This article describes a way using SSIS to shred a XML column from a source table into its respective columns in the destination table.

Let us suppose we have a simple table with a XML column that needs to be shred into destination table. By shredding I mean we would need to put the value of the nodes into the corresponding column of the destination table. The below listing gives the structure of the source XML table:-

CREATE TABLE [dbo].[TestXML](

[id] [int] IDENTITY(1,1) NOT NULL,

[computed\_column] [xml] NULL

) ON [PRIMARY]

GO

Let’s assume the value has been inserted into our source table dbo.TestXML

INSERT INTO [dbo].[TestXML]

([computed\_column])

VALUES

(' <x:books xmlns:x="urn:books">

<book id="743567">

<author>Arun Mishra</author>

<title>The First Article to SSC</title>

<genre>Fiction</genre>

<price>100.00</price>

<pub\_date>2013-01-28</pub\_date>

<review>Interesting book</review>

</book>

</x:books>')

GO

For the given XML structure let us create the destination table with corresponding columns:-

CREATE TABLE [dbo].[xmltest](

[id] [id] NULL,

[author] [varchar](1000) NULL,

[title] [varchar](1000) NULL,

[genre] [varchar](1000) NULL,

[price] [float] NULL,

[pub\_date] [datetime] NULL,

[review] [varchar](1000) NULL

)

GO

# How to Shred XML column ?

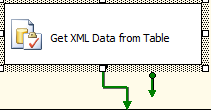
With the structure for the tables created, let us know look into the process for shreddding this XML column into the destination table using a simple SSIS packages.

An SSIS package has built in XML task to shred XML file /supplied XML variable /a direct input. To shred a XML column stored in an table, we would have to consider the below steps:-

1. Catch all the column values
2. Loop through each row
3. Load the data into the corresponding table.

## Step 1:- Catch all the column values:-

The purpose in this step is to catch all the XML value of the source table, in this case, ths would be the values in the computed\_column. SSIS provides us with a very useful Control-Flow task Execute SQL task to capture the required values from the SQL Server table.



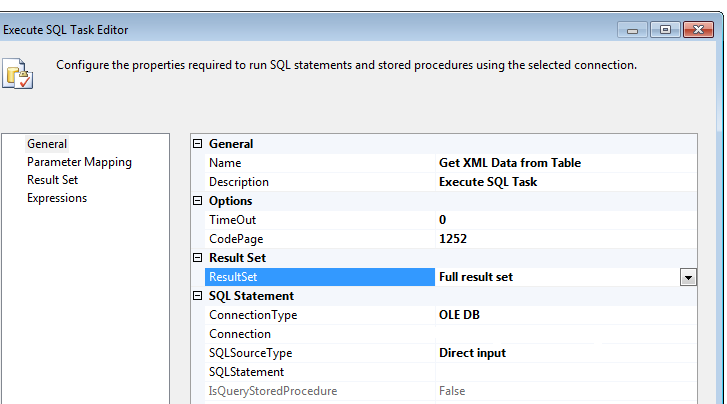
#### Fig1 :- Execute SQL task to capture all column values

Configure the Execute SQL task as below:-

* General Page:- In the general set the below options
* Result sets: - Execute SQL task provides three ways to capture the results of your supplied query, these are :-

1. None: - The query would return no result set.
2. Single: - The query would return a single row as result set.
3. Full Result set: - The query would be returning multiple rows as result set.
4. XML: - The query would return the result in a XML format.

For our problem we would need the Result Set option to be set to “Full Result Set”, why, because our query would be returning more than one rows which we want to load into another table.



#### Fig 2:- Configuration of Execute SQL task

* SQL Statement: - The next configuration item is the SQL statement. Our purpose is to capture the XML column. The SQL syntax for this would be as :-

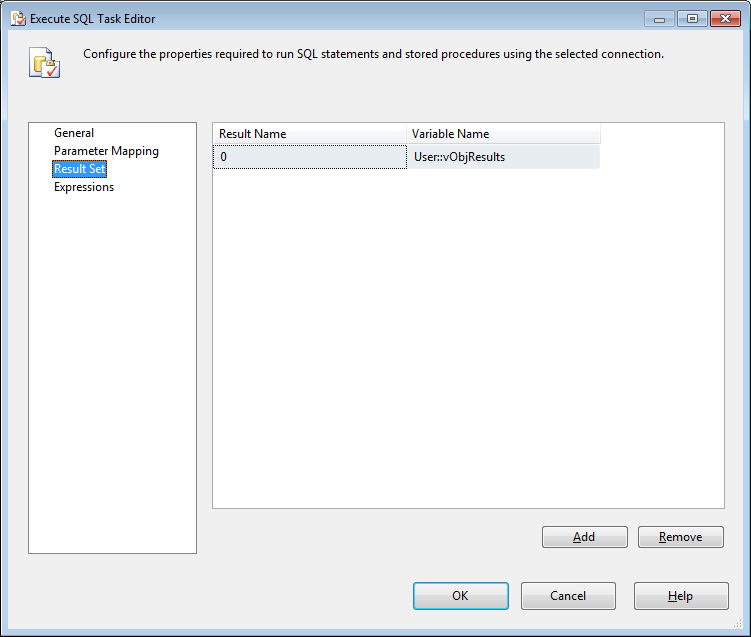
SELECT CONVERT(varchar(4000), computed\_column) AS Expr1

FROM dbo. TestXML

The idea behind the CONVERT() is, as SSIS 2008 still do not have provision for XML data type variable, we would need to cast the output of the query to a string data type variable, to be able to capture it. This variable we would use in the second step of our task (Loop through each row).

Parameter Mapping Page: - As we are not setting any parameter we would leave this one blank.

Result Set Page: - To configure this page, we create a user variable named vObjResults of type Objects. We put the result set name as 0.

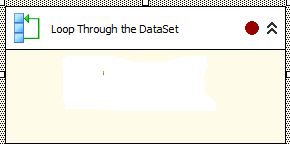


#### Fig 3:- Result Set Configuration Page in Execute SQL task

With the entire configuration for the second step having set up, lets move toward our next step in the process.

## Step 2:- Loop through each row

To loop through the dataset captured from the above task we would require the ForEachLoop task.

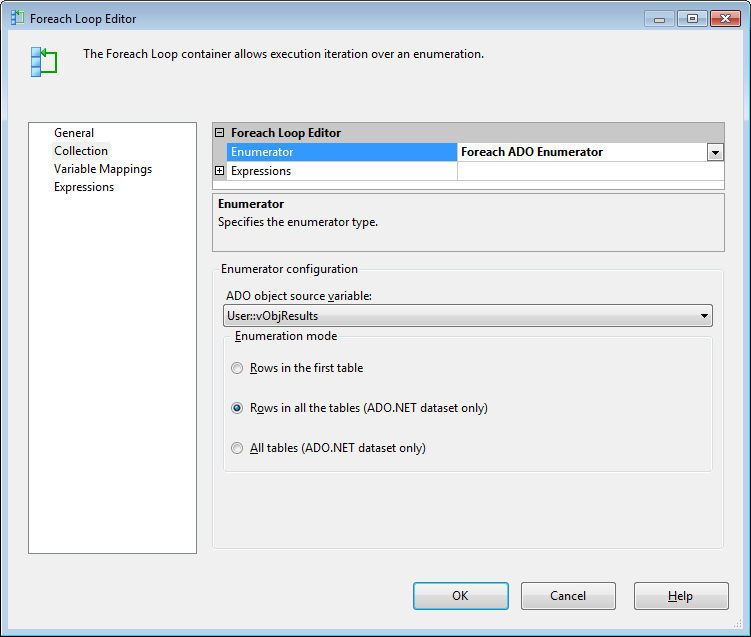


#### Fig 4:- For Each Loop task

Configure the task as below:-

### Collection Page:-

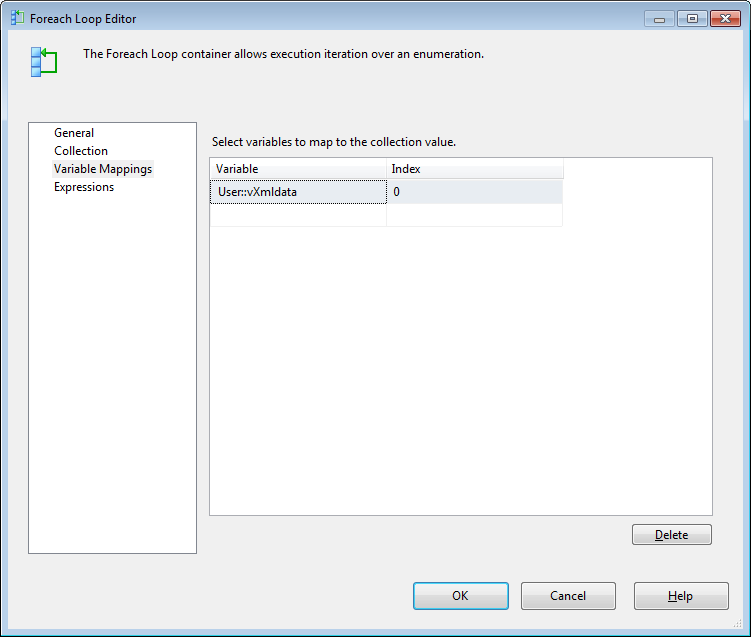
ForEachLoop task provides various Enumerator methods to loop through result set, to loop through our obtained result sets we set enumerator to the ADO Enumerator. Assign the source of the enumerator to the object variable vObjResults and the mode to Rows in all Tables.



#### Fig 5:- For Each Loop task Collection Page configuration

### Variable Mapping Page:-

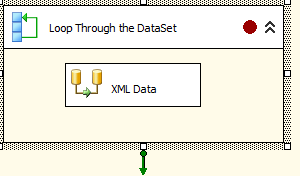
Configure a user variable vXMLData of data type string. And set the index column as 0 as we would be obtaining only the first column.



#### Fig 6:- For Each Loop Variable Mapping page configuration

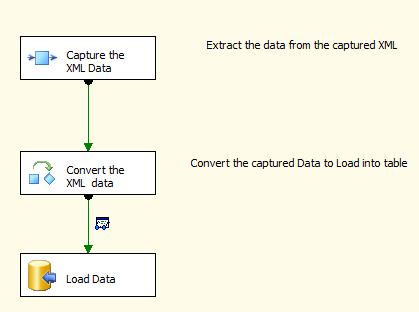
## Step 3:- Load the data into the corresponding table.

To load the Data into our new table dbo.xmlTest we make use of the DataFlow task inside the For Each Loop task.



#### Fig 7:- Data Flow Task inside For Each Loop task

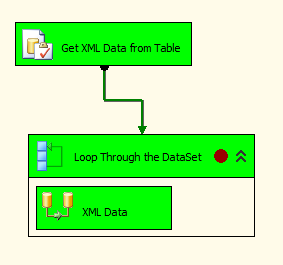
Configure the DataFlow task as below :-



#### Fig8:- Data Flow Source, Transformation and Destination Tasks

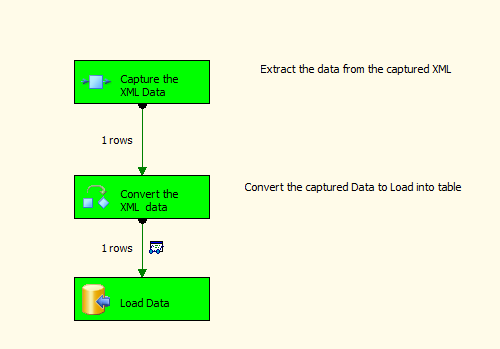
# Execution Results:-

The control flow page shows all green to show the package has executed successfully :-

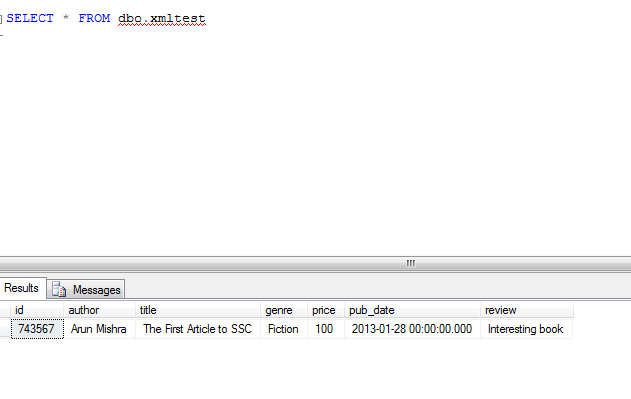


#### Fig 9 :- Execution Result of Control Flow task

The data flow page shows also executes successfully:-



#### Fig 10 :- Execution Result of Data Flow task

If we verify the extracted XML data in the destination table from the SQL Server Management Studio, 

# Conclusion:-

So, we have successfully extracted or shredded our XML data to its respective columns. Most of the cases without using SSIS shredding is carried out using the XQuery or OPEN XML path. The above method shows a way to shred the XML column using SSIS.