# Looping through a collection of SQL tables using the SSIS Foreach Loop Container

## Introduction

A lady named Barbara read my SSIS Foreach Loop Container doc and asked how to use the same container to perform ETL operations on each table in a database instead of a list of files. The answer, or this version of it, seemed like a worthwhile topic to address to a wider audience.

This isn't the only way to tackle this problem, of course. In fact, it's hardly worthwhile using SSIS for this within a single database – I'd be inclined to do this in TSQL - but the technique could be valuable if you wanted to loop through objects in multiple databases.

Anyway, the trick in this instance is to use the Foreach ADO.NET Schema Rowset Enumerator in the Foreach Loop Container. Let's set up a simple test to copy data from a list of tables to another table; first we'll need a sandbox to play in. I'm assuming that you're familiar with SSIS basics, but take a look at some of my other examples if this isn't clear.

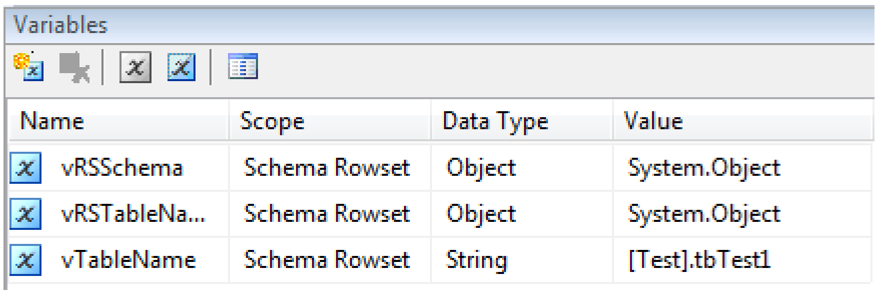
## Test database setup

Run the following script in the Management Studio to create a new database and some tables within it.

```
use master
go

--      Create new database (default opts.) and new tables

create database testdb
go

use testdb
go


CREATE SCHEMA [Test] AUTHORIZATION [dbo]
GO

create table [Test].tbTest1
        (ID     int not null primary key identity(1, 1),
        ValueField int not null);


create table [Test].tbTest2
        (ID     int not null primary key identity(1, 1),
        ValueField int not null);
```

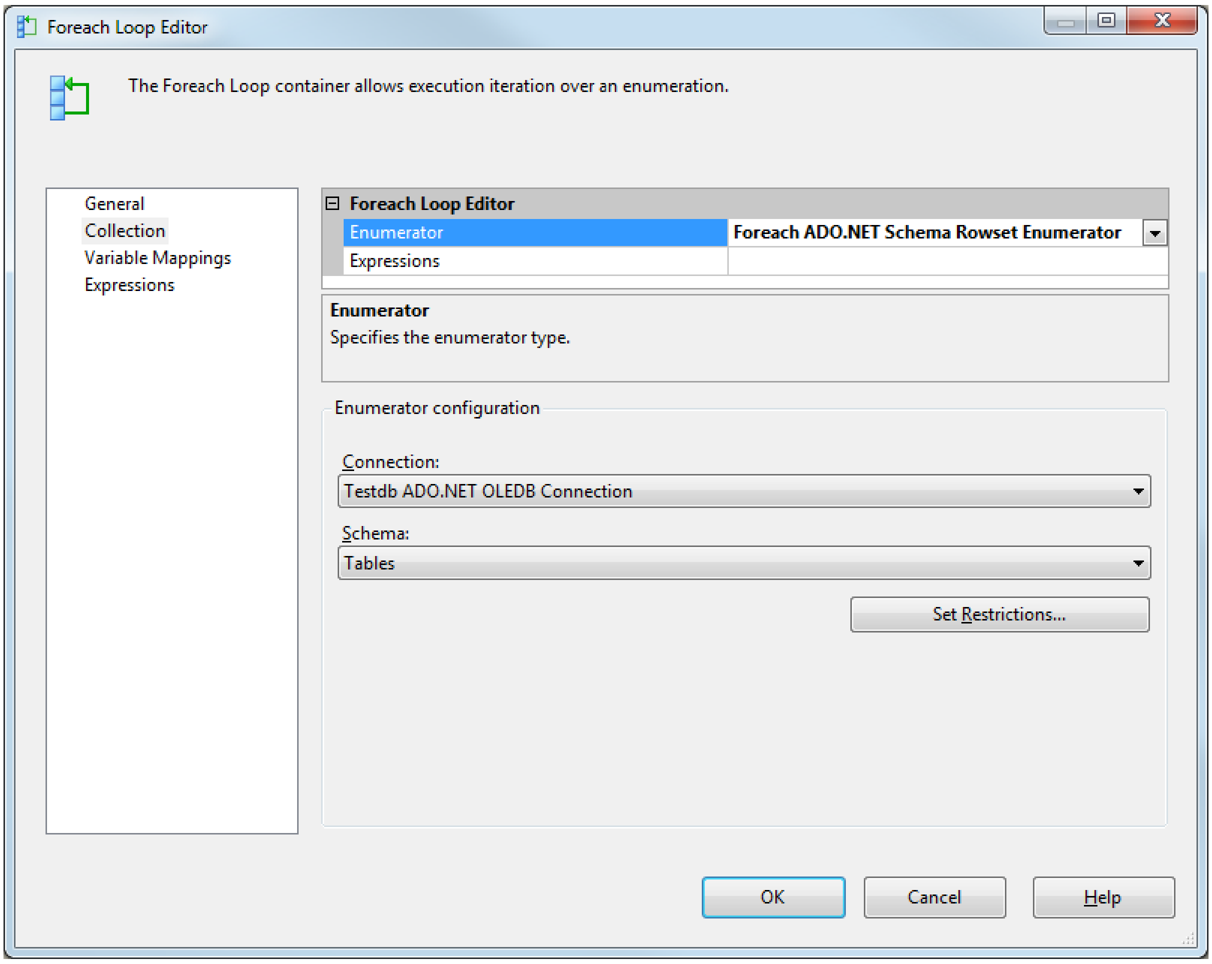
```
create table tbTestDestination
       (ID    int not null primary key identity(1, 1),
       TableName           nvarchar(128)   null,
       TableValue          nvarchar(128) null);


--     Populate tables with test data

insert into [Test].tbTest1 (ValueField)
       select [OBJECT_ID]
       from   master.sys.objects;

insert into [Test].tbTest2 (ValueField)
       select (ValueField + 123)
       from   [Test].tbTest1
       where  ID <   50;
```

Next, create a new Integration Services package named “Schema Rowset” in a new or existing project/solution. Add three variables as follows:



| Name | Scope | Data Type | Value |
|---|---|---|---|
| vRSSchema | Schema Rowset | Object | System.Object |
| vRSTableNa... | Schema Rowset | Object | System.Object |
| vTableName | Schema Rowset | String | [Test].tbTest1 |

Drag a Foreach Loop Container on to the Control Flow surface. Right-click on the container and choose **Edit…** from the context menu. In the Foreach Loop Editor, select **Collection** on the left-hand side.



The Foreach Loop container allows execution iteration over an enumeration.

Foreach Loop Editor

| | |
|---|---|
| Enumerator | Foreach ADO.NET Schema Rowset Enumerator |
| Expressions | |

**Enumerator**
Specifies the enumerator type.

Enumerator configuration

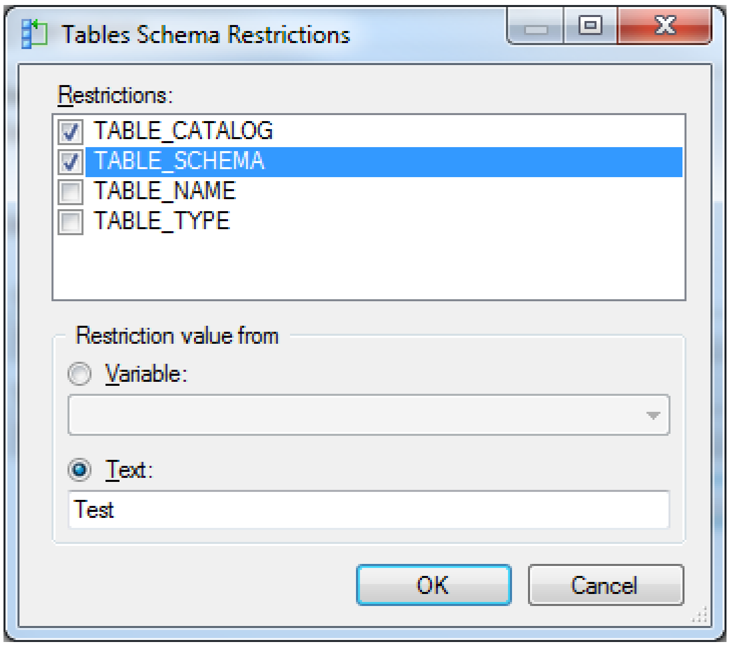
Connection: Testdb ADO.NET OLEDB Connection

Schema: Tables

Select “Foreach ADO.NET Schema Rowset Enumerator” from the Enumerator list, then select **New connection…** from the Connection drop-down. Point the connection manager towards your server and the newly-created **testdb** database.

Select **Tables** from the Schema drop-down. You’ll notice that there’s a long list of objects and metadata that we can select from, but for now (because that was the question) we’ll stick to tables.

Click the **Set Restrictions** button to limit the amount of data the enumerator returns. For this example, I only want to loop through tables in the “Test” schema so I configure ”Test” as the restriction for TABLE_SCHEMA.


Tables Schema Restrictions

Restrictions:

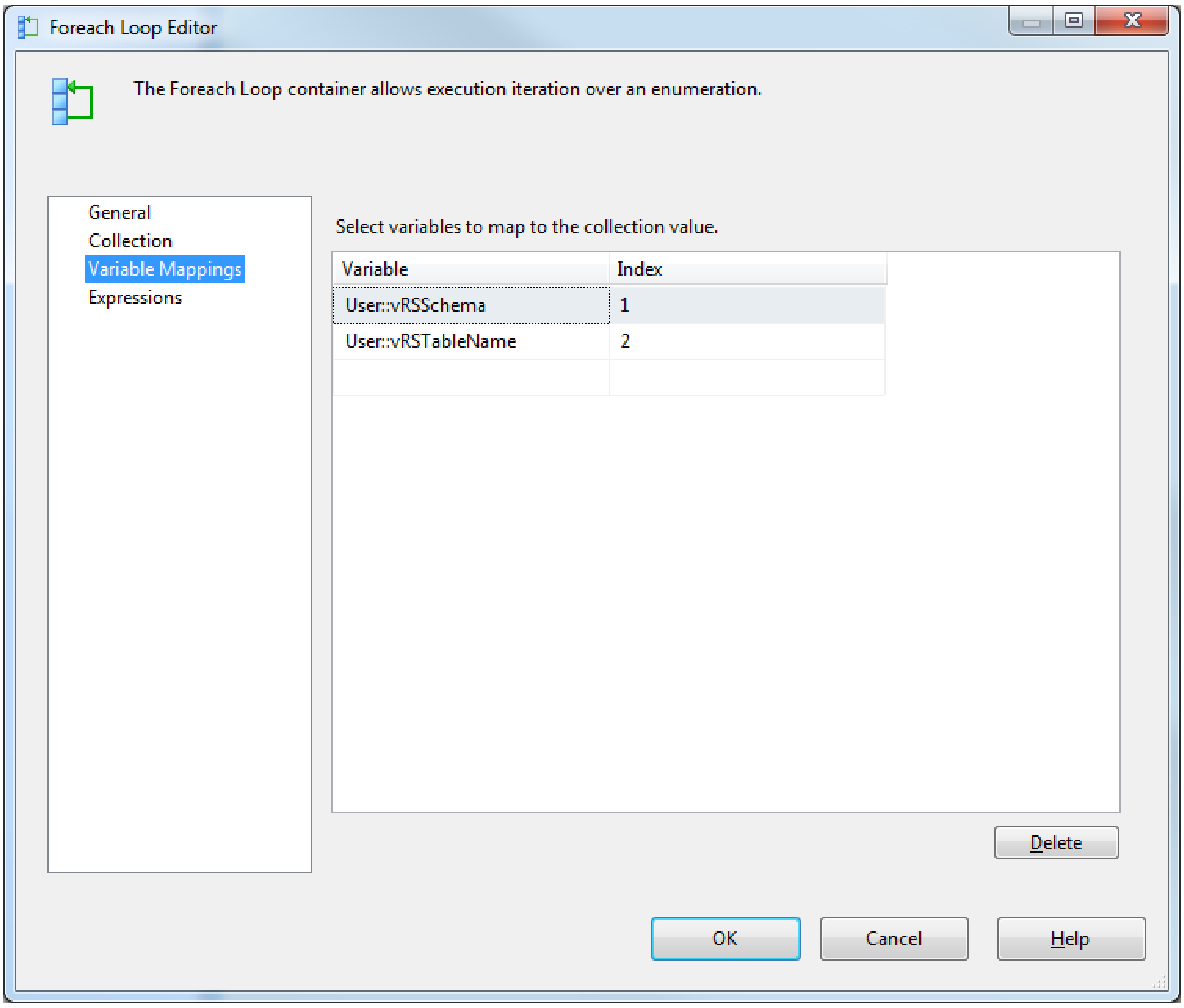
- [✓] TABLE_CATALOG
- [✓] TABLE_SCHEMA
- [ ] TABLE_NAME
- [ ] TABLE_TYPE

Restriction value from

Variable: [ ]

[✓ Text:] Test

Select **Variable Mappings** in the Foreach Loop editor and add mappings for vRSSchema (with Index 1) and vRSTableName (Index 2). (Index 0 will enumerate the database name, should you ever need it).

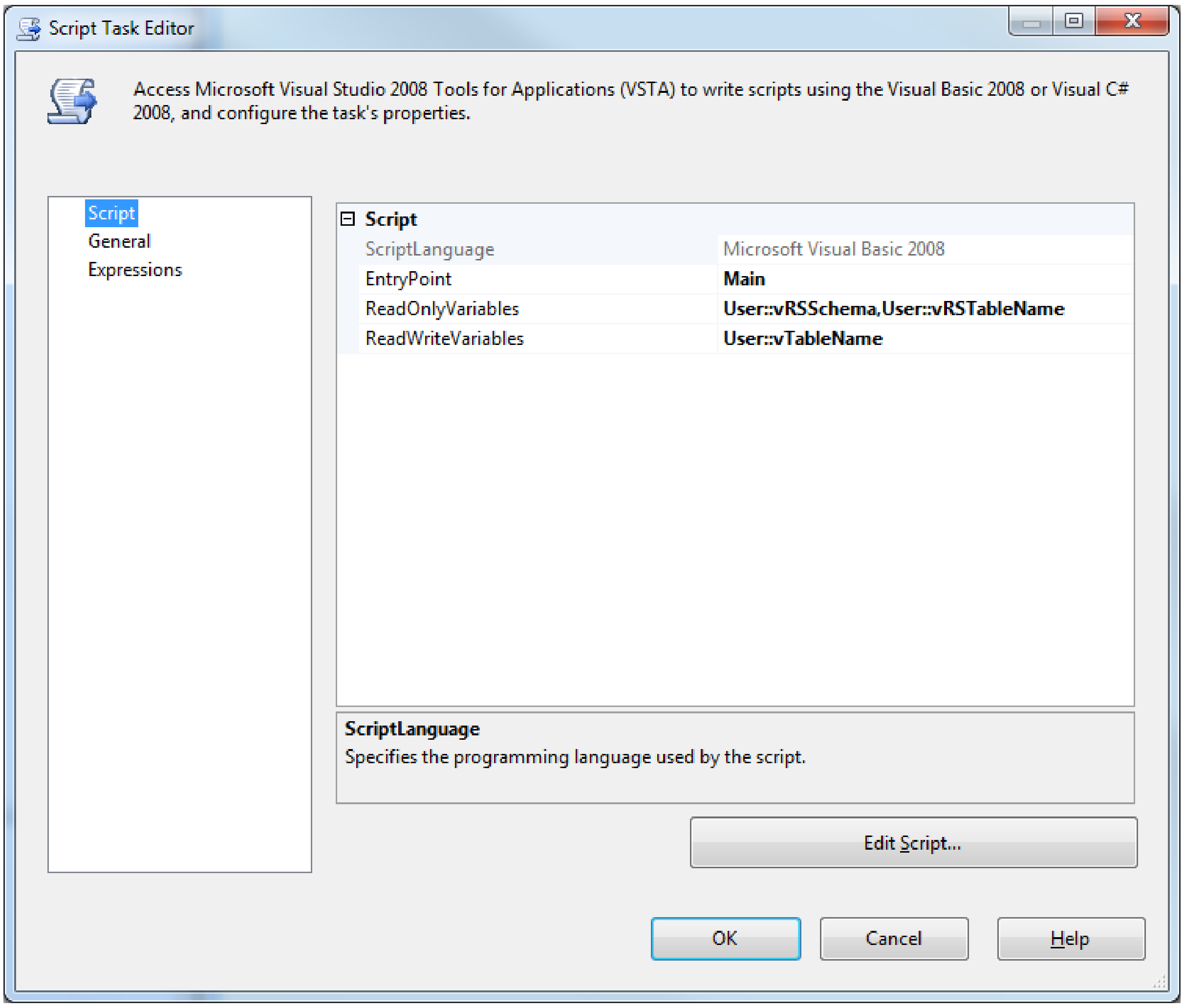


Select variables to map to the collection value.

| Variable | Index |
|---|---|
| User::vRSSchema | 1 |
| User::vRSTableName | 2 |

These are object variables, because only these types of variables are supported for output by this enumerator. The OLEDB Data Flow component requires a string-type variable though, just to make life difficult, so the next step before the tablename provided by the enumerator can be used as a data source is to convert it. To do this, add a script task within the container.

This example uses VB, but C# is also possible since SQL Server 2008 was released. Map the three variables as shown, then click **Edit Script…**



Access Microsoft Visual Studio 2008 Tools for Applications (VSTA) to write scripts using the Visual Basic 2008 or Visual C# 2008, and configure the task's properties.

| Script | |
|---|---|
| ScriptLanguage | Microsoft Visual Basic 2008 |
| EntryPoint | **Main** |
| ReadOnlyVariables | **User::vRSSchema,User::vRSTableName** |
| ReadWriteVariables | **User::vTableName** |

**ScriptLanguage**
Specifies the programming language used by the script.

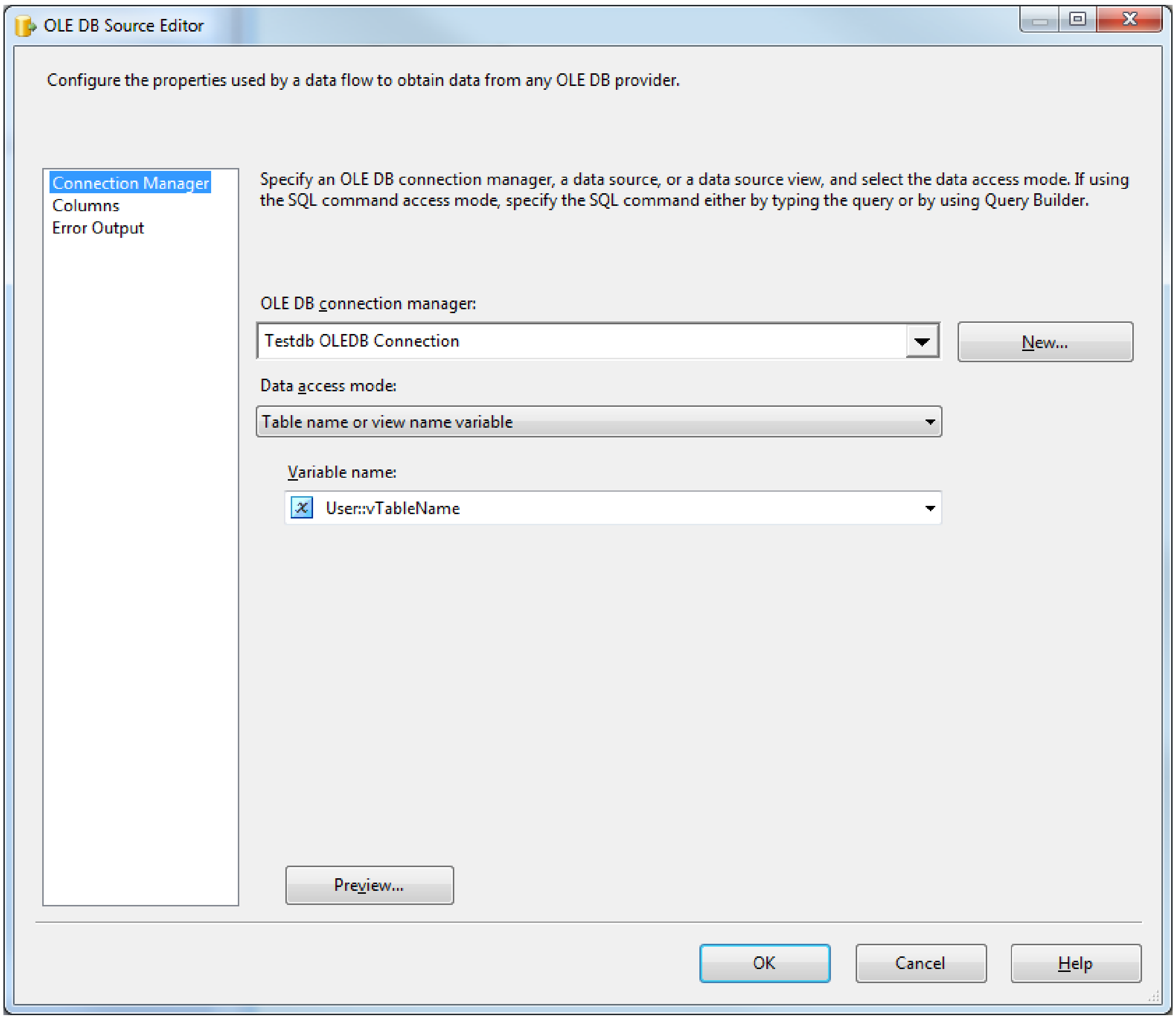
Update the Main() subroutine as follows:

```
Public Sub Main()

 Dim vTableNameStr As String

 vTableNameStr = "[" + Dts.Variables("vRSSchema").Value.ToString + "]."
vTableNameStr = vTableNameStr + Dts.Variables("vRSTableName").Value.ToString


 Dts.Variables("vTableName").Value = vTableNameStr

 Dts.TaskResult = ScriptResults.Success

End Sub
```

…then save and close the script editor. This script will convert the two records and output them as a string in the format **[schema].TableName**

Add a Data Flow task within the Foreach Loop container. Add an OLE DB data source to it.

Create a new OLEDB connection manager for **testdb** by clicking **New…** from the editor, then set the Data access mode to **"**Table name or view name variable" and select **User::vTableName** from the drop-down.


OLE DB Source Editor

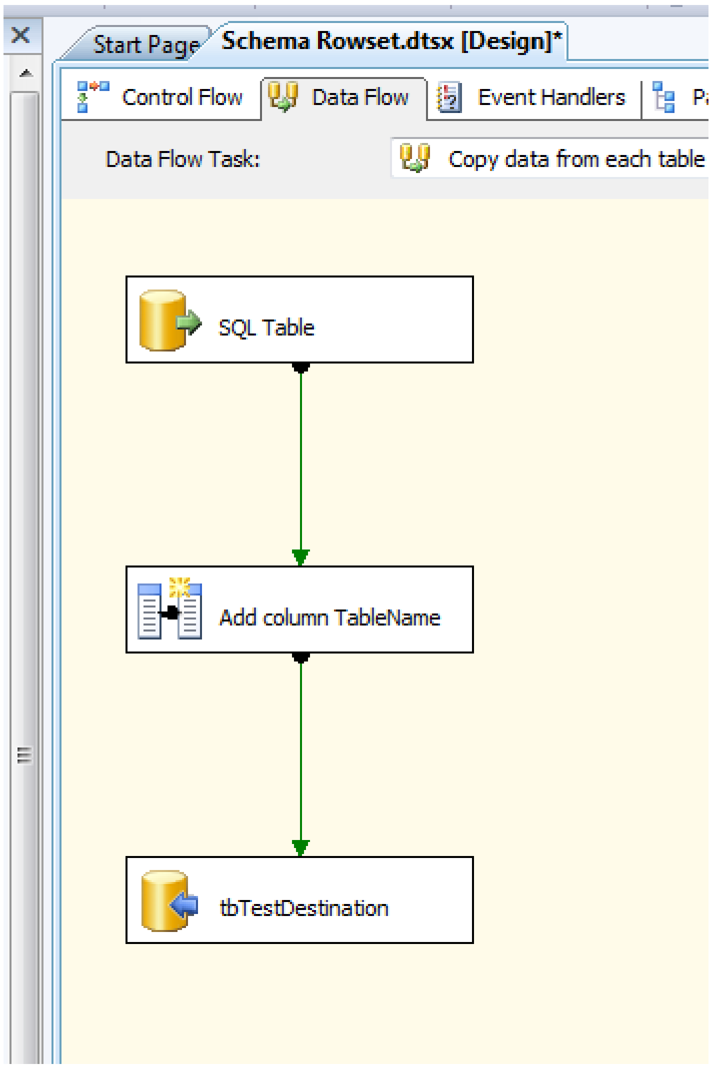
Configure the properties used by a data flow to obtain data from any OLE DB provider.

Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder.

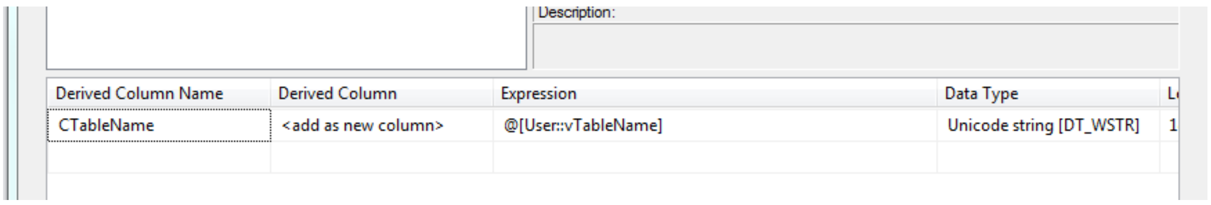
OLE DB connection manager: Testdb OLEDB Connection

Data access mode: Table name or view name variable

Variable name: User::vTableName

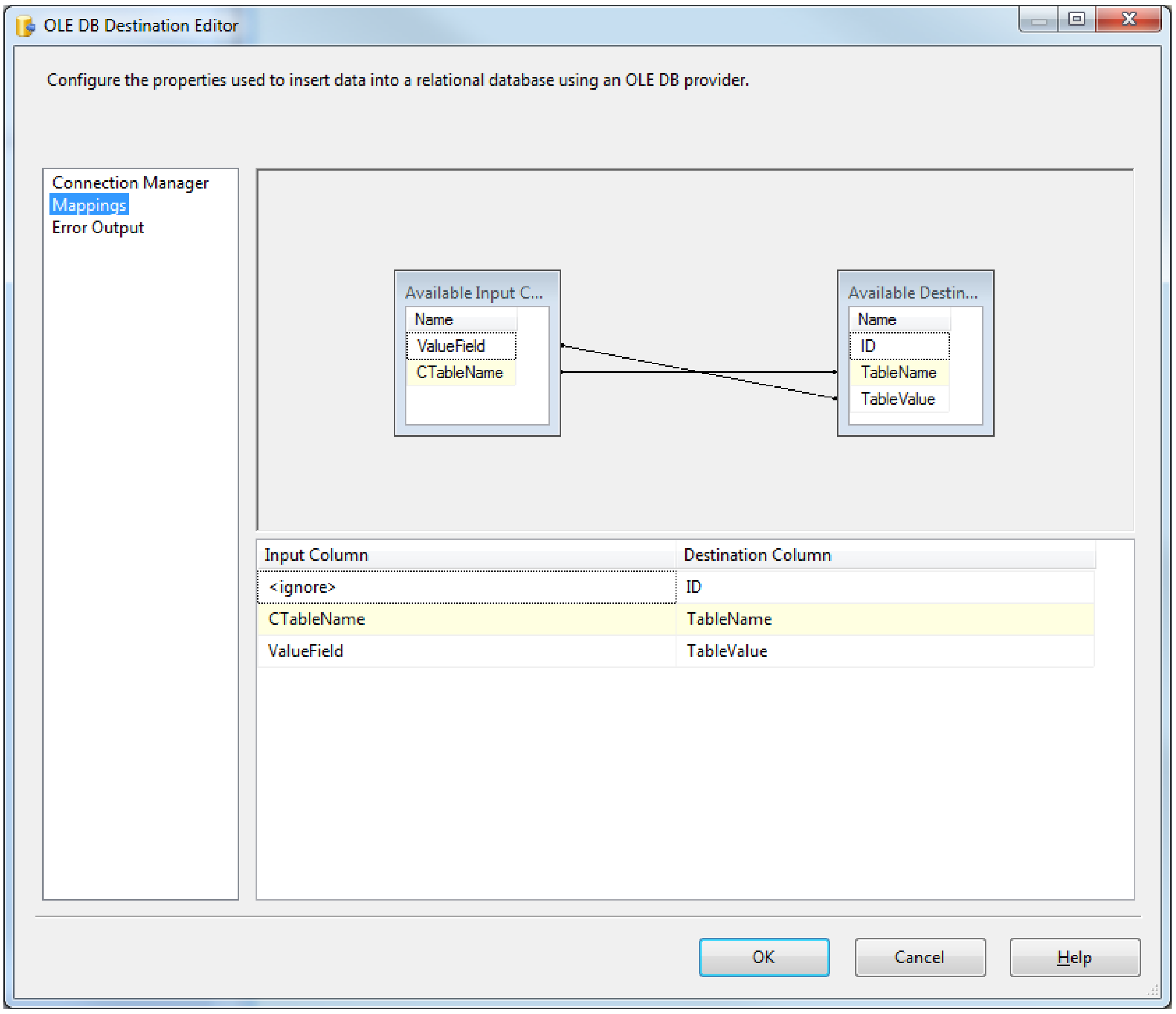


We then add a Derived Column transformation to the data flow. This is a simple copy of the **vTableName** variable as an extra column.


Description:

| Derived Column Name | Derived Column | Expression | Data Type | Le |
|---|---|---|---|---|
| CTableName | <add as new column> | @[User::vTableName] | Unicode string [DT_WSTR] | 1 |

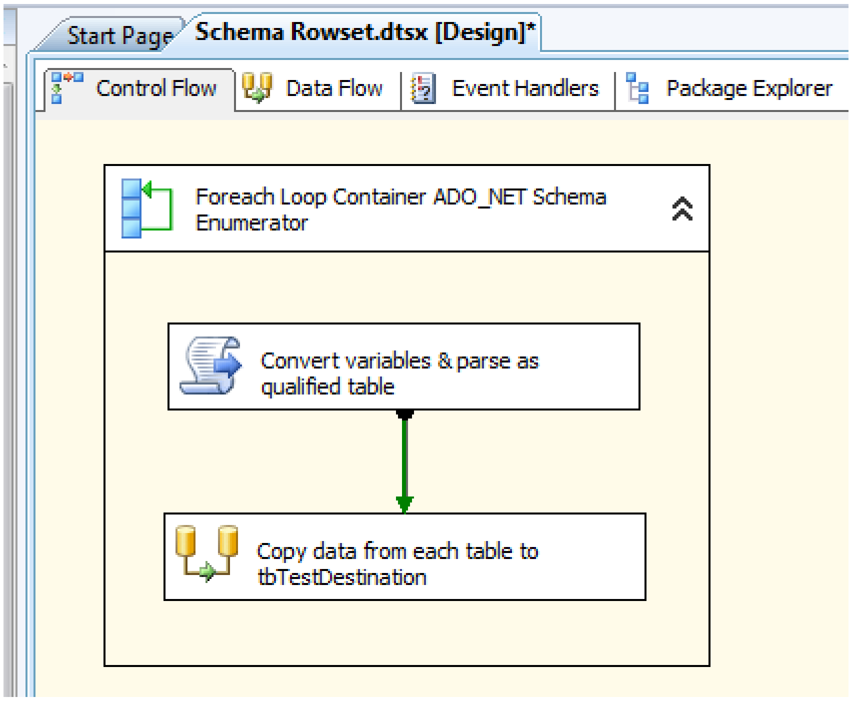
We map this and the **ValueField** to an OLE DB Destination



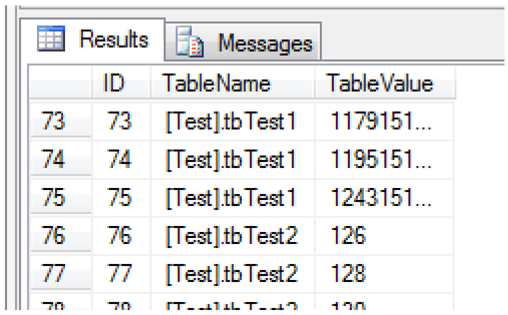
Configure the properties used to insert data into a relational database using an OLE DB provider.

| Input Column | Destination Column |
|---|---|
| <ignore> | ID |
| CTableName | TableName |
| ValueField | TableValue |

…and the package should now be ready to test.


Start Page
Schema Rowset.dtsx [Design]*
Control Flow
Data Flow
Event Handlers
Package Explorer
Foreach Loop Container ADO_NET Schema Enumerator
Convert variables & parse as qualified table
Copy data from each table to tbTestDestination

Run the package and query the **tbTestDestination** table; you should have 124 records looking something like this:



| | ID | TableName | TableValue |
|---|---|---|---|
| 73 | 73 | [Test].tbTest1 | 1179151... |
| 74 | 74 | [Test].tbTest1 | 1195151... |
| 75 | 75 | [Test].tbTest1 | 1243151... |
| 76 | 76 | [Test].tbTest2 | 126 |
| 77 | 77 | [Test].tbTest2 | 128 |

Not a very realistic test, for sure, but the principle should be clear. Not just for looping through tables, but a large number of object types within your database.

Paul Clancy
360Data

See: http://www.360data.nl/Docs/Default.aspx for other SSIS examples