http://www.sqlservercentral.com/articles/XML/62290/
Printed 2008/03/26 06:07PM

XML Workshop XVII - Writing a LOOP to process XML elements in TSQL

By Jacob Sebastian, 2008/02/18

Introduction

In the previous sessions of XML Workshop, we have seen several XML processing examples. Some of the examples demonstrated how to generate XML output in a specific structure. Other few examples demonstrated how to read values from XML columns and XML variables.

While working with XML data, it may be common that you will come across situations where you need to iterate over an XML document. Most of the times we can avoid a LOOP by applying some sort of set-based logic and write a single query that does a batch processing. However, there may be cases when we cannot do a batch process. Assume that you need to run a loop over all the nodes of an XML document and execute a stored procedure for each node in the XML document.

In this session, I will show one of the several ways to iterate over an XML document. As I used to mention, there are always more than one way to do a certain programming task. The approach presented here is one of the several possible options that can be used to achieve the given results.

The following pseudo code shows the basic flow of the iteration.

```
"i" = 1
"cnt" = total number of nodes
while "i" <= "cnt" begin
    take the node at position "i"
    process the node
    increment "i"
end
```

Let us see how to translate the above pseudo code to TSQL code.

Sample XML Document

Let us work use the following XML document as the sample data for this session. The below XML document has 5 "Employee" elements. Let us write the code that iterates over the "Employee" elements and prints the "Name" of each employee.

```
<Employees>
    <Employee ID="101">
        <Name>Jacob</Name>
        <Department>IT</Department>
    </Employee>
    <Employee ID="354">
        <Name>Steve</Name>
        <Department>IT</Department>
    </Employee>
    <Employee ID="456">
        <Name>Bob</Name>
        <Department>IT</Department>
    </Employee>
    <Employee ID="478">
        <Name>Joe</Name>
```

```
        <Department>IT</Department>
    </Employee>
    <Employee ID="981">
        <Name>Louis</Name>
        <Department>IT</Department>
    </Employee>
</Employees>
```

Counting Nodes

Before we run the loop, we need to know how many "Employee" elements exist in the XML document. Once we know the total count of "Employee" nodes, we can run a loop from 1 to @count and at each iteration we can process a single XML node.

To count the number of
nodes, let us use the XQuery function "count()". Here is the code that returns the count of nodes.

```
DECLARE @x XML
SET @x =
'<Employees>
  <Employee ID="101">
    <Name>Jacob</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="354">
    <Name>Steve</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="456">
    <Name>Bob</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="478">
    <Name>Joe</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="981">
    <Name>Louis</Name>
    <Department>IT</Department>
  </Employee>
</Employees>'

SELECT
    @x.query('<count>
                { count(/Employees/Employee) }
              </count>') AS Count

/*
output:

Count
-----------------
<count>5</count>
*/
```

We need to retrieve the value "5" from the XQuery result. The following query retrieves the value "5" form the XQuery result.

```
DECLARE @x XML
SET @x =
'<Employees>
  <Employee ID="101">
    <Name>Jacob</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="354">
```

```
    <Name>Steve</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="456">
    <Name>Bob</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="478">
    <Name>Joe</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="981">
    <Name>Louis</Name>
    <Department>IT</Department>
  </Employee>
</Employees>'

SELECT
    @x.query('<count>
                {count(/Employees/Employee)}
              </count>'
            ).value('count[1]','int') AS cnt

/*
output:

cnt
-----------
5
*/
```

Accessing the XML node at a given position

The next task is to see how to access a NODE at a given position. The XQuery function "position()" can be used to access the XML node at a given position.

The following code shows an example of accessing the 3rd "Employee" node using "position()" method.

```
DECLARE @x XML
SET @x =
'<Employees>
  <Employee ID="101">
    <Name>Jacob</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="354">
    <Name>Steve</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="456">
    <Name>Bob</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="478">
    <Name>Joe</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="981">
    <Name>Louis</Name>
    <Department>IT</Department>
  </Employee>
</Employees>'

SELECT
    x.value('Name[1]', 'VARCHAR(20)') AS Name
FROM @x.nodes('/Employees/Employee[position()=3]') e(x)

/*
```

```
output:

Name
--------------------
Bob
*/
```

Using a variable to specify the position

We just saw how to access an XML node at a given position. Now let us see how to access a node at the position specified by a variable.

The following example shows how to use a variable in the XPath expression.

```
DECLARE @x XML
SET @x =
'<Employees>
  <Employee ID="101">
    <Name>Jacob</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="354">
    <Name>Steve</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="456">
    <Name>Bob</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="478">
    <Name>Joe</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="981">
    <Name>Louis</Name>
    <Department>IT</Department>
  </Employee>
</Employees>'

DECLARE @i TINYINT
SELECT @i = 3

SELECT
    x.value('Name[1]', 'VARCHAR(20)') AS Name
FROM
@x.nodes('/Employees/Employee[position()=sql:variable("@i")]') e(x)

/*
output:

Name
--------------------
Bob
*/
```

Writing the LOOP

It is time to look at the final source code. Here is the loop which runs over all the "Employee" elements and prints the "Name" of each employee.

```
DECLARE @x XML
SET @x =
'<Employees>
  <Employee ID="101">
    <Name>Jacob</Name>
```

```
    <Department>IT</Department>
  </Employee>
  <Employee ID="354">
    <Name>Steve</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="456">
    <Name>Bob</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="478">
    <Name>Joe</Name>
    <Department>IT</Department>
  </Employee>
  <Employee ID="981">
    <Name>Louis</Name>
    <Department>IT</Department>
  </Employee>
</Employees>'

-- Total count of <Employee> Nodes
DECLARE @max INT, @i INT
SELECT
    @max = @x.query('<e>
                        { count(/Employees/Employee) }
                    </e>').value('e[1]','int')

-- Set counter variable to 1
SET @i = 1

-- variable to store employee name
DECLARE @EmpName VARCHAR(10)

-- loop starts
WHILE @i <= @max BEGIN
    -- select "Name" to the variable
    SELECT
        @EmpName = x.value('Name[1]', 'VARCHAR(20)')
    FROM
    @x.nodes('/Employees/Employee[position()=sql:variable("@i")]')
    e(x)

    -- print the name
    PRINT @EmpName

    -- increment counter
    SET @i = @i + 1
END
```

Conclusions

We have learned how to use XQuery functions "count()" and "position()". Then we have seen how to access an XML node at the position specified by a variable. Finally, we have seen a loop which runs over all the nodes of an XML document and prints the name of an element.
