http://www.sqlservercentral.com/articles/XQUERY/70548/
Printed 2010/07/13 07:10AM

XML Workshop 26 – SELECT * FROM XML

By Jacob Sebastian, 2010/06/20

Introduction

One of the responsibilities I have at work is the performance tuning of the databases. Most of the times, I need to query tables that I am not familiar with. Without any clue about the columns a table has, I often do the following sequence of activities.

1. SELECT TOP 10 * FROM thatTable
2. Look at the data to locate the columns I need to process
3. Once the columns are located rewrite the query as:
    - SELECT Colx, Coly from thatTable WHERE Colz = SomeValue

I found ***"SELECT *"*** quite handy when dealing with tables that I am not familiar with. SQL Server does not provide such an easy interface for querying XML documents. Looking at the large number of XQuery questions on the online forums, I feel it would have been very easy if we were able to do the following:

1. SELECT * FROM XML document
2. Look at the elements, attributes and their values
3. Locate the elements and/or attributes that we are interested
4. Rewrite the query to select the desired elements and attributes:
    - SELECT value-of element elx, element ely, attribute atx FROM XML Document

It could have been very easy if there is a way to blindly query an XML document (such as a SELECT *) and a way to uniquely identify the element or attribute that we are interested in and finally tell SQL Server to return us those values.

Keeping this in mind, sometime back I wrote a TSQL function which can be used for querying XML variables or columns. You can find the complete source code listing of the function in this post.

Using the XMLTable() Function

Using the XMLTable() function is quite easy. You can pass an XML document into the function and it will return a tabular representation of the XML data. Let us take the example of an RSS feed and see how we can use the XmlTable() function to read information from it.

The following XML document is extracted from the RSS feed generated by my twitter account. It contains two of my recent tweets (I have modified the RSS feed and created a simpler version for the sake of this demonstration)

```
<rss>
  <channel>
    <title>Twitter / jacobsebastian</title>
    <link>http://twitter.com/jacobsebastian</link>
    <description>Twitter updates from Jacob Sebastian</description>
    <language>en-us</language>
    <ttl>40</ttl>
```

```
    <item>
      <title>Announcing winners of TSQL Challenge 23</title>
      <description>
        We have completed the evaluation of TSQL Challenge 23. http://bit.ly/dmQ63G
      </description>
      <pubDate>Fri, 18 Jun 2010 14:41:12 +0000</pubDate>
    </item>
    <item>
      <title>XQuery Lab 57 - Getting Started with OPENXML</title>
      <description>
        This post intends to help you get started with OPENXML(). http://bit.ly/9gThU7
      </description>
      <pubDate>Thu, 17 Jun 2010 12:35:34 +0000</pubDate>
    </item>
   </channel>
</rss>
```

This simplest way to use the XMLTable() function is as follows:

```
-- In case you want to process an XML Variable
DECLARE @x XML
SELECT @x = '-- XML HERE -- '
SELECT * FROM XMLTable(@x)

-- In case you want to process an XML Column
SELECT * FROM YourTable
CROSS APPLY XMLTable(XMLColumn)
```

Let us try to play a little with the XMLTable() function and see how it helps to read information from XML documents easily.

Reading all "title" elements from the RSS Feed

The following example retrieves all the "title" elements from the RSS feed.

```
DECLARE @x XML
SELECT @x = '
<rss>
  <channel>
    <title>Twitter / jacobsebastian</title>
    <link>http://twitter.com/jacobsebastian</link>
    <description>Twitter updates from Jacob Sebastian</description>
    <language>en-us</language>
    <ttl>40</ttl>
    <item>
      <title>Announcing winners of TSQL Challenge 23</title>
      <description>
        We have completed the evaluation of TSQL Challenge 23.
        http://bit.ly/dmQ63G
      </description>
      <pubDate>Fri, 18 Jun 2010 14:41:12 +0000</pubDate>
    </item>
    <item>
      <title>XQuery Lab 57 - Getting Started with OPENXML</title>
      <description>
        This post intends to help you get started with OPENXML().
        http://bit.ly/9gThU7
      </description>
      <pubDate>Thu, 17 Jun 2010 12:35:34 +0000</pubDate>
    </item>
   </channel>
</rss>'
```

```
SELECT NodeName, ParentName, Value FROM XMLTable(@x)
WHERE NodeName = 'Title'
/*
NodeName ParentName Value
-------- ---------- ------------------------------------------
title    channel    Twitter / jacobsebastian
title    item       XQuery Lab 57 - Getting Started with OPENXML
title    item       Announcing winners of TSQL Challenge 23
*/
```

Note that it retrieves the **title** from the **channel** as well as **item** elements. The following example shows how to retrieve the **title** of **item** nodes only.

```
DECLARE @x XML
SELECT @x = '
<rss>
  <channel>
    <title>Twitter / jacobsebastian</title>
    <link>http://twitter.com/jacobsebastian</link>
    <description>Twitter updates from Jacob Sebastian</description>
    <language>en-us</language>
    <ttl>40</ttl>
    <item>
      <title>Announcing winners of TSQL Challenge 23</title>
      <description>
        We have completed the evaluation of TSQL Challenge 23.
        http://bit.ly/dmQ63G
      </description>
      <pubDate>Fri, 18 Jun 2010 14:41:12 +0000</pubDate>
    </item>
    <item>
      <title>XQuery Lab 57 - Getting Started with OPENXML</title>
      <description>
        This post intends to help you get started with OPENXML().
        http://bit.ly/9gThU7
      </description>
      <pubDate>Thu, 17 Jun 2010 12:35:34 +0000</pubDate>
    </item>
   </channel>
</rss>'

SELECT NodeName, ParentName, Value FROM XMLTable(@x)
WHERE NodeName = 'Title'
AND ParentName = 'item'
/*
NodeName ParentName Value
-------- ---------- ------------------------------------------
title    item       XQuery Lab 57 - Getting Started with OPENXML
title    item       Announcing winners of TSQL Challenge 23
*/
```

Auto-Generating the correct XPath expressions

People often send me emails that reads like “I have the following XML document and when I try to read values from xxx node I am getting a NULL”. Almost always I have found that the problem was an incorrect XPath expression. Many people find it really confusing to write the correct XPath expression pointing to a given element or attribute within an XML document.

The XMLTable() function can be used to generate the required XPath expressions for you. The following example demonstrates this.

```
/*
Retrieve the XPath expression and value of the Item elements
*/
SELECT XPath, Value FROM XMLTable(@x)
WHERE NodeName = 'Title'
AND ParentName = 'item'
/*
XPath                              Value
---------------------------------- -------------------------------------------
rss[1]/channel[1]/item[2]/title[1] XQuery Lab 57 - Getting Started with OPENXML
rss[1]/channel[1]/item[1]/title[1] Announcing winners of TSQL Challenge 23
*/

/*
Locate, copy and paste the XPath expression into your XQuery
*/

SELECT @x.value('rss[1]/channel[1]/item[1]/title[1]','VARCHAR(100)') AS Title
/*
Title
-------------------------------------------------------------------------------
Announcing winners of TSQL Challenge 23
*/
```

Reading information from all the *item* elements

The following example shows how to read all the information from all the *item* elements in the XML document we examined earlier.

```
SELECT * FROM XMLTable(@x)
WHERE ParentName = 'Item'
ORDER BY ParentPosition
/*
ParentPosition NodeName    Value
-------------- ----------- ---------------------------------------
1              description We have completed the evaluation of T..
1              pubDate     Fri, 18 Jun 2010 14:41:12 +0000
1              title       Announcing winners of TSQL Challenge ..
2              description This post intends to help you get sta..
2              pubDate     Thu, 17 Jun 2010 12:35:34 +0000
2              title       XQuery Lab 57 - Getting Started with ..
*/
```

Our XML document had two *item* elements. The above query returns values from all the child elements of the two *item* nodes we had. If you want to read a specific element you can filter it by using the *ParentPosition* column.

```
SELECT * FROM XMLTable(@x)
WHERE ParentName = 'Item'
AND ParentPosition = 2
/*
ParentPosition NodeName    Value
-------------- ----------- ---------------------------------------
2              description This post intends to help you get sta..
2              pubDate     Thu, 17 Jun 2010 12:35:34 +0000
2              title       XQuery Lab 57 - Getting Started with ..
*/
```

Viewing the XML Structure and values

If you want to quickly view the XML structure of the document, you can look at the FullPath and TreeView columns. Here is an example:

```
SELECT FullPath, TreeView, Value FROM XMLTable(@x)
order by id
/*
FullPath                      TreeView           Value
----------------------------- ------------------ ----------------------------
rss                           rss                NULL
rss/channel                    |-channel         NULL
rss/channel/description          |-description   Twitter updates from Jacob ..
rss/channel/item                 |-item          NULL
rss/channel/item/description       |-description We have completed the evalu..
rss/channel/item/pubDate           |-pubDate     Fri, 18 Jun 2010 14:41:12 +..
rss/channel/item/title             |-title       Announcing winners of TSQL ..
rss/channel/item                 |-item          NULL
rss/channel/item/description       |-description This post intends to help y..
rss/channel/item/pubDate           |-pubDate     Thu, 17 Jun 2010 12:35:34 +..
rss/channel/item/title             |-title       XQuery Lab 57 - Getting Sta..
rss/channel/language             |-language      en-us
rss/channel/link                 |-link          http://twitter.com/jacobseb..
rss/channel/title                |-title         Twitter / jacobsebastian
rss/channel/ttl                  |-ttl           40
*/
```

Output Reference

The XMLTable() function returns a number of columns that you can use in different ways to solve some of your XQuery requirements. The following is a complete listing of all the columns returned by this function.

1. **ID:** Row ID, a unique sequence number
2. **ParentName:** Immediate parent name of the current element or attribute
3. **ParentPosition:** Position of the parent element within its parent node
4. **Depth:** Depth of the current node in the XML document
5. **NodeName:** Name of the current element or attribute
6. **Position:** Position of the current element in its parent. Always 1 for attributes
7. **NodeType:** Type of current member: "element" or "attribute"
8. **FullPath:** Full path to the current element/attribute
9. **XPath:** XPath expression pointing to the current element/attribute
10. **TreeView:** A tree representation indicating the position of the current element or attribute
11. **Value:** Text value of the current element or attribute
12. **XmlData:** The XML data contained in the current element or attribute

Click here to download the source code of XmlTable() function.

Conclusions

This post demonstrates an easy way to quickly query XML documents. If you find it very hard to deal with XQuery, you might find this function very helpful. XQuery may be a better choice on production environments, for performance reasons.

About the Author

Jacob Sebastian is a SQL Server MVP, Author, Speaker and Trainer. See Jacob's Blog|Profile|XML Resources