XML Workshop V - Reading Values from XML Columns

Introduction

In Part II of my XML Workshop I had presented a few examples which demonstrated how to read values from XML variables. This article presents a few more examples that read values from XML columns.

There were a few questions in the discussion forums asking more detailed examples that show how to read values from XML columns. One of the questions asked in the forum recently was about reading values from the report server *subscriptions* table . In this article, I am trying to present a set of detailed examples that explain how to read values from the *subscriptions* table of report server.

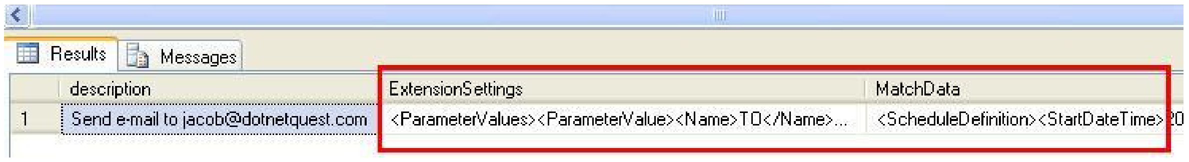
Sample Data

For the purpose of this example, I have created a report subscription. The report has to be emailed to me on a weekly schedule, every Monday, Tuesday, Wednesday and Thursday. Now it is time to play with the *subscriptions* table.

Let us first query the *subscriptions* table and see how the data looks like. Let us run the following query to retrieve the report server subscription details. I have created a subscription with the name 'Send e-mail to jacob@dotnetquest.com'. You should change the subscription name to your own subscription.

```
SELECT
    [description],
    ExtensionSettings,
    MatchData
FROM Subscriptions
WHERE description = 'Send e-mail to jacob@dotnetquest.com'
```

The result shows that the columns *ExtensionSettings* and *MatchData* contain data in XML format. However the data type is not **XML**. They are **NTEXT** columns which contain text data which has an XML structure.

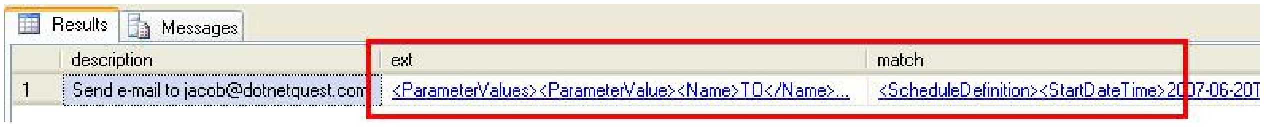


	description	ExtensionSettings	MatchData
1	Send e-mail to jacob@dotnetquest.com	<ParameterValues><ParameterValue><Name>TO</Name>...	<ScheduleDefinition><StartDateTime>20

Before we can use the XML methods on the columns, we need to convert the values to XML data type. The following query converts the column to XML.

```
SELECT
    [description],
    CAST(ExtensionSettings AS XML) ext,
    CAST(MatchData AS XML) AS match
FROM Subscriptions
WHERE description = 'Send e-mail to jacob@dotnetquest.com'
```

Note that we have the results as XML values.



	description	ext	match
1	Send e-mail to jacob@dotnetquest.com	<ParameterValues><ParameterValue><Name>TO</Name>...	<ScheduleDefinition><StartDateTime>2007-06-20T

Now let us start reading values from the XML columns. Let us first read values from the *ExtensionSettings* column. This column contains the information I have configured for the e-mail delivery. Run the following query.

```
/*
XML column value:

<ParameterValues>
```

```
  <ParameterValue>
    <Name>TO</Name>
    <Value>jacob@dotnetquest.com</Value>
  </ParameterValue>
  <ParameterValue>
    <Name>CC</Name>
    <Value>jacob@reliancesp.com</Value>
  </ParameterValue>
  <ParameterValue>
    <Name>BCC</Name>
    <Value>jacob@excellenceinfonet.com</Value>
  </ParameterValue>
  <ParameterValue>
    <Name>ReplyTo</Name>
    <Value>jacob@dotnetquest.com</Value>
  </ParameterValue>
  <ParameterValue>
    <Name>IncludeReport</Name>
    <Value>True</Value>
  </ParameterValue>
  <ParameterValue>
    <Name>RenderFormat</Name>
    <Value>MHTML</Value>
  </ParameterValue>
  <ParameterValue>
    <Name>Subject</Name>
    <Value>@ReportName was executed at @ExecutionTime</Value>
  </ParameterValue>
  <ParameterValue>
    <Name>Comment</Name>
    <Value>Ah..this is a comment.</Value>
  </ParameterValue>
  <ParameterValue>
    <Name>IncludeLink</Name>
    <Value>True</Value>
  </ParameterValue>
  <ParameterValue>
    <Name>Priority</Name>
    <Value>NORMAL</Value>
  </ParameterValue>
</ParameterValues>
*/

SELECT
x.ext.value('Name[1]', 'varchar(20)') as [Name],
x.ext.value('Value[1]', 'varchar(30)') as [Value]
FROM
(
    SELECT
        [description],
        CAST(ExtensionSettings AS XML) ext
    FROM Subscriptions
) AS P
CROSS APPLY ext.nodes('//ParameterValues/ParameterValue') x(ext)
WHERE description = 'Send e-mail to jacob@dotnetquest.com'

/*
OUTPUT:

Name                  Value
--------------------- ------------------------------
TO                     jacob@dotnetquest.com
CC                     jacob@reliancesp.com
BCC                   jacob@excellenceinfonet.com
ReplyTo               jacob@dotnetquest.com
IncludeReport          True
RenderFormat           MHTML
Subject               @ReportName was executed at @E
```

```
Comment              Ah..this is a comment.
IncludeLink          True
Priority              NORMAL
*/
```

Now let us read the values from the *MatchData* column. This column contains the information about the scheduling I have configured. In this example, I have scheduled an e-mail delivery on Monday, Tuesday and Wednesday. Run the following query.

```
/*
<ScheduleDefinition>
  <StartDateTime>2007-06-20T08:00:00.000+05:30</StartDateTime>
  <EndDate>2007-07-12</EndDate>
  <WeeklyRecurrence>
    <WeeksInterval>1</WeeksInterval>
    <DaysOfWeek>
      <Monday>true</Monday>
      <Tuesday>true</Tuesday>
      <Wednesday>true</Wednesday>
      <Thursday>true</Thursday>
    </DaysOfWeek>
  </WeeklyRecurrence>
</ScheduleDefinition>
*/
SELECT
match.value('(//ScheduleDefinition/StartDateTime)[1]','varchar(30)') AS
StartDateTime,
match.value('(//ScheduleDefinition/EndDate)[1]','varchar(10)') AS EndDate,
match.value('(//ScheduleDefinition/WeeklyRecurrence/WeeksInterval)[1]','varchar(10)
AS WeeksInterval,
match.value('(//ScheduleDefinition/WeeklyRecurrence/DaysOfWeek/Sunday)[1]','varchar
AS Sunday,
match.value('(//ScheduleDefinition/WeeklyRecurrence/DaysOfWeek/Monday)[1]','varchar
AS Monday,
match.value('(//ScheduleDefinition/WeeklyRecurrence/DaysOfWeek/Tuesday)[1]','varcha
AS Tuesday,
match.value('(//ScheduleDefinition/WeeklyRecurrence/DaysOfWeek/Wednesday)[1]','varc
AS Wednesday,
match.value('(//ScheduleDefinition/WeeklyRecurrence/DaysOfWeek/Thursday)[1]','varch
AS Thursday,
match.value('(//ScheduleDefinition/WeeklyRecurrence/DaysOfWeek/Friday)[1]','varchar
AS Friday,
match.value('(//ScheduleDefinition/WeeklyRecurrence/DaysOfWeek/Saturday)[1]','varch
AS Saturday
FROM
(
    SELECT
        [description],
        CAST(MatchData AS XML) AS match
    FROM Subscriptions
) AS P
WHERE description = 'Send e-mail to jacob@dotnetquest.com'

/*
OUTPUT:

StartDateTime                 EndDate    WeeksInterval Sunday Monday Tuesday
Wednesday Thursday Friday Saturday
------------------------------ ---------- ------------- ------ ------ -------
--------- --------- ------ --------
2007-06-20T08:00:00.000+05:30 2007-07-12 1             NULL   true   true    true
 true      NULL   NULL
*/
```

Note that we are passing the complete path to the required node in the *value* method. Let us make the syntax simpler by using the *nodes* method. The *nodes* method returns a collection of XML nodes from which you can retrieve the desired values. The XPath expression in the *value* method should be relative to the path of the nodes retrieved by the collection. Here is a simpler query which gives the same result.

```
SELECT
x.m.value('StartDateTime[1]','varchar(30)') AS StartDateTime,
x.m.value('EndDate[1]','varchar(10)') AS EndDate,
x.m.value('(WeeklyRecurrence/WeeksInterval)[1]','varchar(10)') AS WeeksInterval,
x.m.value('(WeeklyRecurrence/DaysOfWeek/Sunday)[1]','varchar(5)') AS Sunday,
x.m.value('(WeeklyRecurrence/DaysOfWeek/Monday)[1]','varchar(5)') AS Monday,
x.m.value('(WeeklyRecurrence/DaysOfWeek/Tuesday)[1]','varchar(5)') AS Tuesday,
x.m.value('(WeeklyRecurrence/DaysOfWeek/Wednesday)[1]','varchar(5)') AS Wednesday,
x.m.value('(WeeklyRecurrence/DaysOfWeek/Thursday)[1]','varchar(5)') AS Thursday,
x.m.value('(WeeklyRecurrence/DaysOfWeek/Friday)[1]','varchar(5)') AS Friday,
x.m.value('(WeeklyRecurrence/DaysOfWeek/Saturday)[1]','varchar(5)') AS Saturday
FROM
(
    SELECT
        [description],
        CAST(MatchData AS XML) AS match
    FROM Subscriptions
) AS P
CROSS APPLY match.nodes('//ScheduleDefinition') x(m)
WHERE description = 'Send e-mail to jacob@dotnetquest.com'
```

XPath is really fun. Let us make the query even simpler. Let us add one more level to the nodes(), '//ScheduleDefinition/WeeklyRecurrence'. With this change, all the XPath expressions given in the *value()* method should be relative to the location of the path given in the *nodes()* method. Note the first fields, which uses "..\" to point to a value in the parent node. Run the following query.

```
SELECT
x.m.value('../StartDateTime[1]','varchar(30)') AS StartDateTime,
x.m.value('../EndDate[1]','varchar(10)') AS EndDate,
x.m.value('(WeeksInterval)[1]','varchar(10)') AS WeeksInterval,
x.m.value('(DaysOfWeek/Sunday)[1]','varchar(5)') AS Sunday,
x.m.value('(DaysOfWeek/Monday)[1]','varchar(5)') AS Monday,
x.m.value('(DaysOfWeek/Tuesday)[1]','varchar(5)') AS Tuesday,
x.m.value('(DaysOfWeek/Wednesday)[1]','varchar(5)') AS Wednesday,
x.m.value('(DaysOfWeek/Thursday)[1]','varchar(5)') AS Thursday,
x.m.value('(DaysOfWeek/Friday)[1]','varchar(5)') AS Friday,
x.m.value('(DaysOfWeek/Saturday)[1]','varchar(5)') AS Saturday

FROM
(
    SELECT
        [description],
        CAST(MatchData AS XML) AS match
    FROM Subscriptions
) AS P
CROSS APPLY match.nodes('//ScheduleDefinition/WeeklyRecurrence') x(m)
WHERE description = 'Send e-mail to jacob@dotnetquest.com'
```

I have one more version of the query. This version uses a longer XPath expression in the *nodes()* method and uses a relative path in the *value()* method.

```
SELECT
x.m.value('../../StartDateTime[1]','varchar(30)') AS StartDateTime,
x.m.value('../../EndDate[1]','varchar(10)') AS EndDate,
x.m.value('../WeeksInterval[1]','varchar(10)') AS WeeksInterval,
x.m.value('Sunday[1]','varchar(5)') AS Sunday,
x.m.value('Monday[1]','varchar(5)') AS Monday,
x.m.value('Tuesday[1]','varchar(5)') AS Tuesday,
x.m.value('Wednesday[1]','varchar(5)') AS Wednesday,
x.m.value('Thursday[1]','varchar(5)') AS Thursday,
x.m.value('Friday[1]','varchar(5)') AS Friday,
x.m.value('Saturday[1]','varchar(5)') AS Saturday
FROM
(
    SELECT
        [description],
        CAST(MatchData AS XML) AS match
    FROM Subscriptions
) AS P
CROSS APPLY match.nodes('//ScheduleDefinition/WeeklyRecurrence/DaysOfWeek') x(m)
WHERE description = 'Send e-mail to jacob@dotnetquest.com'
```

Now let us combine the two queries. Here is the query which retrieves the values of the two columns we saw in the above examples.

```
SELECT
x.ext.value('Name[1]', 'varchar(20)') as [Name],
x.ext.value('Value[1]', 'varchar(30)') as [Value],
LEFT(y.m.value('../../StartDateTime[1]','varchar(30)'),10) AS StartDate,
y.m.value('../../EndDate[1]','varchar(10)') AS EndDate,
y.m.value('../WeeksInterval[1]','varchar(1)') AS WI,
y.m.value('Sunday[1]','varchar(5)') AS Sun,
y.m.value('Monday[1]','varchar(5)') AS Mon,
y.m.value('Tuesday[1]','varchar(5)') AS Tue,
y.m.value('Wednesday[1]','varchar(5)') AS Wed,
y.m.value('Thursday[1]','varchar(5)') AS Thu,
y.m.value('Friday[1]','varchar(5)') AS Fri,
y.m.value('Saturday[1]','varchar(5)') AS Sat
FROM
(
    SELECT
        [description],
        CAST(ExtensionSettings AS XML) ext,
        CAST(MatchData AS XML) AS match
    FROM Subscriptions
) AS P
CROSS APPLY match.nodes('//ScheduleDefinition/WeeklyRecurrence/DaysOfWeek') y(m)
CROSS APPLY ext.nodes('//ParameterValues/ParameterValue') x(ext)
WHERE description = 'Send e-mail to jacob@dotnetquest.com'

/*
OUTPUT:

Name                 Value                          StartDate EndDate    WI   Sun
Mon   Tue   Wed   Thu   Fri   Sat
-------------------- ------------------------------ ---------- ---------- ----
----- ----- ----- ----- ----- ----- -----
TO                   jacob@dotnetquest.com          2007-06-20 2007-07-12 1    NULL
true true true true NULL NULL
CC                   jacob@reliancesp.com           2007-06-20 2007-07-12 1
NULL true true true true NULL NULL
BCC                  jacob@excellenceinfonet.com    2007-06-20 2007-07-12 1    NULL
true true true true NULL NULL
ReplyTo              jacob@dotnetquest.com          2007-06-20 2007-07-12 1    NULL
true true true true NULL NULL
IncludeReport        True                           2007-06-20 2007-07-12 1
NULL true true true true NULL NULL
RenderFormat         MHTML                          2007-06-20 2007-07-12 1    NULL
true true true true NULL NULL
Subject              @ReportName was executed at @E 2007-06-20 2007-07-12 1    NULL
true true true true NULL NULL
Comment              Ah..this is a comment.         2007-06-20 2007-07-12 1    NULL
true true true true NULL NULL
IncludeLink          True                           2007-06-20 2007-07-12 1    NULL
true true true true NULL NULL
Priority             NORMAL                         2007-06-20 2007-07-12 1
NULL true true true true NULL NULL
*/
```

Conclusions

This installment of XML Workshop aims at explaining the usage of XQuery to retrieve values from an XML column. It demonstrates the usage of CROSS APPLY, the new keyword introduced by SQL Server 2005. It also gives a basic introduction to the usage of XPath expressions.