http://www.sqlservercentral.com/articles/XML/62289/
Printed 2008/03/22 03:06PM

XML Workshop XVI - Shaping the XML results

By Jacob Sebastian, 2008/02/18

Introduction

In the previous sessions of XML Workshop, we have seen several examples of generating XML results using FOR XML along with RAW, AUTO, PATH and EXPLICIT modes. In the previous sessions, we have learned how to control the structure of XML being generated. This session presents one more example which shows shaping the query results to a certain pre-defined XML structure.

Tables and Data

Let us have a look at the tables and data needed for this example. Here is the script to generate the tables and insert the data needed for this session.

```
CREATE TABLE Departments (DeptID INT, DeptName VARCHAR(20))
GO
INSERT INTO Departments (DeptID, DeptName)
SELECT 1, 'Software' UNION ALL
SELECT 2, 'Administration'

CREATE TABLE Employees(EmpID INT, EmpName VARCHAR(20), DeptID INT)
GO
INSERT INTO Employees (EmpID, EmpName, DeptID)
SELECT 1, 'Jacob', 1 UNION ALL
SELECT 2, 'Steve', 1 UNION ALL
SELECT 3, 'Bob', 2 UNION ALL
SELECT 4, 'Tom', 2
```

Our task is to generate the following XML from the above tables/data.

```
<Departments>
  <Department DepartmentID="1" DepartmentName="Software">
    <Employees>
      <Employee EmployeeID="1" EmployeeName="Jacob" />
      <Employee EmployeeID="2" EmployeeName="Steve" />
    </Employees>
  </Department>
  <Department DepartmentID="2" DepartmentName="Administration">
    <Employees>
      <Employee EmployeeID="3" EmployeeName="Bob" />
      <Employee EmployeeID="4" EmployeeName="Tom" />
    </Employees>
  </Department>
</Departments>
```

Generating the XML

The XML structure is a little more complex than we might think at first glance. The problem is the "Employees" element right after each department. If it were not there, it would have been easy with

FOR XML AUTO as given below.

```
SELECT
    Department.DeptID AS DepartmentID,
    Department.DeptName AS DepartmentName,
    Employee.EmpID AS EmployeeID,
    Employee.EmpName AS EmployeeName
FROM Departments Department
INNER JOIN Employees Employee ON Department.DeptID = Employee.DeptID
FOR XML AUTO, ROOT('Departments')
```

This will produce the following output:

```
<Departments>
    <Department DepartmentID="1" DepartmentName="Software">
        <Employee EmployeeID="1" EmployeeName="Jacob" />
        <Employee EmployeeID="2" EmployeeName="Steve" />
    </Department>
    <Department DepartmentID="2" DepartmentName="Administration">
        <Employee EmployeeID="3" EmployeeName="Bob" />
        <Employee EmployeeID="4" EmployeeName="Tom" />
    </Department>
</Departments>
```

We could see that this is not the XML result that we needed. We need to put the employee records inside a separate element. The new PATH clause added by SQL Server 2005 is very powerful and can be used for a variety of XML shaping requirements. Let us try to use FOR XML PATH to get the XML structure that we need.

```
SELECT
    d.DeptID  AS '@DepartmentID',
    d.DeptName  AS '@DepartmentName',
    (
        SELECT
            e.EmpID  AS '@EmployeeID',
            e.EmpName   AS '@EmployeeName'
        FROM Employees  e WHERE e.DeptID = d.DeptID
        FOR XML PATH('Employee'), TYPE
    ) AS Employees
FROM Departments d FOR XML PATH('Department'), ROOT('Departments')
```

The outer query generates the <Department> elements. The sub query generates the children of each Department and returns them as an XML node. The TYPE clause is used to return values as XML data type. Here is the result generated by the above query.

```
<Departments>
    <Department DepartmentID="1" DepartmentName="Software">
        <Employees>
            <Employee EmployeeID="1" EmployeeName="Jacob" />
            <Employee EmployeeID="2" EmployeeName="Steve" />
        </Employees>
    </Department>
    <Department DepartmentID="2" DepartmentName="Administration">
        <Employees>
            <Employee EmployeeID="3" EmployeeName="Bob" />
            <Employee EmployeeID="4" EmployeeName="Tom" />
        </Employees>
    </Department>
</Departments>
```

We could also use FOR XML EXPLICIT to generate the above XML, but it needs much more code than what we did in FOR XML PATH. FOR XML PATH can do most of the formatting requirements previously available only with EXPLICIT. Here is the FOR XML EXPLICIT version of the above code.

```
;WITH CTE AS (
    SELECT
        1 AS Tag,
        NULL AS Parent,
        DeptID AS 'Department!1!DepartmentID',
        DeptName AS 'Department!1!DepartmentName',
        NULL AS 'Employees!2!',
        NULL AS 'Employee!3!EmployeeID',
        NULL AS 'Employee!3!EmployeeName',
        DeptID * 100 AS Sort
    FROM Departments
    UNION ALL
    SELECT
        2 AS Tag,
        1 AS Parent,
        NULL, NULL, NULL, NULL, NULL, DeptID * 100 + 1
    FROM Departments
    UNION ALL
    SELECT
        3 AS Tag,
        2 AS Parent,
        NULL, NULL, NULL,
        EmpID, EmpName, DeptID * 100 + 1 + EmpID
    FROM Employees
)
SELECT
    Tag,
    Parent,
    [Department!1!DepartmentID],
    [Department!1!DepartmentName],
    [Employees!2!],
    [Employee!3!EmployeeID],
    [Employee!3!EmployeeName]
FROM cte
ORDER BY sort
FOR XML EXPLICIT, ROOT('Departments')
```

The "Sort" column is used to position records in the correct location. We need to put the employees of each departments right under their own tags and hence a custom Sort Order is generated. FOR XML EXPLICIT will write data to the output stream in the same order as the query returns. Hence we need to ensure that the data is returned in the correct order. Here is the result of the above query.

```
<Departments>
    <Department DepartmentID="1" DepartmentName="Software">
        <Employees>
            <Employee EmployeeID="1" EmployeeName="Jacob" />
            <Employee EmployeeID="2" EmployeeName="Steve" />
        </Employees>
    </Department>
    <Department DepartmentID="2" DepartmentName="Administration">
        <Employees>
            <Employee EmployeeID="3" EmployeeName="Bob" />
            <Employee EmployeeID="4" EmployeeName="Tom" />
        </Employees>
    </Department>
</Departments>
```

Conclusions

This session presented another XML formatting requirement and explained how to achieve it by using FOR XML PATH and FOR XML EXPLICIT. I guess some of you out there will come up with other ways of generating the above XML structure and will share your ideas in the discussion forum.
