http://www.sqlservercentral.com/articles/SS2K5+-+XML/3059/
Printed 2007/11/07 02:21PM

XML Workshop IV - FOR XML EXPLICIT

By Jacob Sebastian, 2007/06/28

Introduction

This is the fourth installment of my XML workshop which aims at explaining/demonstrating XML processing in SQL Server 2005. If you have not read the previous articles in this series, I would suggest that you read them before proceeding. Here are the links to the previous articles.

- Part I focuses on generating XML data from the results of a query using *FOR XML* directive. It explains the usage of *AUTO* and *RAW*.
- Part II focuses on retrieving values from the elements and attributes of an XML variable.
- Part III focuses on performing more advanced XML generation using the *PATH* directive along with *FOR XML*.

In the previous articles we had seen several examples which demonstrated different ways to generate XML data. Using *FOR XML* directive along with *AUTO, RAW* and *PATH* we can achieve almost all XML formatting/processing requirements. In this article, I would present a few examples which show the power of the keyword *EXPLICIT*.

FOR XML EXPLICIT

Using *EXPLICIT* is much more complex than using *AUTO, RAW* and *PATH*. Almost all XML formatting requirements can be achieved by using *AUTO, RAW* and *PATH*. Their usage is pretty simple and basic. However, there are times when we need more complex XML formatting requirements which *AUTO, RAW* and *PATH* cannot handle. *EXPLICIT* supports very complex XML formatting and gives you more control over how the output is generated.

EXPLICIT expects that the query results will be in a specific structure. All the information needed for the formatting is provided in the query results. The following examples will explain this in detail.

Let us try to generate the XML structure we created previously with *FOR XML PATH*. As I have mentioned earlier, using *EXPLICIT* is much more complex than using the other directives. To present it as simple as possible, I will take a step by step approach to generate the sample XML structure that we need to generate.

Here is the output that we need.

```
<CustomersByRegion>
  <Country name="England" currency="Pound Sterling">
    <City name="London">
      <Customer id="TH" name="Thomas Hardy" phone="444-444-4444" />
    </City>
  </Country>
  <Country name="India" currency="Rupee">
    <City name="New Delhi">
      <Customer id="JS" name="Jacob Sebastian" phone="555-555-5555" />
    </City>
  </Country>
  <Country name="USA" currency="US Dollars">
```

```
    <City name="NJ">
      <Customer id="EN" name="Elizabeth Lincoln" phone="333-333-3333" />
      <Customer id="MK" name="John Mark" phone="111-111-1111" />
    </City>
    <City name="NY">
      <Customer id="WS" name="Will Smith" phone="222-222-2222" />
    </City>
  </Country>
</CustomersByRegion>
```

Let us now start generating the above XML structure using the *EXPLICIT* mode with *FOR XML*. As I have mentioned earlier, the usage of EXPLICIT is a bit complex. So I will present a step by step example which will explain the usage in detail.

Step 1: Generate the *Country* Node

Let us see, how we could generate the *Country* node. EXPLICIT expects the query results to be in a specific format. The following example shows the structure of the query results and the XML structure generated from the query results.

```
/*
Let us first generate the results in the required structure. Once the
results are generated in the required format, we shall go ahead and
generate the XML.
*/

SELECT
    1 AS Tag,
    NULL AS Parent,
    c.CountryName AS 'Country!1!name',
    c.Currency AS 'Country!1!currency'
FROM
    Countries c
/*
OUTPUT:

Tag         Parent     Country!1!name       Country!1!currency
----------- ---------- -------------------- --------------------
1           NULL        USA                  US Dollars
1           NULL        England              Pound Sterling
1           NULL        India                 Rupee

"Tag" is a mandatory column. It tells the XML generator the level of the
element
    in the XML hierarchy. In the example, I have put "1" to tell the XML
    generator that it is the top LEVEL node.
"Parent" is the second mandatory column. It tells the XML generator about
the
    parent of the current node. I have put NULL to tell the XML generator
    that the current element does not have a parent node.

After the first 2 mandatory (pre-defined) columns, I have put the data that
I need.

"Country!1!name"
    "Country" is the name of the element
    "1" specifies the LEVEL of the node in the hierarchy
    "name" is the name of the attribute
"Country!1!currency"
    "Country" is the name of the element
    "1" specifies the LEVEL of the node in the hierarchy
    "currency" is the name of the attribute
```

```
*/

/*
Now let us generate the XML using FOR XML EXPLICIT
*/
SELECT
    1 AS Tag,
    NULL AS Parent,
    c.CountryName AS 'Country!1!name',
    c.Currency AS 'Country!1!currency'
FROM
    Countries c
FOR XML EXPLICIT

/*
OUTPUT:

<Country name="USA" currency="US Dollars" />
<Country name="England" currency="Pound Sterling" />
<Country name="India" currency="Rupee" />
*/
```

Step 2: Generate the *City* Node

Now let us modify our query and generate the next level node. The following query generates an XML structure with the first 2 nodes that we require.

```
/*
Just like what we did in the previous example, let us first generate
the result set and have a close look at its structure. Here is the result
set
that we need to generate the first two nodes (country and city).
*/

SELECT
    1 AS Tag,
    NULL AS Parent,
    c.CountryName AS 'Country!1!name',
    c.Currency AS 'Country!1!currency',
    NULL AS 'City!2!name'
FROM
    Countries c

UNION ALL

SELECT
    2 AS Tag,
    1 AS Parent,
    Country.CountryName,
    Country.Currency,
    City.CityName
FROM Cities City
INNER JOIN Countries Country ON (Country.CountryID = City.CountryID)
ORDER BY 'Country!1!name', 'City!2!name'

/*
OUTPUT:

Tag         Parent      Country!1!name       Country!1!currency    City!2!
name
----------- ----------- -------------------- -------------------- ---------
----------
1           NULL        England              Pound Sterling       NULL
```

```
2           1           England             Pound Sterling       London
1           NULL        India                Rupee                NULL
2           1           India                Rupee                New Delhi
1           NULL        USA                 US Dollars           NULL
2           1           USA                 US Dollars           NJ
2           1           USA                 US Dollars           NY

"Tag"
    Note that, this time we have a few records with value "2" in the
results
    The records with tag "2" are the second level nodes.
"Parent"
    Note that we have records with value "1" which says that those records
have
    a parent node. The value in the column "Parent" refers to "Tag" of the
parent
    record.
"City!2!name"
    "City" is the name of the second level element
    "2" refers to the "Tag" which specifies that it is the second level
node.
    "name" is the name of the attribute.
*/

/*
Now let us generate the XML and see the results.
*/

SELECT
    1 AS Tag,
    NULL AS Parent,
    c.CountryName AS 'Country!1!name',
    c.Currency AS 'Country!1!currency',
    NULL AS 'City!2!name'
FROM
    Countries c

UNION ALL

SELECT
    2 AS Tag,
    1 AS Parent,
    Country.CountryName,
    Country.Currency,
    City.CityName
FROM Cities City
INNER JOIN Countries Country ON (Country.CountryID = City.CountryID)
ORDER BY 'Country!1!name', 'City!2!name'
FOR XML EXPLICIT

/*
OUTPUT:

<Country name="England" currency="Pound Sterling">
  <City name="London" />
</Country>
<Country name="India" currency="Rupee">
  <City name="New Delhi" />
</Country>
<Country name="USA" currency="US Dollars">
  <City name="NJ" />
  <City name="NY" />
</Country>
*/
```

Step 3: Generate the *Customer* element

So far we had been progressing steadily. Now let us generate the third level nodes. The following example does that.

```
/*
As usual, let us first generate the result set and understand its structure
before proceeding with XML generation.
*/

SELECT
    1 AS Tag,
    NULL AS Parent,
    c.CountryName AS 'Country!1!name',
    c.Currency AS 'Country!1!currency',
    NULL AS 'City!2!name',
    NULL AS 'Customer!3!id',
    NULL AS 'Customer!3!name',
    NULL AS 'Customer!3!phone'
FROM
    Countries c
UNION ALL
SELECT
    2 AS Tag,
    1 AS Parent,
    Country.CountryName,
    Country.Currency,
    City.CityName,
    NULL,
    NULL,
    NULL
FROM Cities City
INNER JOIN Countries Country ON (Country.CountryID = City.CountryID)
UNION ALL
SELECT
        3 AS Tag,
        2 AS Parent,
        Country.CountryName AS [name],
        Country.Currency,
        City.CityName AS [name],
        Customer.CustomerNumber AS [id],
        Customer.CustomerName AS [name],
        Customer.Phone
    FROM
        Customers Customer
        INNER JOIN Cities City ON (City.CityID = Customer.CityID)
        INNER JOIN Countries Country ON (Country.CountryID =
City.CountryID)
ORDER BY 'Country!1!name', 'City!2!name'

/*
OUTPUT:

Tag Parent Country!1!name Country!1!currency   City!2!name Customer!3!id
Customer!3!name   Customer!3!phone
---- ------ -------------- -------------------- ----------- ------------- -
---------------- ----------------
1    NULL   England        Pound Sterling       NULL        NULL
NULL             NULL
2    1      England        Pound Sterling       London      NULL          NULL
          NULL
3    2      England        Pound Sterling       London      TH
Thomas Hardy     444-444-4444
```

```
   53 1    NULL   India         Rupee                NULL        NULL
NULL             NULL
   54 2    1     India         Rupee                New Delhi   NULL         NULL
          NULL
   55 3    2     India         Rupee                New Delhi   JS
Jacob Sebastian   555-555-5555
   56 1    NULL   USA            US Dollars          NULL        NULL
NULL             NULL
   57 2    1     USA            US Dollars          NJ          NULL         NULL
          NULL
   58 3    2     USA            US Dollars          NJ          EN
Elizabeth Lincoln 333-333-3333
   59 3    2     USA            US Dollars          NY          MK
John Mark         111-111-1111
   60 2    1     USA            US Dollars          NY          NULL         NULL
          NULL
   61 3    2     USA            US Dollars          NY          WS
Will Smith        222-222-2222
   62
   63 "Tag"
   64     Note that, this time we have a few records with value "3" which refers
to the third level
   65     in the XML hierarchy.
   66 "Parent"
   67     The new records (Tag = 3) have their parent set to "2" to indicate that
the parent of this
   68     element is the record with "Tag" having a value of "2"
   69 "Customer!3!*"
   70     These three columns contain the information needed for the third level
node.
   71 */
   72
   73 /*
   74 Let us GENERATE the XML now.
   75 */
   76
   77 SELECT
   78     1 AS Tag,
   79     NULL AS Parent,
   80     c.CountryName AS 'Country!1!name',
   81     c.Currency AS 'Country!1!currency',
   82     NULL AS 'City!2!name',
   83     NULL AS 'Customer!3!id',
   84     NULL AS 'Customer!3!name',
   85     NULL AS 'Customer!3!phone'
   86 FROM
   87     Countries c
   88 UNION ALL
   89 SELECT
   90     2 AS Tag,
   91     1 AS Parent,
   92     Country.CountryName,
   93     Country.Currency,
   94     City.CityName,
   95     NULL,
   96     NULL,
   97     NULL
   98 FROM Cities City
   99 INNER JOIN Countries Country ON (Country.CountryID = City.CountryID)
  100 UNION ALL
  101 SELECT
  102         3 AS Tag,
  103         2 AS Parent,
  104         Country.CountryName AS [name],
```

```
105         Country.Currency,
106         City.CityName AS [name],
107         Customer.CustomerNumber AS [id],
108         Customer.CustomerName AS [name],
109         Customer.Phone
110     FROM
111         Customers Customer
112         INNER JOIN Cities City ON (City.CityID = Customer.CityID)
113         INNER JOIN Countries Country ON (Country.CountryID =
City.CountryID)
114 ORDER BY 'Country!1!name', 'City!2!name'
115 FOR XML EXPLICIT
116
117 /*
118 OUTPUT:
119
120 <Country name="England" currency="Pound Sterling">
121   <City name="London">
122     <Customer id="TH" name="Thomas Hardy" phone="444-444-4444" />
123   </City>
124 </Country>
125 <Country name="India" currency="Rupee">
126   <City name="New Delhi">
127     <Customer id="JS" name="Jacob Sebastian" phone="555-555-5555" />
128   </City>
129 </Country>
130 <Country name="USA" currency="US Dollars">
131   <City name="NJ">
132     <Customer id="EN" name="Elizabeth Lincoln" phone="333-333-3333" />
133     <Customer id="MK" name="John Mark" phone="111-111-1111" />
134   </City>
135   <City name="NY">
136     <Customer id="WS" name="Will Smith" phone="222-222-2222" />
137   </City>
138 </Country>
139 */
```

Step 4: Generate the *Root* Node

We are almost done. We have all the data that we need in the desired XML format. However we are missing the *root* node, *CustomersByRegion*. Unlike *AUTO* and *RAW* modes, *EXPLICIT* does not provide a way to generate a *root* node. To generate a root node, let us generate a dummy result set and UNION it with our data. The dummy result set will be the TOP MOST node and other nodes will be pushed downwards by one level. The following example shows it in detail.

```
 1 SELECT
 2     1 AS Tag,
 3     NULL AS Parent,
 4     NULL AS 'CustomersByRegion!1', -- empty root element
 5     NULL AS 'Country!2!name',
 6     NULL AS 'Country!2!currency',
 7     NULL AS 'City!3!name',
 8     NULL AS 'Customer!4!id',
 9     NULL AS 'Customer!4!name',
10     NULL AS 'Customer!4!phone'
11 UNION ALL
12 SELECT
13     2 AS Tag,
14     1 AS Parent,
15     NULL,
16     c.CountryName,
17     c.Currency,
```

```
    NULL,
    NULL,
    NULL,
    NULL
FROM
    Countries c
UNION ALL
SELECT
    3 AS Tag,
    2 AS Parent,
    NULL,
    Country.CountryName,
    Country.Currency,
    City.CityName,
    NULL,
    NULL,
    NULL
FROM Cities City
INNER JOIN Countries Country ON (Country.CountryID = City.CountryID)
UNION ALL
SELECT
        4 AS Tag,
        3 AS Parent,
        NULL,
        Country.CountryName AS [name],
        Country.Currency,
        City.CityName AS [name],
        Customer.CustomerNumber AS [id],
        Customer.CustomerName AS [name],
        Customer.Phone
    FROM
        Customers Customer
        INNER JOIN Cities City ON (City.CityID = Customer.CityID)
        INNER JOIN Countries Country ON (Country.CountryID =
City.CountryID)
ORDER BY 'Country!2!name', 'City!3!name', Parent
FOR XML EXPLICIT

/*
OUTPUT:

<CustomersByRegion>
  <Country name="England" currency="Pound Sterling">
    <City name="London">
      <Customer id="TH" name="Thomas Hardy" phone="444-444-4444" />
    </City>
  </Country>
  <Country name="India" currency="Rupee">
    <City name="New Delhi">
      <Customer id="JS" name="Jacob Sebastian" phone="555-555-5555" />
    </City>
  </Country>
  <Country name="USA" currency="US Dollars">
    <City name="NJ">
      <Customer id="EN" name="Elizabeth Lincoln" phone="333-333-3333" />
    </City>
    <City name="NY">
      <Customer id="MK" name="John Mark" phone="111-111-1111" />
      <Customer id="WS" name="Will Smith" phone="222-222-2222" />
    </City>
  </Country>
</CustomersByRegion>
*/
```

Note that I have added a dummy result set as the first level node. This result set returns a row with all *NULL* values except the Tag. Note that I did not specify an attribute name with the element (*CustomersByRegion!1*). This creates an element without any attribute.

Conclusions

In this article, I had tried to explain the usage of *EXPLICIT* directive along with *FOR XML*. Most of the XML formatting requirements can be done with keywords *AUTO*, *RAW* and *PATH*. Using those keywords are very simple. You would need *EXPLICIT* only when a given requirement cannot be fulfilled by those directives.
