS 中的渐变维度向导

在 SQL Server 之前的版本中，你不得不通过自定义逻辑来管理实现渐变维度的处理，这些自定义逻辑常常嵌入在数据转换服务（Data Transformation Services，DTS）包里。SSIS 2005 有一个转换功能，可以通过运行一个配置向导基本实现 SCD。

- 用 SSIS 2005 向导处理渐变维度(上)
- 用 SSIS 2005 向导处理渐变维度(下)

通过编写脚本和程序来扩展 SSIS 包

SQL Server 整合服务（SSIS）是微软在 SQL Server 2005 中引入的数据转换服务（DTS）的替代物，它加载了大量不同的组件来导入数据和将数据转换为实际数据，而不仅仅是被动地导入。但是，这里有一个问题，即使是新的 SSIS 工具也无法包括所有我们需要做的工作。

因此，微软提供了两个基本的方法来扩展 SSIS 的功能。其中一个方法是相对简单的，适用于没有很多的编程经验，也不想要编写复杂逻辑的用户；而另一个方法是较复杂的，它允许更有经验的程序员深入研究 SSIS 和更广泛的扩展 SSIS。

简单的方法：脚本

大多数人可能或多或少地熟悉脚本技术，并且 SSIS 使用 VB.NET 来允许程序员在 SSIS 包中用脚本编写行为。与自定义对象相反，脚本的范围是小而精的;它是用于我们对现有数据包允许或者已经实现的上下文进行一定修改的。

在 SSIS 数据包中，有两个元素我们可以用来添加脚本处理：Script Task（在 Integration Services Designer 应用的 Control Flow 窗口）以及 Script Component（在 Data Flow 窗口）。每一个都最好在稍微有点不同的环境中使用。

Script Task 是更适合用于软件包中通用目的的 Flow Control——它比 Script Component 更全局化更强大，但是也复杂得多。它在软件包的 Data Flow 之外运行，并且不受 Data Flow 工作方式的限制，虽然 Script Task 一般只当一个软件包被触发的时候才运行（虽然我们可以特别地编译）。Task 同样支持断点和调试，当我们编写一个带有控制逻辑或者执行某些决策的相当复杂的脚本时，这些是非常有用的。例如，Script Task 可以查询 Active Directory 获取一些信息或者与另外一个数据知识库进行数据交互——两者都可以在运行一个包之前进行。

Script Component 与 Data Flow 运作的方式结合得更紧密。Script Component 不是在整个包运行一次，它的主要过程是为每个正在处理的数据行运行一次。Script

Components 也同样有 3 个基本的运行上下文：数据源、数据转换或者数据目的地。Component 也同样比较少交互——比如，它并不支持 Script Task 那样的调试。Script Component 的最主要用途包括诸如一行行地数据转换、编译一个自定义 ODBC 目标、实时错误处理或者不通过原 SSIS 方法处理的转换操作。

高级方法：自定义对象编程

虽然脚本在 SSIS 包中的功能已经强大，但是，有时候它还是没有办法完成某些任务。在某些情况下，我们必须从头开始编写（或者让某人编写）一个自定义 SSIS 扩展。这并不是一件容易的事情;它要求对编程有全面的理解。但是，通过自定义的对象，我们可以使用 SSIS 进行远比简单自动化任务复杂的事情。

比如，如果我们有一个不能支持任何现有 SSIS 转换的数据源（例如，有些奇怪的专有数据源），那么我们可以写入一个自定义的连接管理对象来像使用本地数据源一样使用该数据。与此类似的是，我们可以用由 SSIS 提供的相同的程序库来创建自定义任务、日志组件或者数据流组件。

上面每个类型的项目都是可以作为 SSIS 所支持语言的一个基类、属性和方法集：Visual Basic、C#、C++、J# 和 Jscript. C++、C# 和 VB 都倾向于创造最好的结果，因为在这些上下文中它们都倾向于由开发人员和供应商两者同时广泛支持。关键在于我们所使用的语言不能是一个阻碍;它们都可以插入相同的对外编程接口中。我们同时也可以在需要的时候通过标准 Windows 形式创建自定义对象的用户接口。

SSIS 自定义对象可以创建的一个极其强大的自定义 Foreach 遍历器。假设你需要创建一套编程类来为一个集合中每个对象执行特定的操作，比如数据库中的表。如果你想在大量的上下文中执行这个操作而不重复编写代码，那么这是其中的一个最佳方法。特别是当你已经对一些新的数据类型创建了一个自定义的连接管理器（与上面的例子一样），并且想创建一个自定义 foreach 操作来处理时，这种方法是非常有用的。

总结

我们所选择的扩展 SSIS 的方法，不管是脚本或者是编程，都是可以根据我们的需要和能力而决定的。因为，我们可以使用这两种方法——甚至是同时使用！——我们并不需要为我们手头上的任务做额外的工作。我们也可以按照要求混合使用。

关于更多的例子，当然，你可以查看上面的连接（直接连接到微软的 SSIS 包扩展文档），虽然它们主要阐述具体的实现和例子。同时，还有一些关于这个主题的博客：例如，Peter DeBetta's SQL Programming Blog。我也可以推荐阅读 SSIS Junkie 博客，这些上面的文章都有关于如何在 SSIS 中实现这些方法（包括编程和其它的）的探讨。

（作者：Serdar Yegulalp 译者：曾少宁 来源：TT 中国）

在 SSIS 中自定义 VB.Net 脚本（上）

SQL Server Integration Services（SSIS）提供了一个强大的提取、转换和加载数据的环境，同时，大多数时候我们都可以发现 SSIS 中的内置组件更胜任管理数据。然而，在某些情况下，我们可能需要扩展 SSIS 来满足特定需求，这就是 Script 组件派上用场的地方了。使用 Script 组件，我们可以添加自定义 Visual Basic.Net 代码到数据流上，从而在任意 SSIS 包中使用.Net 全部功能。

在本文中，我将演示如何添加 Script 组件到一个 SSIS 包上。我所使用的示例包在控制流中包含一个数据流任务。数据流任务配置了 3 个组件：OLE DB Source, Script 组件和 Flat File Destination。图 1 显示了 Visual Studio 中显示在 Data Flow 标签中的这些组件。

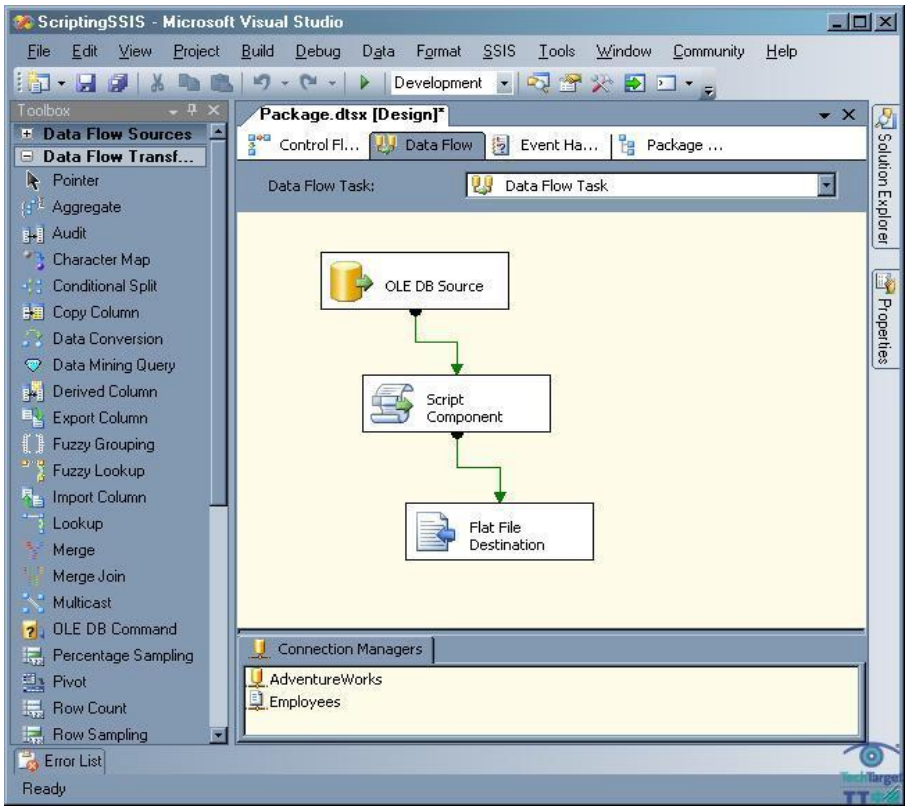


图 1：Visual Studio 中的 Data Flow 视图。

Script 组件将根据员工的名、中间名和姓以及它们入职年月生成用户名。我从 SQL Server 2005 附带的 AdventureWorks 示例数据库中提取了员工信息。注意，SSIS 同样支持在控制流中使用 Script 任务，然而，本文只关注在数据流中使用的 Script 组件。

虽然我的关注点是 Script 组件，但是我也将提供一些关于如何配置其它组件的信息，以便你可以用来创建自己的解决方法。如果我们需要帮助来创建 SSIS 包或者配置组件，那么可以查看 SQL Server 2005 Books Online。

配置 OLE DB Source 组件

OLE DB Source 组件使用本地 OLE DB 连接管理器来连接到 AdventureWorks 数据库上。此外，组件指定了下面的 Transact-SQL 命令来从 Person.Contact 表和 HumanResources.Employee 表中检索数据：

```
SELECT c.FirstName, c.MiddleName,
c.LastName, e.HireDate
FROM Person.Contact c
INNER JOIN HumanResources.Employee e
ON c.ContactID = e.ContactID
```

正如所看到的，我根据 ContactID 字段在两个表中创建了一个简单的连接运算（Join）。我所检索到的信息将用来创建自定义的用户名。

配置 Script 组件

在我们配置了 OLE DB Source 组件之后，我们已经可以配置 Script 组件了。当添加一个 Script 组件时，我们必须指定是否使用这个组件作为一个源（Source）、转换（Transformation）或者目标（Destination）。在这个例子中，我将 Script 组件作为一个转换来使用，因为我想在数据流中拦截数据，并根据该数据生成一个包含用户名称的额外的字段。换言之，我准备在数据从源传输到目标时对该数据进行转换。

在我们添加了 Script 组件之后，就连接 OLE DB Source 组件到 Script 组件的数据流路径。接着，双击 Script 组件，打开 Script Transformation 编辑器。图 2 显示了编辑的 Input Columns 页面。在该页面中，我们通过选择字段名称旁边复选框来指定哪些字段可作为 Script 组件的输入使用。对于这个例子，我选中了所有的 4 个字段。

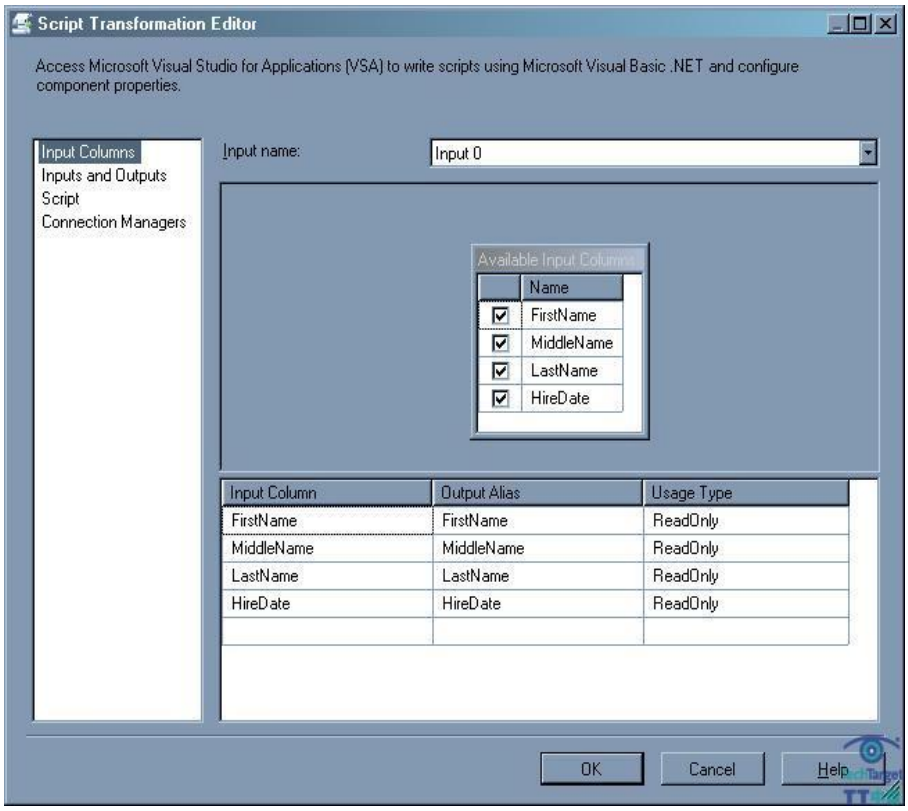


图 2：编辑器的 Input Columns 页面。

接下来，选择 Script Transformation 编辑器的 Inputs and Outputs 页面（如图 3 所示）并展开 Input 0 和 Output 0 节点。注意，我们在 Input Columns 页面中所选择的输入字段也同样在此显示。

我们必须添加一个输出字段来保存脚本将生成的用户名。为了添加一个字段，可以在 Output 0 节点中选择 Output Columns 子节点，点击“Add Column”，接着输入“UserName”作为字段名称。在 Inputs and Outputs 页面右边的“Data Type Properties”选项中，指定字符串作为数据类型和长度为 8。

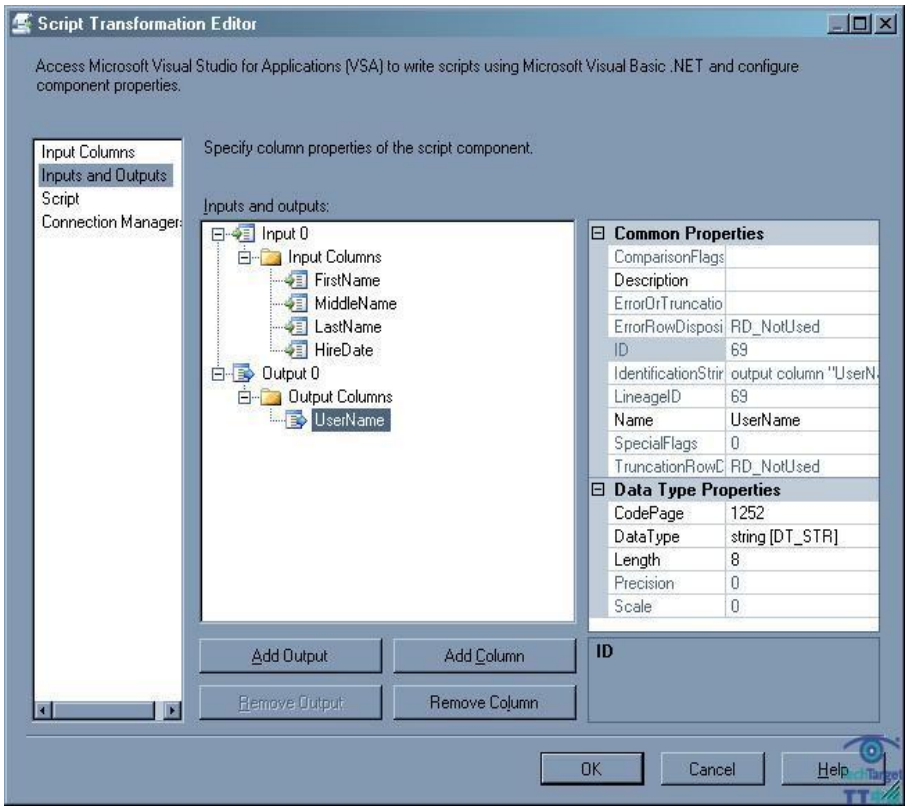


图 3：Script Transformation 编辑器的 Inputs and Outputs 页面

在这个例子中，这些就是在 Script 组件中我们所需要配置的。如果我们打算在脚本中使用系统或者包变量，那么我们必须添加这些到编辑器的 Script 页面，然而，这个方法并不要求这些类型的变量。

（作者：Robert Sheldon 译者：曾少宁 来源：TT 中国）

在 SSIS 中自定义 VB.Net 脚本（下）

添加 Visual Basic.Net 代码

现在我们可以添加实际的代码到 Script 组件上了。打开 Script Transformation 编辑器的 Script 页面，点击“Design Script”。 “Microsoft Visual Studio for Applications”窗口会出现，并且还包含了我们需要使用的初始 Visual Basic.Net 代码，如图 4 所示。

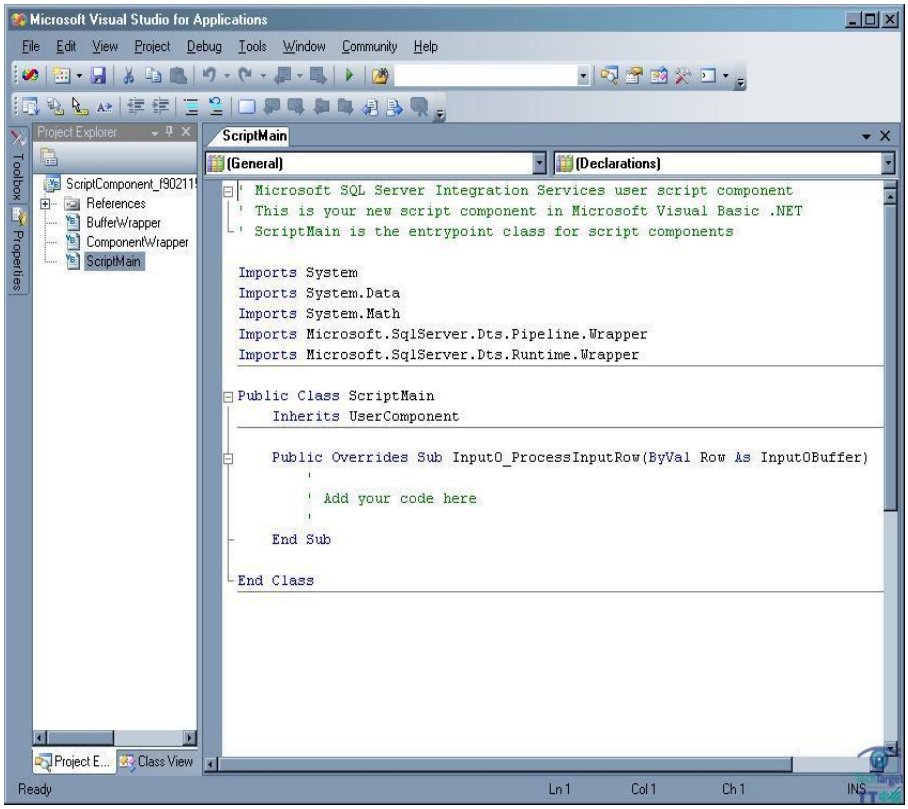


图 4：带有 VB.Net 代码的“Visual Studio for Applications”窗口。

代码开头是几个关于 Script 组件的注释。注释都是跟在一个单引号后面。我们可以删除这些注释然后添加其它注释上去。注意，此处生成的 Visual Basic 代码仅限于 Script Transformation 组件的。当作为源或者目标时，组件生成的代码是不同的。

自动生成的代码以一系列的 Imports 语句开始来定义我们可能要在脚本中使用的初始命名空间。我们可以根据需要添加更多的命名空间。ScriptMain 类跟在 Imports 语句之后，然后是类中的 Input0_ProcessInputRow（）方法。我们必须以这个类和方法开始，然后再添加我们的自定义代码到方法中，此处的注释说明："在此处添加你的代码"。

正如我们所看到的，SSIS 让一切变得简单多了。只需要在指定的地方上输入我们的代码。对于本文中所探讨的方法，我首先定义了一组变量，然后将变量合并在一起来创建用户名。你可以点击此处下载所有 Visual Basic.Net 代码。

首先，我创建名字、中间名和姓变量，它们分别是 First、Middle 和 Last。在第一行中，我声明了 First 变量类型为字符串，接着，在第二行中，我给它赋值为名字的第一个字母：

```
Dim First As String
First = LCase (Row.FirstName.Substring (0, 1) )
```

注意，我使用了 Row 变量——它是在 Input0_ProcessInputRow（）方法中定义的——来从输入中取回 FirstName 字段。当我取到这个名字时，我使用了 Substring（）方法来只截取结果的第一个字母。接着，我将结果传递给 LCase（）方法，将字母全部转换为小写。

对于 Middle 变量，我执行类似地操作，但是我首先检查了 MiddleName 值是否是一个 NULL 值：

```
Dim Middle As String
If Row.MiddleName_IsNull Then
Middle = ""
Else
Middle = LCase (Row.MiddleName.Substring (0, 1) )
End If
```

这里我使用一个 If-Else-End If 结构来判断值是否为 NULL。如果它是一个 NULL，那么我将 Middle 值设置为空的字符串。否则，我会跟处理名字一样取回第一个字母。

对于 Last 变量，我希望从 LastName 字段中选取前面 4 个字母。然而，我需要检查名称包含的字母是否少于 4 个，以及是否包含一个单引号：

```
Dim Last As String
If Row.LastName.Length < 5 Then
```

```
If Row.LastName.Contains ("'") Then
Last = LCase (Row.LastName.Replace ("'", "") )
Else
Last = LCase (Row.LastName)
End If
Else
If Row.LastName.Contains ("'") Then
Last = LCase (Row.LastName.Replace ("'", "") .Substring (0, 4) )
Else
Last = LCase (Row.LastName.Substring (0, 4) )
End If
End If
```

首先，我在最外部使用一个 If-Else-End If 条件来判断名字中包含的字母是否少于 5 个。如果是，那么我使用整个姓的部分。如果不是，那么我只使用它前面 4 个字母。

为了处理好名字可能包含一个单引号的情况，我在外部的 If-Else-End If 条件内再嵌入一个 If 和 Else 语句。如果名字中包含一个单引号，那么我会用一个空的字符串替代它。否则，我会使用名字本身。

最后，我将声明和赋值 YearHired 变量，它从 HireDate 字段中获取年份：

```
Dim YearHired As String
YearHired = CStr (Row.HireDate.Year) .Substring (2, 2)
```

注意，我首先使用 Row 变量来检索 HireDate 字段中的 Year 属性。接着，我使用 CStr（）方法来将日期转换为字符串，然后使用 Substring（）方法来检索年的最后两个数字。

在我定义了这些变量之后，我通过组合这些变量来创建用户名：

```
If Row.HireDate < CDate ("2000-01-01") Then
Row.UserName = First + Last + YearHired
Else
Row.UserName = First + Middle + Last + YearHired
End If
```

再次，我使用了 If-Else-End If 条件来确定使用哪个逻辑。对于 2000 年之前所雇用的员工，我根据名字的首字母、姓的前 4 个字母和雇用年份的最后两位数字来创建一个用

户名。对于其他人，我还将中间名的首字母包括在内，除此之外其它都使用了相同的逻辑。我将用户名称赋值到 UserName 字段上，这样，它就可以与其它的字段一样通过 Row 变量来读取了。

这就是添加一个 Script 组件到数据流上所需要的操作。在我们添加了代码之后，就可以关闭 Visual Studio for Applications 窗口以及 Script Transformation 编辑器。这样，我们就可以添加我们的目标组件了。

配置 Flat File Destination 组件

在我们添加了 Flat File Destination 组件之后，就连接从 Script 组件到目标的数据流路径，同时打开 Flat File 目标编辑器。现在添加一个指向文本文件的 Flat File 连接管理器。对于这个例子，我使用 c:\employees.txt，并将原来的 4 个字段和一个新的 UserName 字段映射到目标上。这里所有项我都使用了默认的设置。在我们完成了之后，关闭所有打开的编辑器并保存 SSIS 包。现在运行包。员工的信息——以及新的用户名——将被添加到文本文件中。

当然，此处我所探讨的方法只是一个非常基本的例子。然而，它确实可以演示添加自定义代码到 SSIS 需要的所有组件。你也可以尝试创建其它类型的脚本和添加脚本源和目标。而且你会发现，Script 组件能够访问大量的 Visual Basic.Net 开发环境，并且还允许你扩展 SSIS 包来实现所需要的任何数据提取、转换和加载（ETL）。

（作者：Robert Sheldon 译者：曾少宁 来源：TT 中国）

用 SSIS 和 Visual Studio 导出 SQL Server 数据到 Excel 文件（上）

本文选自 Jayaram Krishnaswamy 编写的《在 Visual Studio 2005 中使用 SQL Server 集成服务（SSIS）初学者指南》一书，讲述了创建 SSIS 包来实现把 SQL Server 2005 数据库表中数据导入到微软 Excel 电子表格的基础知识。

你也会学到一些关于使用字符映射数据转换组件的基础知识。在动手练习过程中，你会练习从 SQL Server 2005 表转移数据到微软 Excel 2003 数据表文件中。你将使用由"基于连接管理的连接到 SQL Server 2005 服务器的源组件"和"连接到 Excel 连接管理器的 Excel 目标组件"组成的数据流任务。

为了执行下列步骤，你需要一个源数据组件和一个目标数据组件：源数据是从 MyNorthwind 数据库中提取的（该数据源只是 Northwind 数据库重命名后的版本），该数据库运行于 SQL Server 2005 服务器;我们的目标是加载该数据到微软 Excel 2003 数据表文件中。你还需要建立一条路径来连接它们。此外，你也还要插入一个字符映射数据流任务，它会转换其中一个数据字段内部的文本，这样一来，该列所有字符转换后都变成了大写形式。

把 SQL Server 数据转移到 Excel 文件

第一步：创建 SQL Server BI 项目并添加一个数据流任务

第二步：配置数据读取组件（DataReader）的连接管理器

第三步：为 SQL Server 数据设置数据读取源组件

第四步：给 SQL Server 数据转换组件设置字符映射机制

第五步：添加 Excel 目标组件并创建连接到字符映射组件的路径

第六步：配置微软 Excel 目标组件

第七步：测试从 SQL Server 表到 Excel 的数据转移

第一步：创建 SQL Server BI 项目并添加一个数据流任务

在这一部分，你将创建一个商业智能项目，然后修改默认包对象的名称。既然我们要做的事与数据有关，你将会添加一个数据流任务，还会给数据流中添加一个数据读取组件（DataReader）。

按照第二章或第三章的介绍，创建一个商业智能项目 Ch5。

把默认包名称 Package.dtsx 改为 TableToXls.dtsx。

从工具箱中拖拽一个数据流任务组件到控制流页。

点击打开数据流标签页，该标签显示了数据流页。

现在你就可以访问工具箱中的数据流选项了。这些选项有：数据流源组件，数据流目标组件，数据流转换组件（参考第一章）。

从数据流源组件组中拖拽数据读取（DataReader）源组件到数据流页面。

第二步：配置数据读取组件（DataReader）的连接管理器

关于配置连接到本地 SQL Server 2005 数据读取（DataReader）源组件的方法已介绍过了。这里只列出几个与本章需要展示的内容相关的步骤。

在连接管理器页面下方单击右键，然后从右键菜单中选择“新建 ADO.Net 连接……”。

如果你是在读过第四章（并做过其中的练习）以后才进行本章的练习，你会看见配置 ADO.NET 连接管理器屏幕显示之前配置的连接管理器。如果你需要创建新的连接配置，你可以按照上一章中讲的步骤进行。

在配置 ADO.NET 连接管理器窗体中点击确定（OK）按钮。

连接管理器“Localhost.MyNorthwind.sa”就被添加到连接管理器页面了。

第三步：为 SQL Server 数据设置数据读取（DataReader）源组件

在下拉菜单中右键点击数据读取源组件（DataReader）。

在下拉菜单中选择编辑（Edit），就可以打开数据读取（DataReader）源组件高级编辑器。首先，你需要指定一个数据读取组件（DataReader）可用的连接管理器。

在刚显示的数据读取（DataReader）源组件高级编辑器中，点击连接管理器标签。

点击列表标题连接管理器下面的空白区域（灰色区域），在这里，你会看到你在第一步中添加的连接管理器。

选择这个连接管理器。

接下来，点击组件属性标签打开数据读取组件（DataReader）属性。在这里，你会发现它要求填写一个 SQLCommand（目前唯一空白的填充项）。

点击它旁边的省略号按钮会显示一个文本编辑器，在那里面你可以写入 SQL 命令。

你可以直接写上下面这句 SQL 命令：

```
SELECT CustomerID, CompanyName, Address, City, PostalCode

FROM Customers
```

点击刷新按钮。

这个查询意思是让 DataReader 读取 5 列字段的数据。下图是从 SQL Server 2005 Management Studio 中得到的一个表数据示例。如果你能回想起来，上一章我们也用过同样的这些字段列。

点击列映射标签。

打开的列映射页面中会显示 DataReader 输出的列。

在编辑器的最后一个标签“输入和输出属性”中，你可以从外部列，输出列和 DataReader 错误输出中添加或者删除项目。在本教程中，我们不做修改。

	CustomerID	CompanyName	Address	City	PostalCode
1	ALFKI	Alfreds Futterkiste	Obere Str. 57	Berlin	12209
2	ANATR	Ana Trujillo Emparedados y ...	Avda. de la Constitución ...	México D.F.	05021
3	ANTON	Antonio Moreno Taquería	Mataderos 2312	México D.F.	05023
4	AROUT	Around the Horn	120 Hanover Sq.	London	WA1 1DP

到此为止，配置 DataReader 从 SQL Server 2005 数据库中提取五个字段列的工作就完成了。

（作者：Jayaram Krishnaswamy 译者：冯昀晖 来源：TT 中国）

用 SSIS 和 Visual Studio 导出 SQL Server 数据到 Excel 文件（下）

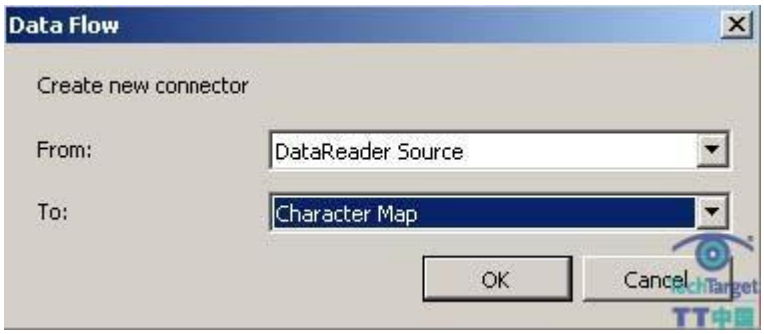
第四步：为 SQL Server 数据转换组件设置字符映射

字符映射转换组件是在第一章中讲到的，但是在这里我们要体验一下它的转换功能。转换组件处理接收到的文本字符串，输出为转换后的字符串。例如：在上面我们看到的图片中，公司名是大小写混合的，使用转换组件后，我们会在数据写到 Excel 之前，把出现在公司名这一列的所有字符转换为大写——例如：Alfreds Futterkiste 会转换为 ALFREDS FUTTERKISTE。

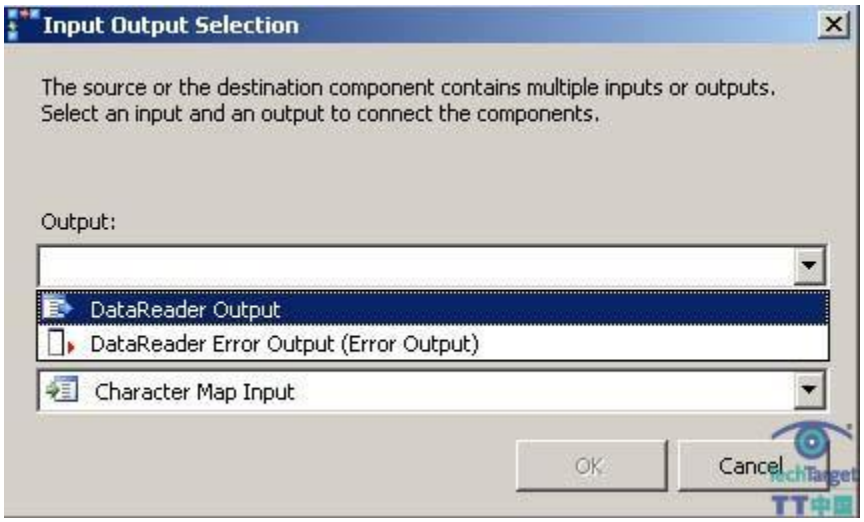
在工具箱的数据流转换组件组里选择字符映射数据流项，把它拖拽到数据流页面的设计画布上。

在 DataReader 源组件上点击右键，在右键菜单项中点击添加路径。

在弹出显示的数据流窗口中，选择转换目标为字符映射。如下图所示：



在上面的窗口中点击确定（OK），此时会显示下面的新窗口。



在这里你需要指定这个组件的输出项，从下拉列表里选择即可。

从下拉列表里选择 DataReader 输出源。

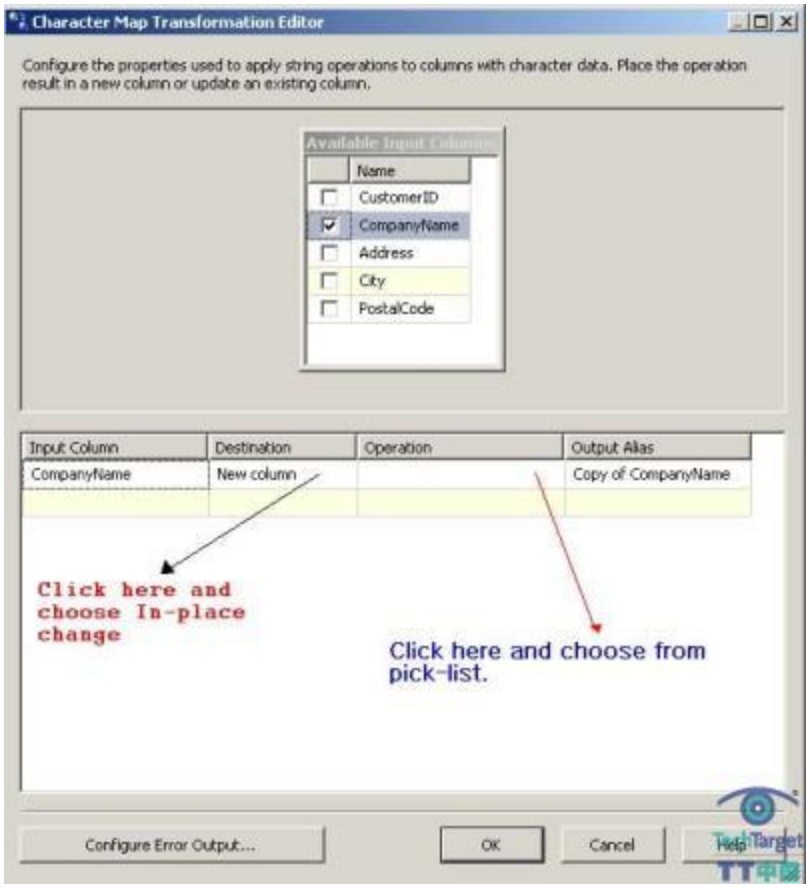
现在，在本教程中不涉及任务错误处理。选好上面的选项后，点击确定（OK）按钮就生效了。

在上面的窗口中点击确定（OK）按钮。

这就建立了从 DataReader 源到字符映射数据流组件的通道。通道建立好了，但仍然需要进一步配置。

在字符映射组件上点击右键，在右键菜单中选择编辑。

此时会弹出如下图所示的字符映射编辑器。选中列公司名前面的复选框。

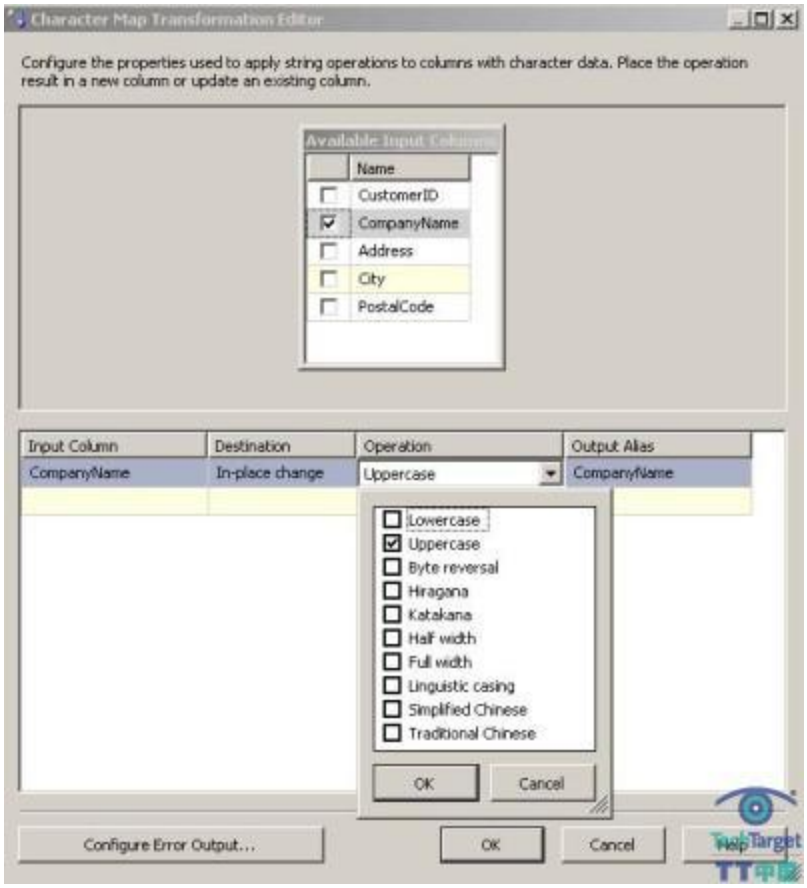


如果如上图所示的默认就被选中了，那么输出部分就会多出来一列。我们从下拉列表中选择“In-place change”选项。

点击上面窗体中目标列下面的新建列单元格。在下拉列表中选择“In-place change”。

在操作（Operation）下面点击，在出现的下拉列表中选中大写。如下图所示。

输出别名不用修改，因为它在选中公司名时已经被关联修改了。



在选择列表中点击确定（OK）按钮，然后点击字符映射转换编辑器中的确定（OK）按钮。

字符映射配置就完成了。

第五步：添加 Excel 目标组件并给字符映射组件创建路径

在这一步中，我们将添加一个 Excel 目标组件。然后，我们会建立一个从字符映射组件到 Excel 目标组件的路径。

从工具箱的数据流目标组件组选择 Excel 目标组件，把它添加到数据流页面上。

可以通过在工具箱中双击该组件或者通过拖拽操作来完成这一步。在字符映射组件上右击，在弹出的邮件菜单中选择添加路径。

此时会打开一个数据流窗口，在这里你可以建立一个数据流路径，显示“From：”路径为字符映射。

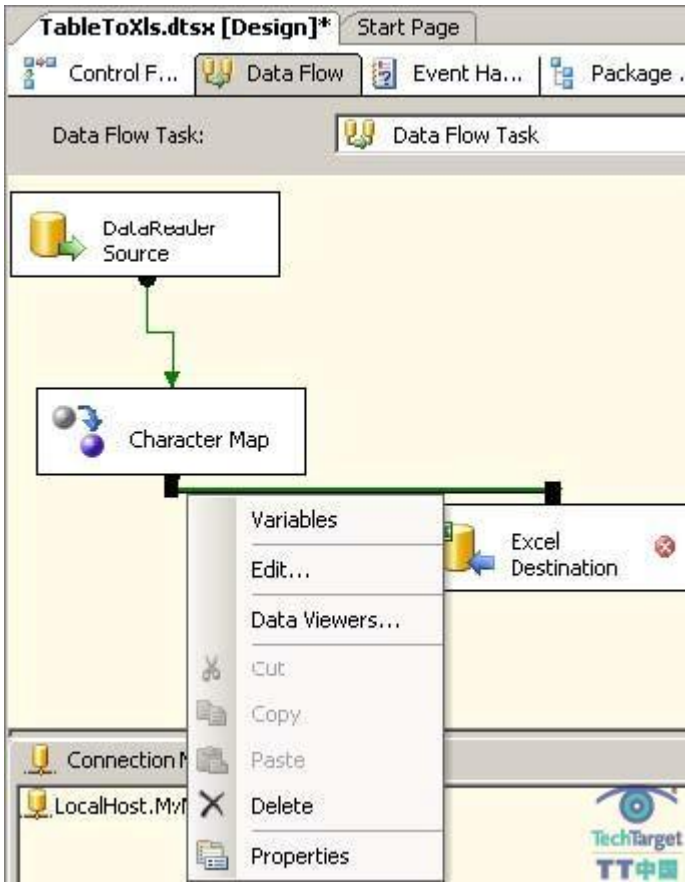
点击“To：”旁边的下拉列表，会显示 Excel 目标组件和 DataReader 源组件。

选择 Excel 目标组件，然后点击屏幕上的确定（OK）按钮。

此时会打开一个输入输出选择窗口，窗口中会显示可用的输出和输入。输出窗口是空的，而输入显示 Excel 目标输入。路径应该从字符映射到 Excel 目标输入连接。

选好上面的选项，然后点击屏幕上的确定（OK）按钮。

你会看到有一条绿色的线从字符映射数据流组件连向 Excel 目标组件，如下图所示。建立连接路径的过程也可以被简化，可以通过在数据流页面上从字符映射到 Excel 目标组件拖拽一条绿箭头完成，与前面的做法一样的效果。如下图所示，你可以通过右击这条绿色箭头线，在右键菜单中选择”编辑……”来编辑路径。



第六步：配置微软 Excel 目标组件

前面步骤中的绿箭头代表了数据传递的路径。Excel 目标组件也需要一个连接管理器。

Excel 目标组件连接在连接管理器中定义的链接属性来连接你硬盘上的微软 Excel 文件。

在 Excel 目标组件上右击，在右键菜单中选择编辑。

此时会弹出 Excel 目标组件编辑器。Excel 需要一个 OLEDB 连接管理器，如果（你或者其他之前的用户）没有配置连接管理器，下拉列表就是空的。

点击新建按钮。

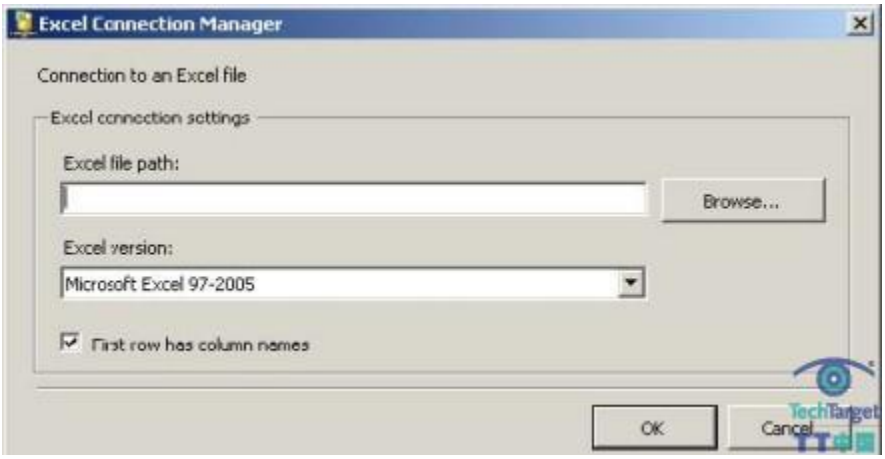
Excel 连接管理器窗口如下面的图片显示。这里，你需要使用浏览按钮选择一个 Excel 文件作为转移数据的目标。包程序运行时，数据会被写到这个目标文件里。

打开 Windows 资源管理器，在 C 盘创建一个 Excel 文件。在本教程中，我们创建一个名为 TableToXls.xls 的 Excel 文件。

除了连接到一个已经存在的文件，Excel 连接管理器还支持在你用浏览按钮选择的本机文件夹创建文件。

现在就用浏览按钮找到刚刚创建的文件，并选中该文件。

在 Excel 连接管理窗口中点击确定（OK）按钮。

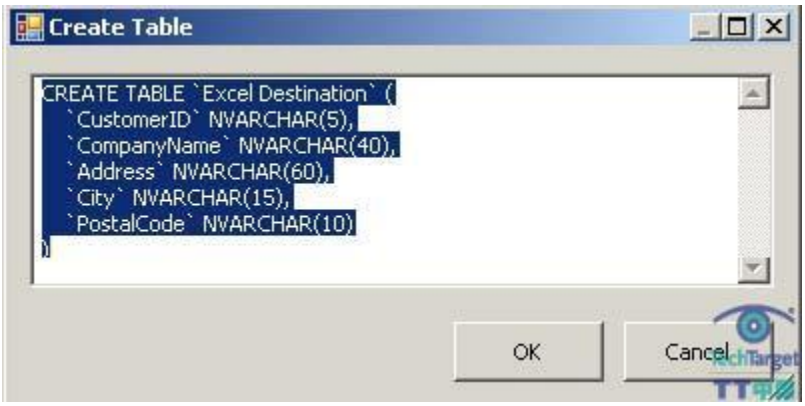


对于数据访问模式这一项，采用默认模式，表或者视图。

你必须指定要使用的 Excel sheet 页的名称。（不要点击下拉列表来指定 sheet 页，下拉列表中会显示新创建 Excel 文件的三个 Excel sheet 页，它们都只有一列。）

点击新建按钮。

你现在正在创建一个新的 Excel sheet 页。弹出的创建表格的窗口中显示了即将导入组件的列，如下图所示。

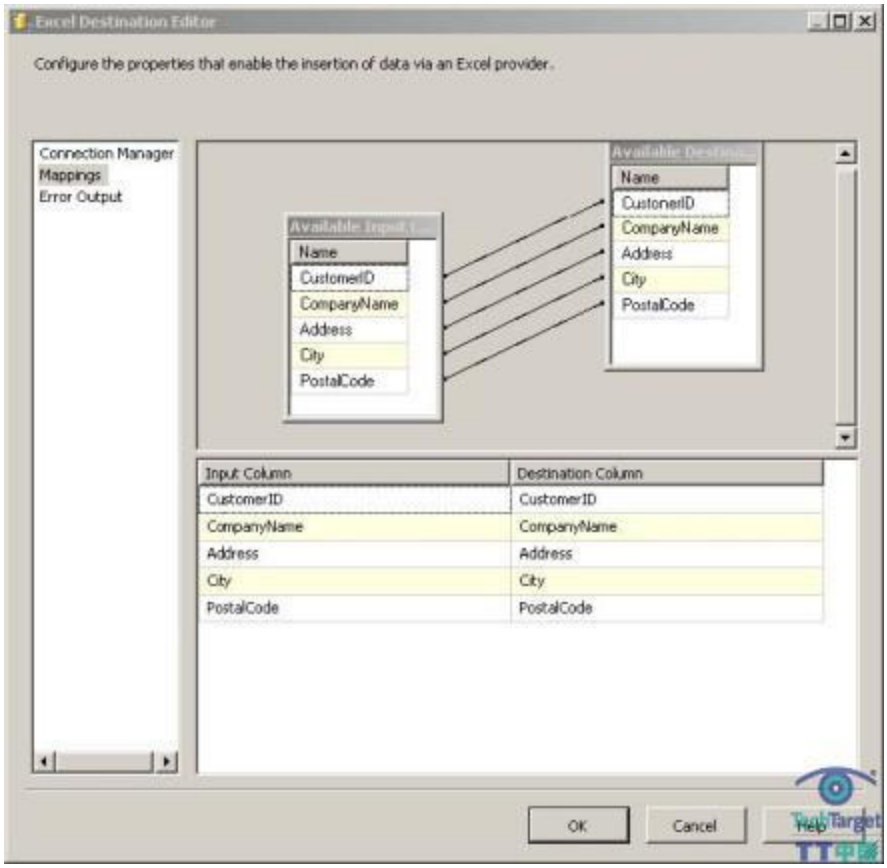


点击上图中的确定（OK）按钮。

一个新的名为 Excel Destination 的 Excel sheet 也会被添加到 TableToXLS.xls 文件中。如果你打开检查一下该文件（TableToXLS.xls），你会看到列表的表头都已经添加到该页中了。

在 Excel 目标组件编辑器面板左侧点击映射，会在右侧显示从输入到输出的映射。

这里显示了从字符映射数据流组件导向目标文件的所有列，如下图所示。



点击该窗口的确定（OK）按钮。现在，包已经完全配置好，可以运行了。

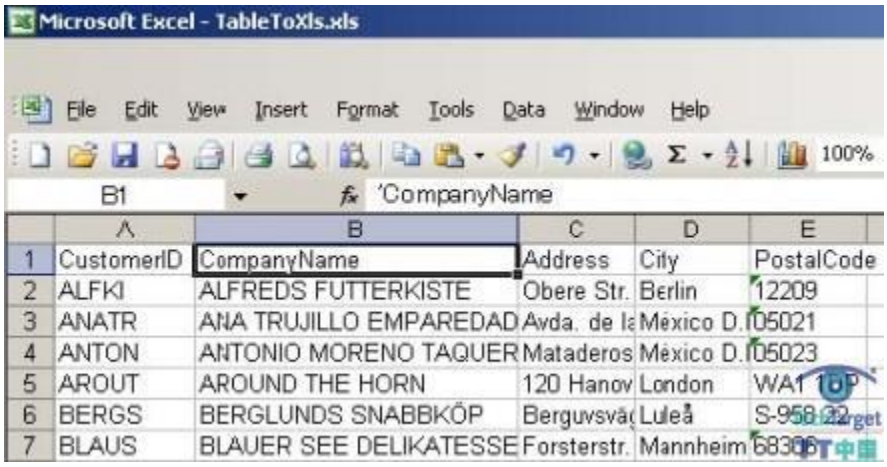
第七步：测试从 SQL Server 数据库表向 Excel sheet 页执行数据转移

在解决方案浏览器中右击，在右键菜单中点击执行包。

数据流页面上的三个组件都变成了绿色，这表示包执行成功，没有错误。你可以检查“Canvas”中的进度标签，那里会显示包执行过程的所有细节。

现在打开 TableToXLs.xls 文件看看。

下图中显示了该文件中的几行数据。注意字符映射数据转换组件已经把公司名这一列的所有字符转换为大写了。



	A	B	C	D	E
1	CustomerID	CompanyName	Address	City	PostalCode
2	ALFKI	ALFREDS FUTTERKISTE	Obere Str.	Berlin	12209
3	ANATR	ANA TRUJILLO EMPAREDAD	Avda. de la	México D.	05021
4	ANTON	ANTONIO MORENO TAQUER	Mataderos	México D.	05023
5	AROUT	AROUND THE HORN	120 Hanov	London	WA1 1DP
6	BERGS	BERGLUNDS SNABBKÖP	Berguvsvä	Luleå	S-958 22
7	BLAUS	BLAUER SEE DELIKATESSE	Forsterstr.	Mannheim	68306

总结

本章讲述了以下内容：

- 从 SQL Server 2005 数据库单个表中提取数据导入 Excel 文件所使用数据流组件的配置方法。

- 字符映射数据转换组件的用法。

选择 Excel 作为源，SQL Server 数据库作为目标可以实现反方向的数据转移。

（作者：Jayaram Krishnaswamy 译者：冯昀晖 来源：TT 中国）

不验证而使用 SQL 属性打开 SSIS 包（上）

当我们在 SSIS Designer 中打开 SQL Server Integration Services（SSIS）包或者添加组件到一个包时，默认情况下 SSIS 会检查不同组件使用的数据源。这个对 SSIS 包验证过程可以确保外部元数据是有效的。如果元数据是无效的，那么我们将接收到警报或者错误消息。有时，我们可能也想不经验证就重写默认的行为和打开 SSIS 包。SSIS 支持两种属性——DelayValidation 和 ValidateExternalMetadata——它们可以控制单个组件上的认证过程。

创建基本的 SSIS 包

一个常见的需要控制验证过程的情况是当在一个 SSIS 包中创建数据库对象，接着要在这个包中访问这些对象的时候。图 1 显示了某一个 SSIS 包，它创建一个表，填充数据，接着读取该表。

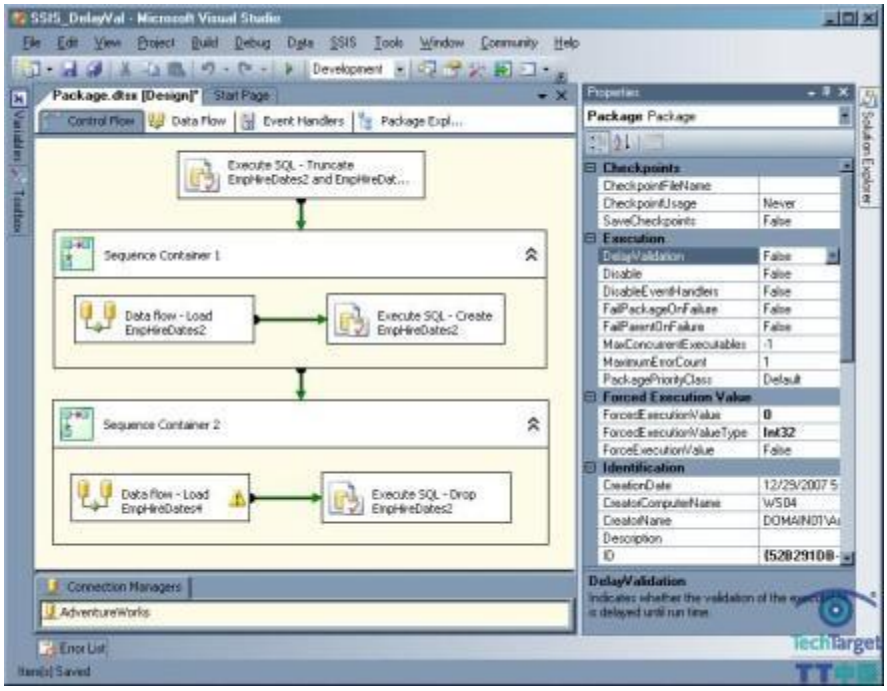


图 1：SSIS 数据包创建和填充一个表，然后读取该表。

为了支持这个包，我首先创建一组表，这些表除了名称之外其它都是一样的。

```
SELECT c.FirstName, c.MiddleName,
c.LastName, e.HireDate
INTO HumanResources.EmpHireDates1
FROM Person.Contact c
INNER JOIN HumanResources.Employee e
ON c.ContactID = e.ContactID
SELECT * INTO HumanResources.EmpHireDates2
FROM HumanResources.EmpHireDates1
SELECT * INTO HumanResources.EmpHireDates4
FROM HumanResources.EmpHireDates1
```

这个包开始执行一个 Execute SQL 任务，截取 EmpHireDates2 和 EmpHireDates4 表。第一个 Data Flow 从 EmpHireDates1 表中检索数据并将数据插入到 EmpHireDates2 表。在 Data Flow 任务之后，有一个 Execute SQL 任务基于从 EmpHireDates2 中取出的数据创建 EmpHireDates3 表。在这个任务中，我使用下面的 Transact-SQL 语句：

```
SELECT * INTO HumanResources.EmpHireDates3
FROM HumanResources.EmpHireDates2
```

第二个 Data Flow 任务从 EmpHireDates3 表检索数据并将数据插入 EmpHireDates4 表。接着，跟在这个数据流任务后面的 Execute SQL 任务删除 EmpHireDates3 表。

正如我们在图 1 中所看到的，第二个 Data Flow 任务显示一个验证警告（黄色三角形）。这是因为 EmpHireDates3 表已经不存在了，但是该表仍然被一个数据流组件所引用。图 2 显示这个 Data Flow 任务的 Data Flow 标签页。注意，引用了 EmpHireDates3 表的 OLE DB 源显示了一个错误。

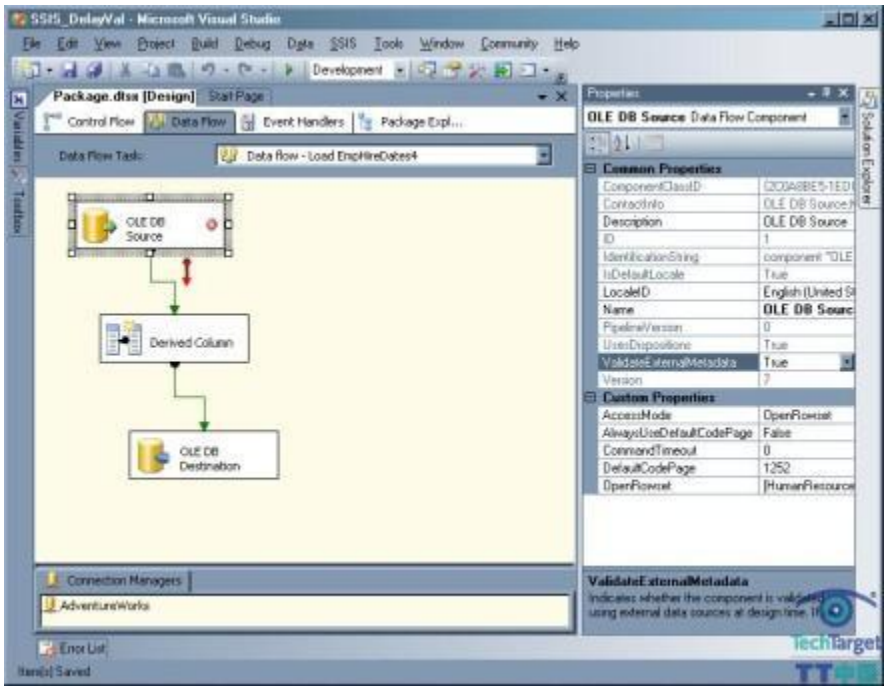


图 2：Data Flow 任务的 Data Flow 标签页。

（作者：Robert Sheldon 译者：曾少宁 来源：TT 中国）

不验证而使用 SQL 属性打开 SSIS 包（下）

如果我们运行这个 SSIS 包，那么我们将接收到一个显示数据流中存在一个错误的验证错误消息（如图 3 所示）。因为，默认情况下 SSIS 在运行包之前会验证包的每个数据源，而认证之所以失败是因为 EmpHireDates3 并不存在，即使这个包创建了这个表。

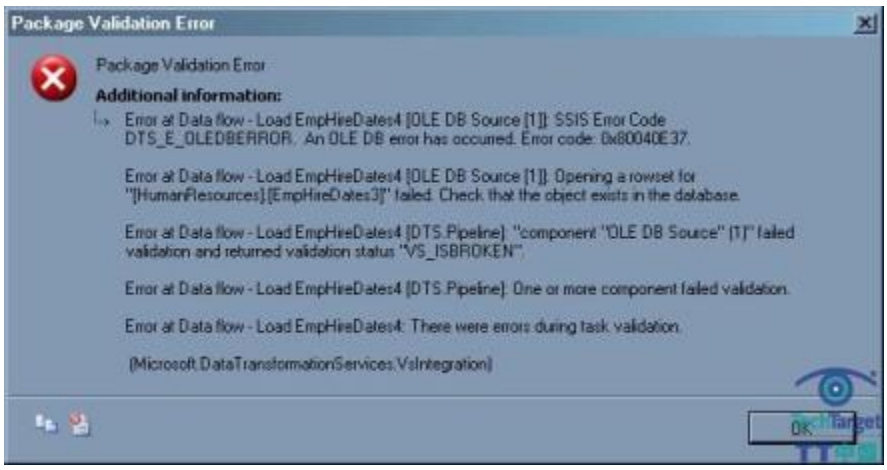


图 3：检索打开 SSIS 数据包时的认证错误消息

DelayValidation 属性

毫无疑问，SSIS 有应对的方法。下面我们看看如何在不验证的情况下使用 DelayValidation 属性打开 SSIS 包。包中的每个可执行文件，以及包本身的执行，都支持 DelayValidation 属性。默认情况下，属性是设置为 False 的。然而，如果我们设置值为 True 的话，那么 SSIS 将运行包并延迟认证该可执行文件直到它运行后。请看回图 1，我们可以看到 Properties 窗口中的 DelayValidation 属性。这个就是包可执行的 DelayValidation 属性。如果我们将它设置为 True，那么包将如预期一样运行。

注意，在包级设置 DelayValidation 属性并不能阻止打开包时的最初的认证过程。它只是在我们运行包时延迟包级的验证。这就意味着包将继续运行，如我在例子中所探讨的，但是当我们打开包时警报仍然出现。如果我们也想阻止打开包时认证的话，理解这一点是很重要的。在这种情况下，我们必须将任务级上的 DelayValidation 属性设置为 True。

避免认证数据源是很有用的一个原因是，有时我们打开 SSIS 包的过程会花费很长的时间。这可能是因为这是一个远程数据源、数据源的响应是很缓慢的或者其它原因。如果

我们正在开发一个包，并且我们需要经常打开这个包，那么我们可能需要花费很长的时间来等待 SSIS 验证一个数据源。在这样的情况下，在任务级设置 DelayValidation 属性为 True 将可以为我们节省大量的时间。

ValidateExternalMetadata 属性

DelayValidation 属性只适用于可执行文件。换言之，我们只可以在任务、容器和包本身设置这个属性。这个属性并不适用于单个数据流组件。然而，数据流组件，如 OLE DB 源组件支持 ValidateExternalMetadata 属性。请看回图 2，我们将看到 Properties 窗口的 OLE DB 源组件属性。

默认情况下，ValidateExternalMetadata 属性是设置为 True 的，因此任何时候我们打开包、添加组件或者运行包，SSIS 都会认证外部元数据。我们也可以通过设置属性为 False 的来重载这个行为，这样的结果是，SSIS 在运行了该组件之后才验证数据源。

ValidateExternalMetadata 属性是一个更细粒度的管理 SSIS 数据源验证过程的方法。然而，这个属性设置为 True 时，它会有助于我们避免包在使用事务出现锁死问题。基于这个原因，在任务级上使用 DelayValidation 属性可能是更好的选择。

这两个属性对于开发 SSIS 包都是很有用的，特别是当验证过程相当缓慢的时候。但是提前验证包同样也是很有价值的，并且还可以节省时间，因此，一定要明智地使用这些属性。

（作者：Robert Sheldon 译者：曾少宁 来源：TT 中国）

SSIS 中使用事件处理程序的五个步骤（上）

有很多关于创建 SQL Server 集成服务（SSIS）包的信息，都关注怎样开发控制流元素和数据流元素。众所周知，你在 SSIS 设计器的“控制流”标签页开发控制流元素，在“数据流”标签页开发数据流元素。SSIS 设计器还提供了“事件处理程序”标签页，它支持你设计基于包可执行体（包括文件，任务等）和它们生成事件的事件处理程序。在本文中，我会给你介绍事件处理程序标签页，并展示怎样给指定的可执行体以及它的事件开发一个事件处理程序。

要为 SSIS 包创建事件处理程序，你必须首先实现该包中要包含的可执行体。可执行体可以是任何控制流任务，或者是容器，或者它可以引用该包本身。你可以对包内的任何可执行体创建事件处理程序。

为了演示下面的事件处理程序例子，我创建了一个简单的 SSIS 包。该包从 AdventureWorks 示例数据库检索数据并把这些数据插入 EmpHireDates 表和 EmpHireDates2 表。我用下面的 Transact-SQL 语句创建这两个表：

```
SELECT c.FirstName, c.MiddleName,
  c.LastName, e.HireDate
  INTO HumanResources.EmpHireDates
  FROM Person.Contact c
  INNER JOIN HumanResources.Employee e
  ON c.ContactID = e.ContactID
  SELECT * INTO HumanResources.EmpHireDates2
  FROM HumanResources.EmpHireDates
```

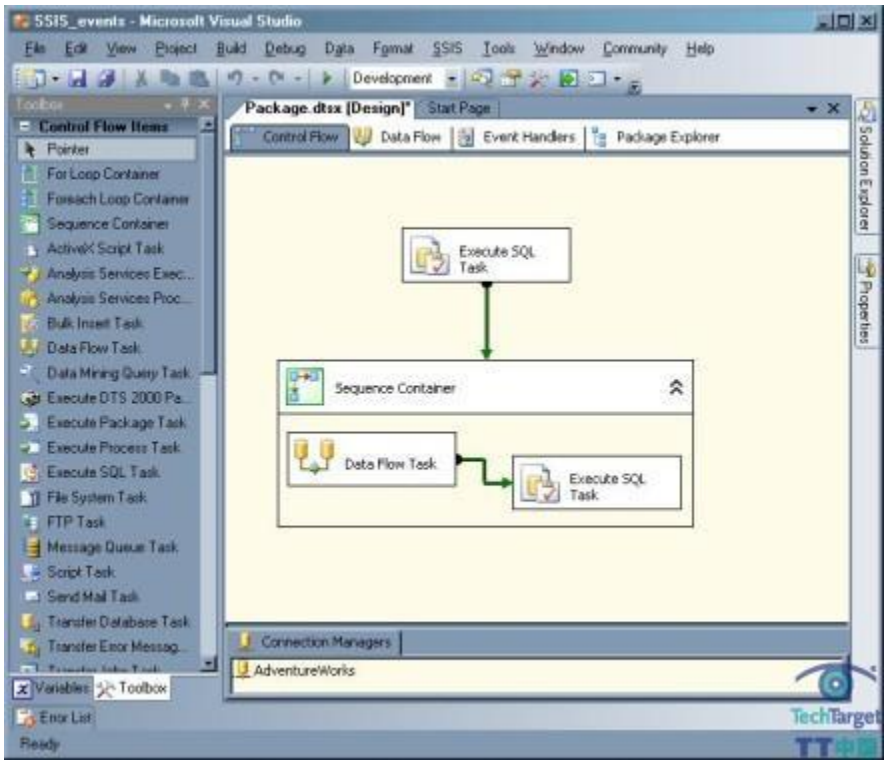


图 1：这个包的控制流标签页。

第一个执行 SQL 的任务清空 EmpHireDates 表和 EmpHireDates2 表。数据流任务从 Employee 表和 Contact 表检索数据，然后把数据导入 EmpHireDates 表。

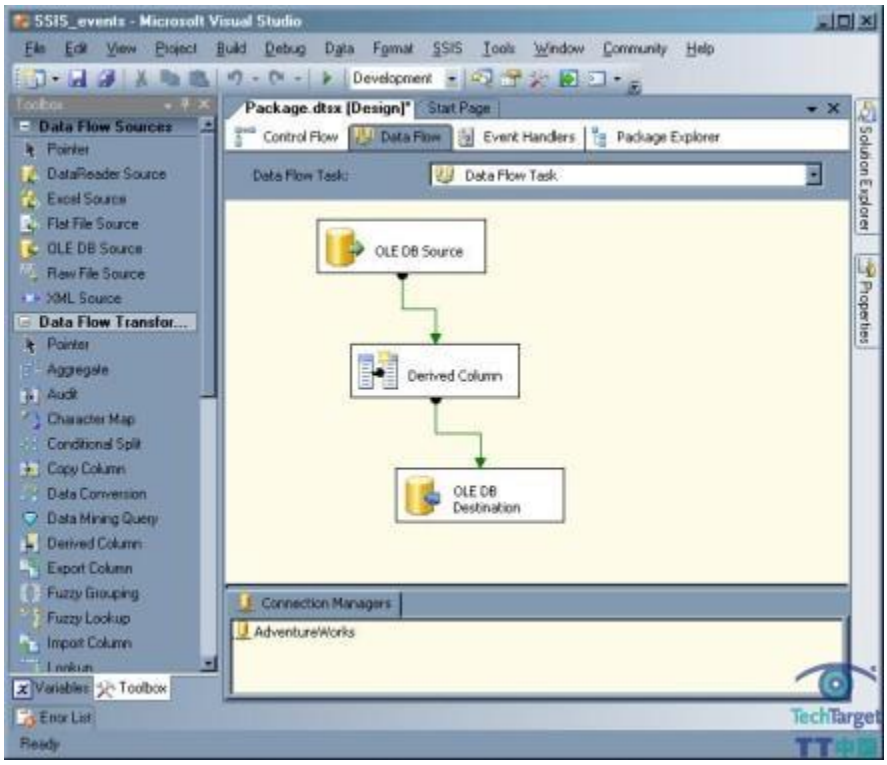


图 2：数据流标签页详细信息。

控制流里第二个执行 SQL 的任务是从 EmpHireDates 表检索数据，然后把数据导入 EmpHireDates2 表。图 1 和图 2 中显示的 SSIS 包是非常简单的。然而，要想说清楚怎样创建事件处理程序，这个包提供的元素还不够。要是你点击事件处理程序标签页，你会看到于图 3 类似的设计界面。在设计界面上方有两个重要的元素：可执行体下拉列表和事件处理程序下拉列表。

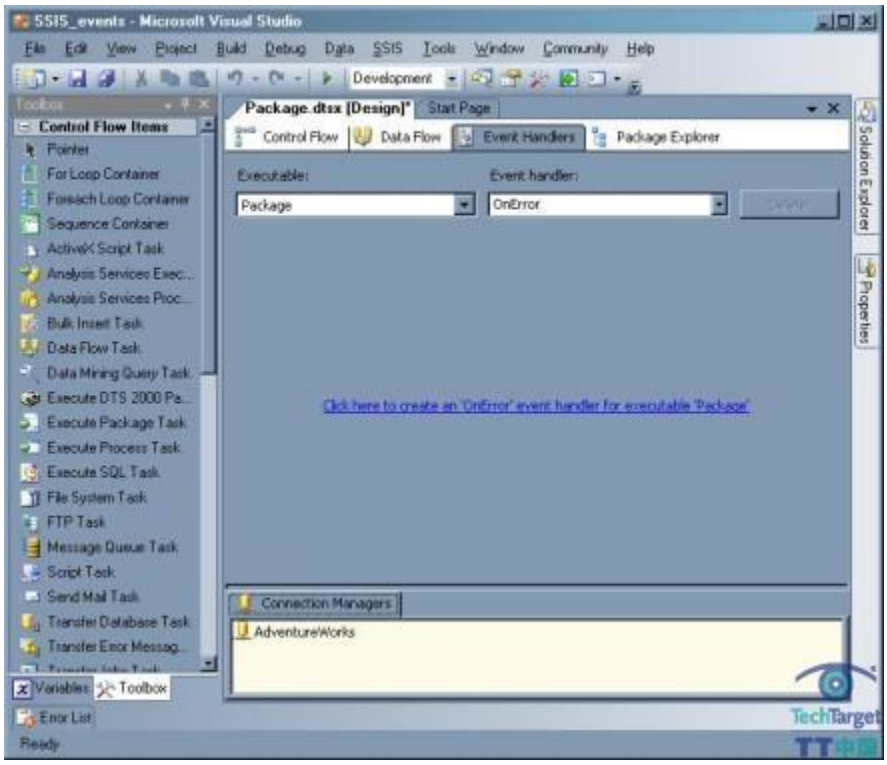


图 3：可执行体的和事件处理程序下拉列表。

第 2 步：选择可执行体

可执行体下拉列表会显示一个当前包中存在的所有可执行体的分层级树形视图。在图 4 中，你会看到包执行体在列表的最上方。在包执行体下面是可执行 SQL 任务和可执行序列容器。因为控制流中的序列容器包含一个数据流任务和一个执行 SQL 的任务。你可以在可执行序列容器下面的“可执行体（Executables）”文件夹中找到这两个执行体。

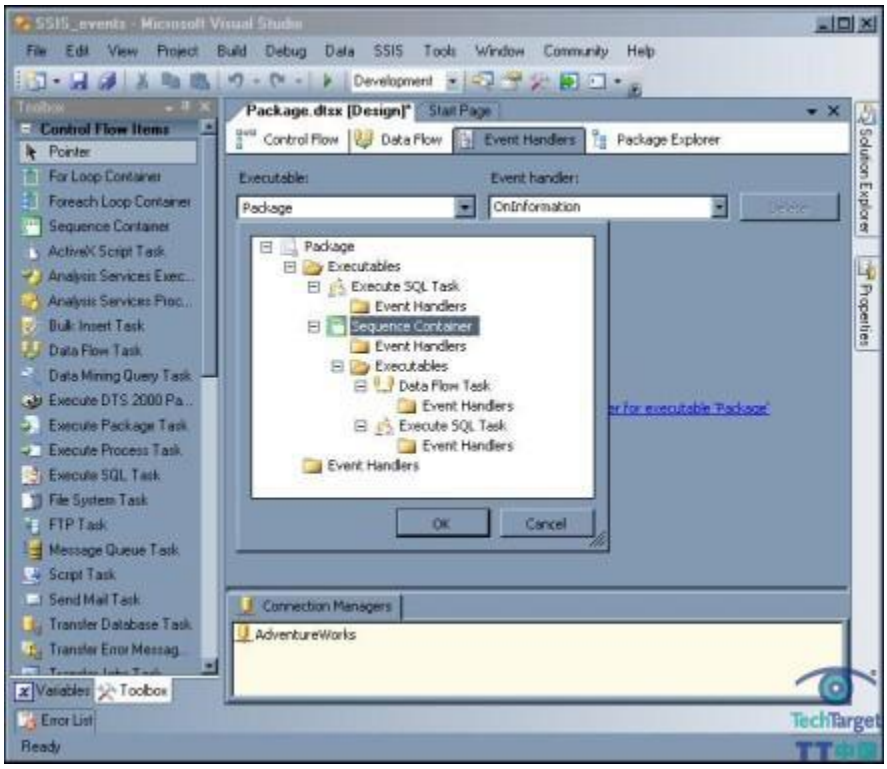


图 4：执行 SQL 任务和序列容器。

在每个可执行体下面，你也会找到一个事件处理程序文件夹。不管你在什么时候给可执行体定义事件处理程序，该事件处理程序都会出现在对应的事件处理程序文件夹下。要想给可执行体定义事件处理程序，先要在下拉列表里选择可执行体，然后点击确定（OK）。在本文中，我给序列容器执行体创建了一个事件处理程序。

第 3 步：选择事件

你可以给每一个可执行体创建一个或多个事件处理程序。每一个事件处理程序都必须在事件处理程序下拉列表中的事件列表中选择，如图 5。例如，如果你选择序列容器执行体，你可以创建一个基于 OnError 事件，OnInformation 事件，OnPostExecute 事件或者下拉列表中的其他事件的事件处理程序。大部分事件的名称是自解释的，如果需要的话，你可以在 Microsoft SQL Server 2005 联机指南中找到每一种事件类型的详细说明。

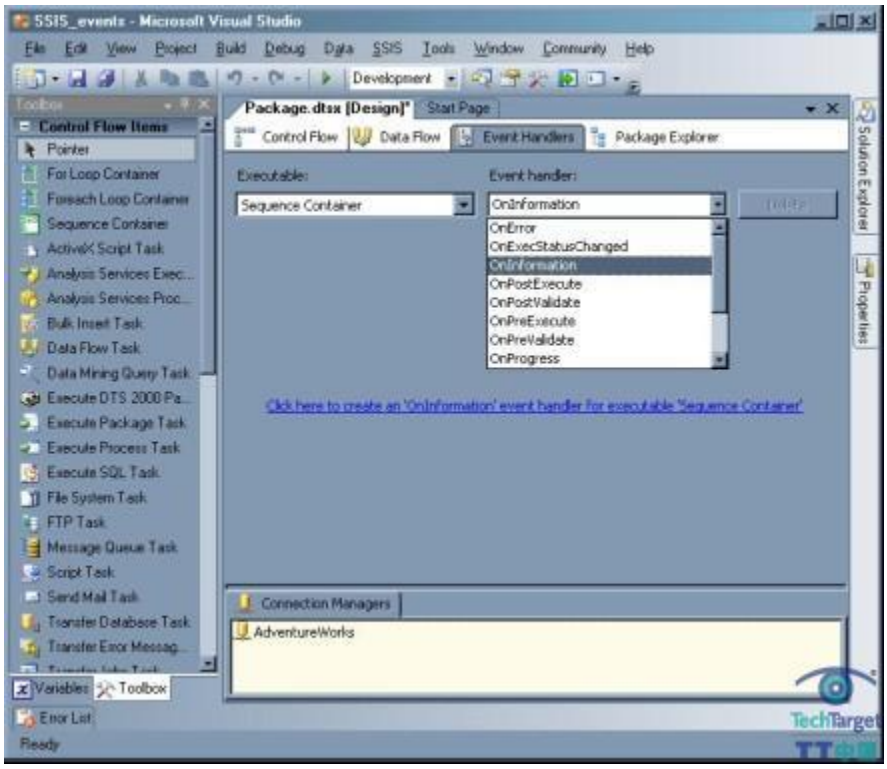


图 5：事件处理程序下拉列表。

当你运行包，然后运行执行体时，它会产生一个或多个预定义事件。它产生哪个事件取决于该操作的成功失败。例如：如果序列容器在运行时遇到错误，它会触发 OnError 事件;或者在执行体开始运行之前，它会触发 OnPreExecute 事件。有时事件由容器内部的任务触发。例如：数据流任务能产生 OnInformation 事件。

(作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国)

SSIS 中使用事件处理程序的五个步骤（下）

在本文中，我选择 OnInformation 事件为例。也就是说，我给序列容器执行体创建了一个 OnInformation 事件处理程序。这样一来，每当容器内任何任务或容器本身产生 OnInformation 事件时， OnInformation 事件处理程序就会执行其已定义的任何行为。

第 4 步：开发工作流

选择可执行体和事件只是你进行创建事件处理程序的第一步。事件处理程序要有实际意义，必须执行一些实际行为。

定义工作流行为的方法跟定义控制流的方法一样。如果你回到图 3，你可以看到事件处理程序设计界面有一个链接。一旦你选择了执行体的和事件，你应该点击该链接激活设计界面，然后创建初始事件处理程序。这个活动是专门针对你正创建的事件处理程序的执行体或者事件对的。假如你选择一个不同的执行体或者事件对，设计界面会再一次出现图形 3 那样的界面。

在你激活设计界面之后，就要添加控制流元素到该界面，如图 6 所示。在这种情况下，我增加了一个执行 SQL 的任务。这样一来，每当序列容器或它的任务之一产生 OnInformation 事件时，执行 SQL 的任务便会运行。你可以同样容易地增加其他的任务，比如发出邮件任务或消息队列任务。就像控制流一样，你可以使用优先限制来连接工作流任务并把任务封入容器，而且你可以在你的容器和任务中使用系统变量和用户定义变量。

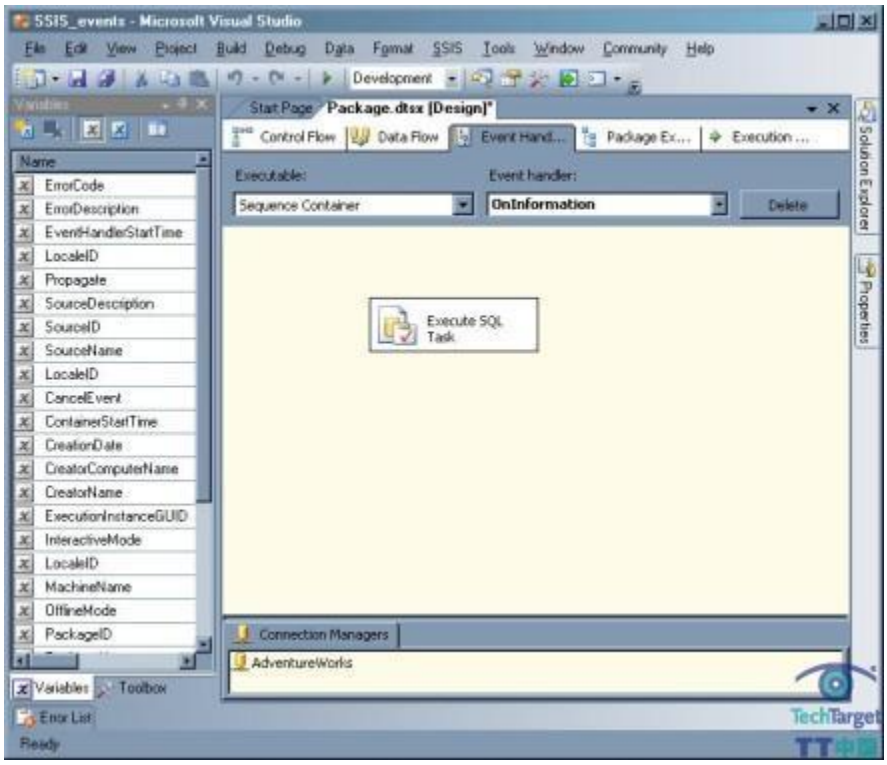


图 6：给设计界面添加控制流元素。

第 5 步：使用变量

图 6 中的变量窗口显示了使用变量的方法，这些变量的作用域对已选择的执行体是可用的。在这里，我用了几个系统变量把 OnInformation 事件日志记录到 SQL Server 中的 SSIS_events 表。我使用下面的 Transact-SQL 来创建这个表。

```
CREATE TABLE dbo.SSIS_events
 (
EIGUID UNIQUEIDENTIFIER NOT NULL,
PackageName NVARCHAR（50） NOT NULL,
SourceName NVARCHAR（50） NOT NULL,
EventInfo NVARCHAR（200） NOT NULL,
TimeLogged DATETIME NOT NULL DEFAULT（getdate（） ）
)
```

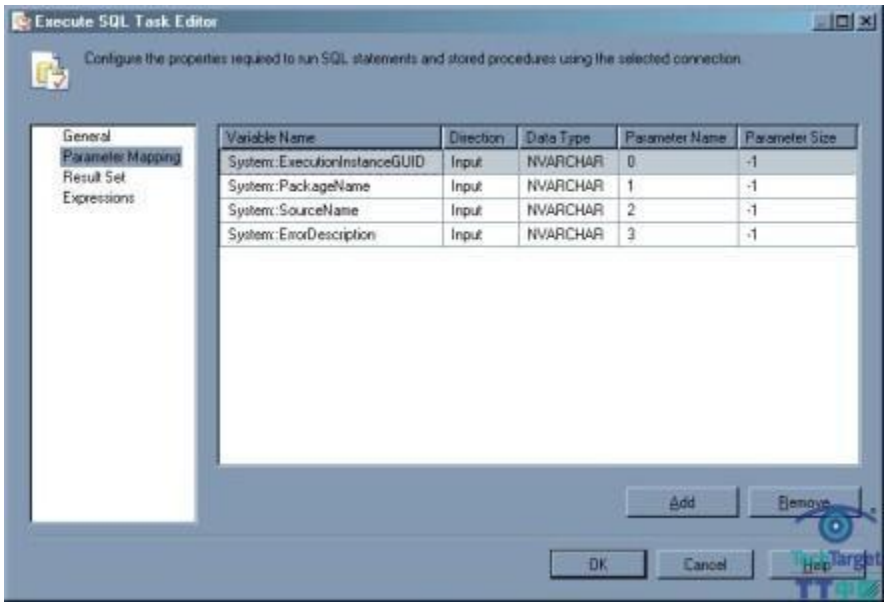


图 7：执行 SQL 任务编辑器的参数映射页

在执行 SQL 任务中使用变量时，你必须给任务里 SQL 语句中的参数占位符映射变量。在本例中，我映射了四个系统变量给参数。图 7 显示了执行 SQL 任务编辑器的参数映射页。每一个映射都被定义为输入，参数名只是整数，很简单，从零开始。

定义好参数映射之后，你就可以定义 SQL 语句。我在执行 SQL 任务中使用了下面的语句：

```
 INSERT INTO dbo.SSIS_events
(EIGUID, PackageName, SourceName, EventInfo)
 VALUES  (?,?,?,?)
```

注意问号当做参数占位符的用法。问号的顺序与参数映射页定义的参数名相对应。例如：第二个变量是 PackageName 。该变量的参数名是 1，实际上是第二个参数。这就表示第二个问号与这个变量对应，这样包名称会作为第二个值插入表中。如果你运行这个包，SSIS 将在下面的表中记录 OnInformation 事件的日志：

EIGUID	PackageName	SourceName	EventInfo	TimeLogged
7359F704-8267-4E6B-8785-1FA2ACB0A1E4	Package	Data Flow Task	Validation phase is beginning.	2007-12-26 16:53:37.700
7359F704-8267-	Package	Data Flow	Prepare for Execute	2007-12-26

4E6B-8785-1FA2ACB0A1E4		Task	phase is beginning.	16:53:38.030
7359F704-8267-4E6B-8785-1FA2ACB0A1E4	Package	Data Flow Task	Pre-Execute phase is beginning.	2007-12-26 16:53:40.907
7359F704-8267-4E6B-8785-1FA2ACB0A1E4	Package	Data Flow Task	Execute phase is beginning.	2007-12-26 16:53:42.623
7359F704-8267-4E6B-8785-1FA2ACB0A1E4	Package	Data Flow Task	The final commit for the data insertion has started.	2007-12-26 16:53:43.047
7359F704-8267-4E6B-8785-1FA2ACB0A1E4	Package	Data Flow Task	The final commit for the data insertion has ended.	2007-12-26 16:53:43.110
7359F704-8267-4E6B-8785-1FA2ACB0A1E4	Package	Data Flow Task	Post Execute phase is beginning.	2007-12-26 16:53:43.200
7359F704-8267-4E6B-8785-1FA2ACB0A1E4	Package	Data Flow Task	Cleanup phase is beginning.	2007-12-26 16:53:43.297
7359F704-8267-4E6B-8785-1FA2ACB0A1E4	Package	Data Flow Task	"component "OLE DB Destination" (90)" wrote 290 rows.	2007-12-26 16:53:43.373

数据流任务会产生 OnInformation 事件。序列容器本身在容器内不执行 SQL 任务。当然，这只是一个例子——你可以记录任何想记录的信息，你也可以执行其他任何类型的行为。另外，你可以给其他可执行体创建事件处理程序。例如，你可能想让包执行失败时发送一份电子邮件，或者你可能只想对数据流任务记录错误事件，而不管其他任务。或你可以给数据流任务创建多个事件处理程序，而不管其他执行体。

这里要指出的是： SQL Server 集成服务（SSIS）中的事件处理程序标签页非常的灵活，它支持根据你想监视的事件和你想执行的动作执行各种行为。事件处理程序标签页与控制流标签页，或者数据流标签页一样灵活，一样具有可扩展性——你只是需要知道怎样使用它。

（作者：Robert Sheldon 译者：冯昀晖 来源：TT 中国）

用 SSIS 2005 向导处理渐变维度(上)

例如：在图一（渐变维列）中，屏幕截图显示 AccountDescription 列被设置为变化型属性，AccountType 列是历史型属性，Operator 是不变属性。

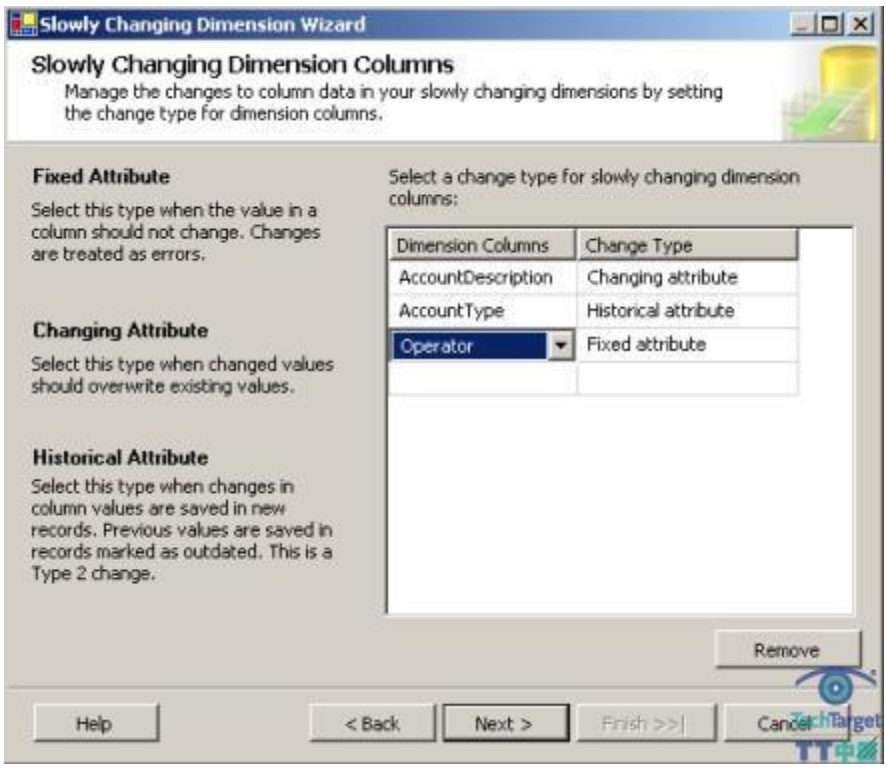


图 1，变化类型：不变属性，变化型属性，历史型属性。

在图 2（历史型选项）中，你可以配置你的实现细节。你可以让 SSIS 包在检测到不变属性变化时执行失败；而且你可以在检测到变化型记录被修改时，修改所有匹配的记录，包括过期记录。大家可以看看我以前对第一个选项的讨论。

第二个选项对那些会有重复值，重复跨多个维度成员的属性比较有用。例如：同一个账户类型被应用到多个账户，如果账户类型从“负债”变成了“流动负债”，你可能想对所有此类型账户应用这一变更。

接下来，向导会提示你选择识别当前记录和过期记录的方式。有两个选项供你选择：把记录标识为过期（或废弃，超期或者你选择的其他形容词）；或者通过给该条记录添加

过期日期来表示本记录是过期记录。如果你选择了后者，你可以使用 SSIS 的一个全局变量来判断更新过期维记录的日期和时间值。我更喜欢使用标志当前记录的方式，所有其他记录可以认为是过期的。当然，你可以从多个选项中选择。

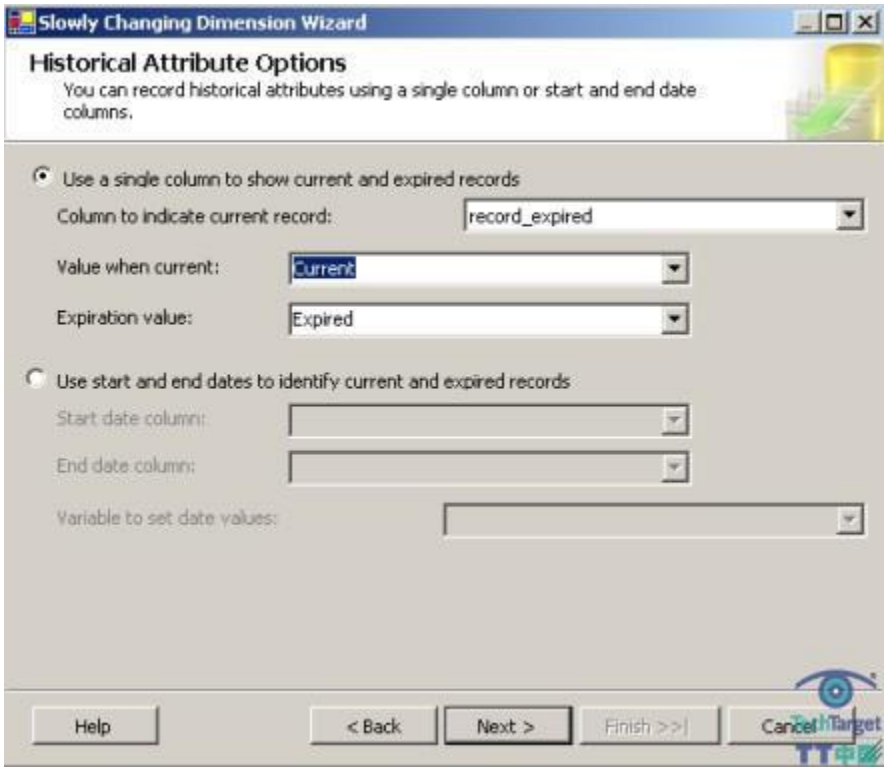


图 2，配置你的实现细节。

图 3 所示界面（数据流任务）允许你配置对自动推断维成员的支持。自动推断成员在你加载一条记录给实际表时就创建了，它没有相应的维记录。为了分析正确，每一个实际表记录必须与维度表中的一条记录相关联。

在数据仓库中有多种途径做到这一点。SCD 向导允许你创建一个新的自动推断成员记录，其中所有维属性设置为 Null。在你的维度表里设置一个“未知”成员（可能是让代理键等于-999，或者另一个特殊的不在正常范围内的值）常常很有用，省得创建自动推断成员。在本例中我们只关注 Type1 类型的 SCD 和 Type2 类型的 SCD，并且不关注自动推断成员。所以我们取消默认选中，然后点击下一步。

这就是向导为找出维护 SCD 要求的整个数据流需要的所有信息。你可以从下面的屏幕截图中看到，向导替你做了大量的工作。

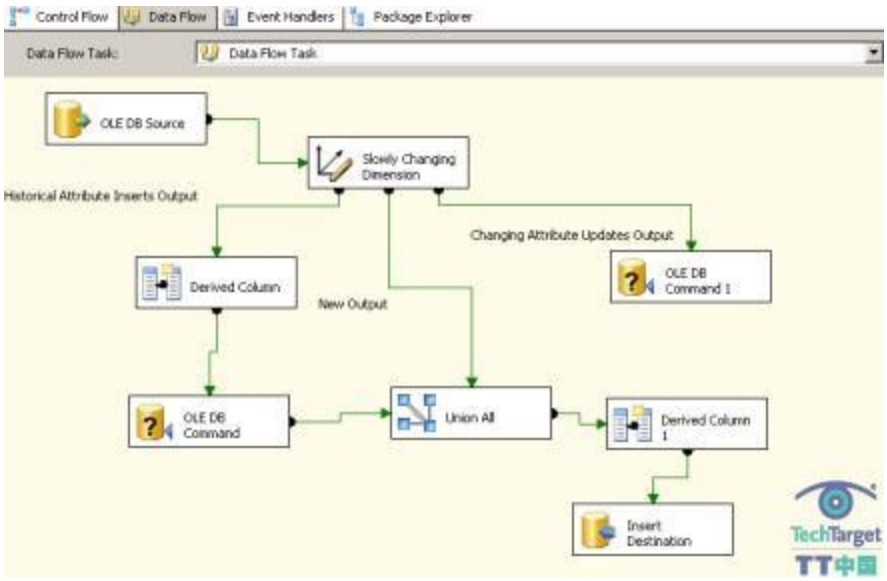


图 3，使用数据流任务配置对自动推断维成员的支持。

你可以定制向导生成的数据流来满足应用程序的需要。但是，首先我们要检查最新创建的转换组件：如果 AccountType 得值被改变了，OLEDB 命令转换器就更新“record_expired（过期记录）”属性。如果数据源的值变了，SSIS 会创建新纪录，这样 Union All 转换组件会把这条记录和原来存在的数据源记录联合起来。OLEDB 命令 1 转换组件会用类似下面的 SQL 命令来更新 AccountDescription 的值——基于业务键 AccountCodeAlternateKey 的变化属性。

```
UPDATE [dbo].[DimAccount] SET [AccountDescription] = ?

WHERE [AccountCodeAlternateKey] = ?
```

最后，插入目标转换组件会填入目标表 DimAccount。

AccountCodeAlternateKey	AccountDescription	AccountType	Operator	record_expired
1120	Receivables	Wonderful Assets	+	Current
1120	Receivables	Assets	+	Null

现在，我们在事务数据库中更新几个成员，然后运行包看看 SCD 向导维护变化维度是否正确。事实上，我们可以确认在把属性值 AccountType 得值从“assets”改为“wonderful assets”时，触发了在 Account 维创建新记录，并把新创建的记录标记为当前记录：

AccountCodeAlternateKey	AccountDescription	AccountType	Operator	record_expired
10	Active Asset	Assets	+	Null

另一方面，AccountDescription 列的变化不创建新行——它们只是简单地覆盖原来的值。

就像我在本文一开始就提到的，在你的数据仓库里有一些维护渐变维度（SCD）其他方法不能通过向导使用。然而，通过在 SSIS 中点击几下，就实现 Type1 和 Type2 渐变维度的能力肯定能加快 ETL 开发的速度。

（作者：Michelle Gutzait 译者：孙瑞 来源：TT 中国）

用 SSIS 2005 向导处理渐变维度(下)

例如：在图一（渐变维列）中，屏幕截图显示 AccountDescription 列被设置为变化型属性，AccountType 列是历史型属性，Operator 是不变属性。

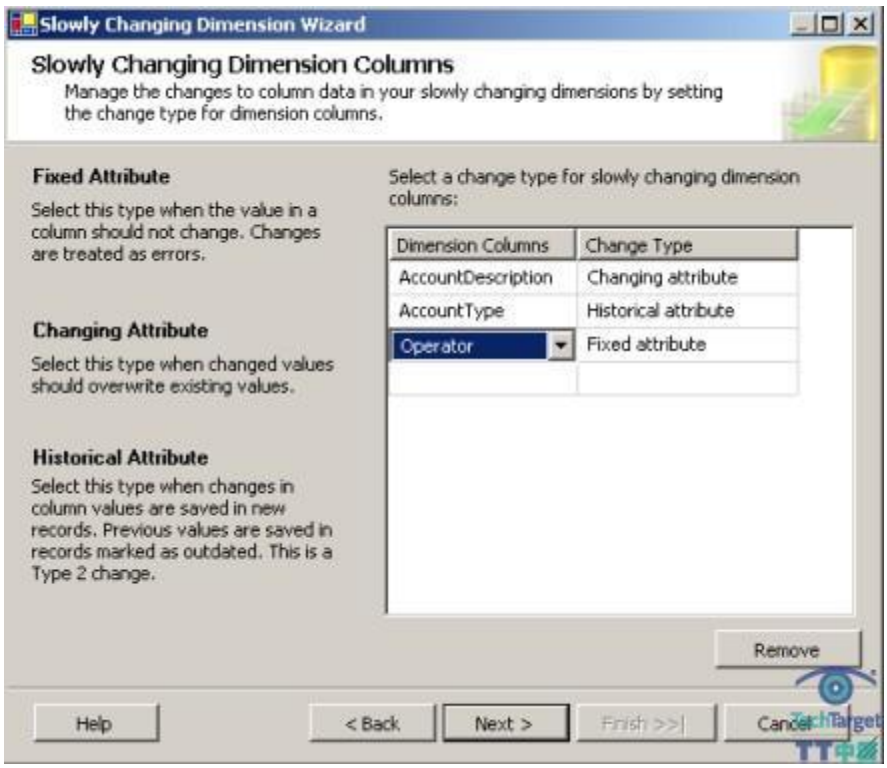


图 1，变化类型：不变属性，变化型属性，历史型属性。

在图 2（历史型选项）中，你可以配置你的实现细节。你可以让 SSIS 包在检测到不变属性变化时执行失败；而且你可以在检测到变化型记录被修改时，修改所有匹配的记录，包括过期记录。大家可以看看我以前对第一个选项的讨论。

第二个选项对那些会有重复值，重复跨多个维度成员的属性比较有用。例如：同一个账户类型被应用到多个账户，如果账户类型从“负债”变成了“流动负债”，你可能想对所有此类型账户应用这一变更。

接下来，向导会提示你选择识别当前记录和过期记录的方式。有两个选项供你选择：把记录标识为过期（或废弃，超期或者你选择的其他形容词）；或者通过给该条记录添加

过期日期来表示本记录是过期记录。如果你选择了后者，你可以使用 SSIS 的一个全局变量来判断更新过期维记录的日期和时间值。我更喜欢使用标志当前记录的方式，所有其他记录可以认为是过期的。当然，你可以从多个选项中选择。

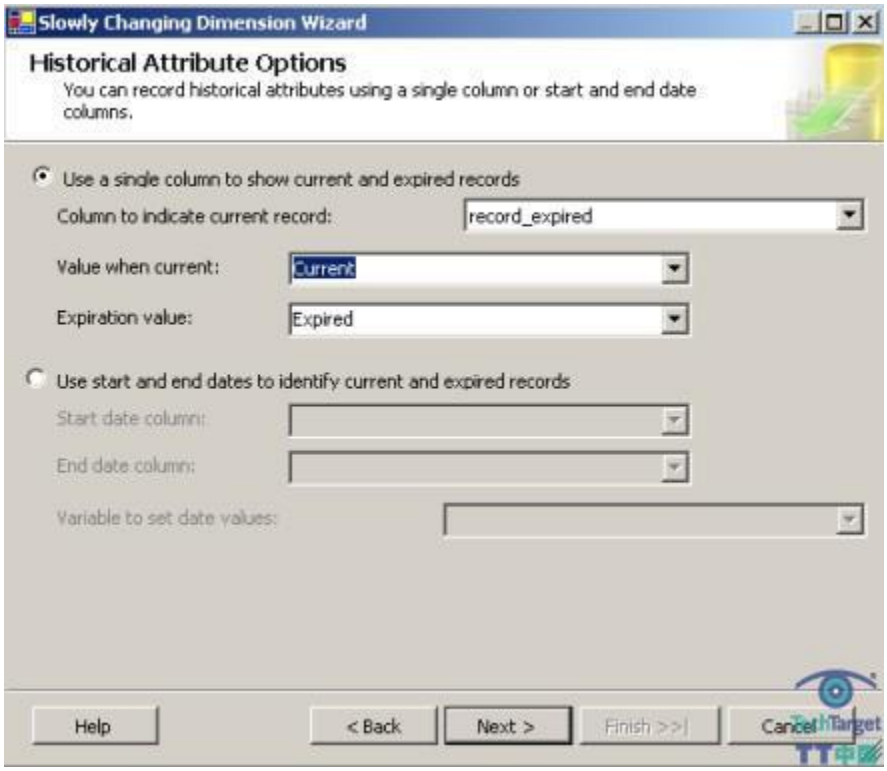


图 2，配置你的实现细节。

图 3 所示界面（数据流任务）允许你配置对自动推断维成员的支持。自动推断成员在你加载一条记录给实际表时就创建了，它没有相应的维记录。为了分析正确，每一个实际表记录必须与维度表中的一条记录相关联。

在数据仓库中有多种途径做到这一点。SCD 向导允许你创建一个新的自动推断成员记录，其中所有维属性设置为 Null。在你的维度表里设置一个“未知”成员（可能是让代理键等于-999，或者另一个特殊的不在正常范围内的值）常常很有用，省得创建自动推断成员。在本例中我们只关注 Type1 类型的 SCD 和 Type2 类型的 SCD，并且不关注自动推断成员。所以我们取消默认选中，然后点击下一步。

这就是向导为找出维护 SCD 要求的整个数据流需要的所有信息。你可以从下面的屏幕截图中看到，向导替你做了大量的工作。

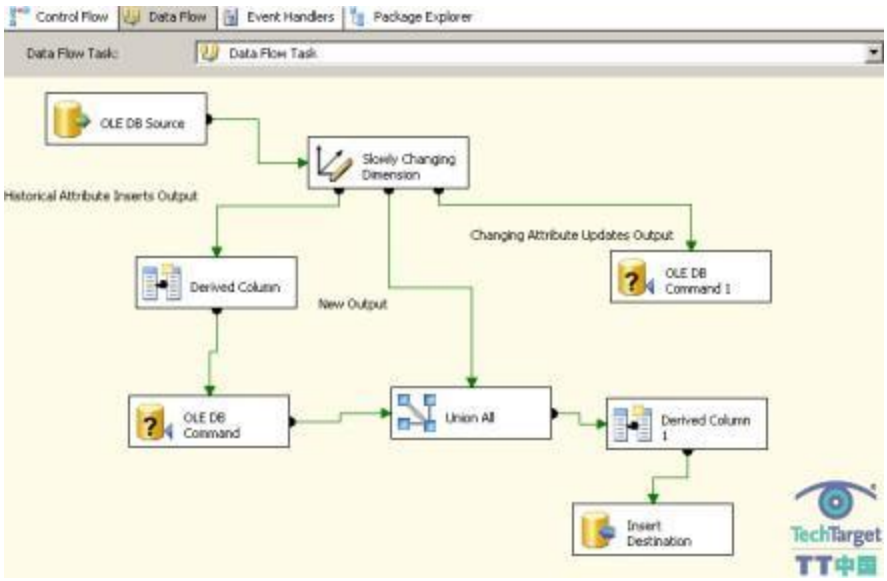


图 3，使用数据流任务配置对自动推断维成员的支持。

你可以定制向导生成的数据流来满足应用程序的需要。但是，首先我们要检查最新创建的转换组件：如果 AccountType 得值被改变了，OLEDB 命令转换器就更新“record_expired（过期记录）”属性。如果数据源的值变了，SSIS 会创建新纪录，这样 Union All 转换组件会把这条记录和原来存在的数据源记录联合起来。OLEDB 命令 1 转换组件会用类似下面的 SQL 命令来更新 AccountDescription 的值——基于业务键 AccountCodeAlternateKey 的变化属性。

```
UPDATE [dbo].[DimAccount] SET [AccountDescription] = ?

WHERE [AccountCodeAlternateKey] = ?
```

最后，插入目标转换组件会填入目标表 DimAccount。

AccountCodeAlternateKey	AccountDescription	AccountType	Operator	record_expired
1120	Receivables	Wonderful Assets	+	Current
1120	Receivables	Assets	+	Null

现在，我们在事务数据库中更新几个成员，然后运行包看看 SCD 向导维护变化维度是否正确。事实上，我们可以确认在把属性值 AccountType 得值从“assets”改为“wonderful assets”时，触发了在 Account 维创建新记录，并把新创建的记录标记为当前记录：

AccountCodeAlternateKey	AccountDescription	AccountType	Operator	record_expired
10	Active Asset	Assets	+	Null

另一方面，AccountDescription 列的变化不创建新行——它们只是简单地覆盖原来的值。

就像我在本文一开始就提到的，在你的数据仓库里有一些维护渐变维度（SCD）其他方法不能通过向导使用。然而，通过在 SSIS 中点击几下，就实现 Type1 和 Type2 渐变维度的能力肯定能加快 ETL 开发的速度。

（作者：Michelle Gutzait 译者：孙瑞 来源：TT 中国）