XML Workshop VI - Typed XML and SCHEMA Collection

Introduction

SQL Server 2005 supports two flavors of *XML*, namely *TYPED* and *UNTYPED*. A *TYPED XML* column or variable is bound to an *XML* schema which defines the structure of the *XML* that the variable or column can store. An *UNTYPED XML* variable or column can store any *XML* value. However, a *TYPED XML* variable or column can store only *XML* values with the specific structure defined by the *SCHEMA*.

Just like a TABLE has a schema which defines the columns, their data type, precision etc, the *XML SCHEMA* which is bound to a *TYPED XML* variable/column specifies the structure of the *XML* that it can store. Using *TYPED XML*
will make your code more efficient as SQL Server has detailed knowledge about the structure of your *XML* column/variable.

In this session of the *XML Workshop*, I am trying to present a few examples, which would explain how to create and *XML* schema for a required *XML* structure.

The Problem

For the purpose of this example, let us assume that we need to define an *XML* structure which contains Customer Information for an order processing application. The application receives customer information from other applications within the enterprise. Our application expects the customer data to arrive in a specific *XML* format. We want *SQL Server* to perform the validation while inserting the *XML* data to the table. To facilitate this, we need to create an *XML* schema which specifies the required structure of the *XML* and bind it to the column in the table. When a column or variable is bound to an XML schema, SQL Server will perform validations while inserting or updating data, to make sure that the value matches with the given *XML* schema.

Here is the *XML* structure that our application requires.

```
<Customers>
  <Customer CustomerNumber="A001">
    <Name>
      <FirstName>Jacob</FirstName>
      <MiddleName>V</MiddleName>
      <LastName>Sebastian</LastName>
    </Name>
    <Address>
      <Street>401, Time Square</Street>
      <City>Ahmedabad</City>
      <State>Gujarat</State>
      <Zip>380006</Zip>
    </Address>
    <Contact>
      <Phone>999 999 9999</Phone>
      <Fax>888 888 8888</Fax>
      <Email>jacob@dotnetquest.com</Email>
    </Contact>
  </Customer>
</Customers>
```

Our task is to create an *XML* schema for the above *XML* structure. We will accept the value only if it is as per the above structure. So let us start defining the *XSD* Schema required to validate the above *XML* structure.

Creating the XML Schema

At first glance, an *XSD* Schema might look very confusing. In this session, I am trying to present an approach which starts with a basic schema and enhances it to meet our requirements. So, let us first create a basic schema with minimum code.

```
/*
    Let us first make the basic XML schema for our customer XML structure.
    In the declaration below, we are creating an XML Schema Collection
    with the name "CustomerSchema". The schema defines an element
    named "Customers"
*/

CREATE XML SCHEMA COLLECTION CustomerSchema AS '
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="Customers">
  </element>
</schema>
'
GO
/*
The "schema" element defines the schema. This is the root element of our
schema.
"<element name="customers">" defines the root node of our XML structure.

Now let us create an XML variable which is bound to the "CustomerSchema"
*/

DECLARE @cust AS XML(CustomerSchema)
SET @cust = '
<Customers>
</Customers>
'

/*
When you assign a value to a variable, which is bound to a schema, SQL Server
validates the value being assigned. For example, the following code will
generate
an error, because the element "Customer" is not defined in the schema.
*/

DECLARE @wrong AS XML(CustomerSchema)
SET @wrong = '<Customer></Customer>'

/*
OUTPUT:

Msg 6913, Level 16, State 1, Line 2
XML Validation: Declaration not found for element 'Customer'. Location:
/*:Customer[1]
*/
```

Now we have a minimal schema. Let us start enhancing it. Below the *Customers* element (root) we need 0 or more child elements named *Customer*. Let us write the *SCHEMA* for it. [Code]

```
/*
Let us drop the previous SCHEMA and create the new version.
*/

DROP XML SCHEMA COLLECTION CustomerSchema
GO

/*
    Let us enhance the schema so that the "Customers" element can contain
    0 or more "Customer" elements.
*/
CREATE XML SCHEMA COLLECTION CustomerSchema AS '
<schema xmlns="http://www.w3.org/2001/XMLSchema">
```

```
   14   <element name="Customers">
   15     <complexType>
   16       <sequence>
   17         <element name="Customer" minOccurs="0">
   18         </element>
   19       </sequence>
   20     </complexType>
   21   </element>
   22 </schema>
   23 '
   24 GO
   25
   26 /*
   27 "Customer" element contains other elements and attributes. Hence we marked it
as
   28 a "complexType".
   29
   30 The "sequence" indicator specifies the order in which the child elements should
   31 occur inside the parent element. Ignore "sequence" for the time being. I will
explain
   32 "sequence" in the next example.
   33
   34 "minOccurs=0" specifies that the element "Customer" is optional.
   35
   36 Let us test the schema.
   37 */
   38
   39 DECLARE @cust AS XML(CustomerSchema)
   40 SET @cust = '
   41 <Customers>
   42     <Customer />
   43 </Customers>
   44 '
```

Let us move ahead. Each *Customer* element should contain a mandatory attribute named *CustomerNumber*. Let us enhance the *SCHEMA* to support this. [Code]

```
    1
    2 DROP XML SCHEMA COLLECTION CustomerSchema
    3 GO
    4
    5 /*
    6     Each "Customer" element should have an attribute named "CustomerNumber".
    7     Let us enhance the schema again.
    8 */
    9 CREATE XML SCHEMA COLLECTION CustomerSchema AS '
   10 <schema xmlns="http://www.w3.org/2001/XMLSchema">
   11   <element name="Customers">
   12     <complexType>
   13       <sequence>
   14         <element name="Customer" minOccurs="0">
   15           <complexType>
   16             <attribute name="CustomerNumber" type="string" use="required" />
   17           </complexType>
   18         </element>
   19       </sequence>
   20     </complexType>
   21   </element>
   22 </schema>
   23 '
   24 GO
   25
   26 /*
   27 use="required" specifies that the attribute "CustomerNumber" is mandatory. The
following code
   28 will generate a compile time error
   29
   30 DECLARE @cust AS XML(CustomerSchema)
   31 SET @cust = '
   32 <Customers>
```

```
    <Customer />
</Customers>
'

Correct XML value is given below.
*/

DECLARE @cust AS XML(CustomerSchema)
SET @cust = '
<Customers>
    <Customer CustomerNumber="A001"/>
</Customers>
'
```

At the next step, we will add the *SCHEMA* for the 3 child elements under the *Customer* element: *Name*, *Address* and *Contact*. [Code]

```

DROP XML SCHEMA COLLECTION CustomerSchema
GO

/*
    Each "Customer" element should have "Name", "Address" and "Contact"
    nodes.
*/
CREATE XML SCHEMA COLLECTION CustomerSchema AS '
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="Customers">
    <complexType>
      <sequence>
        <element name="Customer" minOccurs="0">
          <complexType>
            <sequence>
              <element name="Name" minOccurs="1" maxOccurs="1" />
              <element name="Address" minOccurs="1" maxOccurs="1" />
              <element name="Contact" minOccurs="1" maxOccurs="1" />
            </sequence>
            <attribute name="CustomerNumber" type="string" use="required" />
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>
</schema>
'
GO

/*
Note that I have set "minOccurs" and "maxOccurs" to 1 which specifies that
each element should be present in the XML data EXACTLY once.

We have 3 child elements under the "Customer" element. Note the usage of
"sequence". "sequence" specifies that the elements should occur exactly in the
same
order. The following example will generate an error, because the "Address"
element
is placed after the "Contact" element
*/

DECLARE @cust AS XML(CustomerSchema)
SET @cust = '
<Customers>
    <Customer CustomerNumber="A001">
        <Name />
        <Contact />
        <Address />
    </Customer>
</Customers>
'

```

```
/*
OUTPUT:

Msg 6965, Level 16, State 1, Line 13
XML Validation: Invalid content. Expected element(s):Address where element
'Contact' was specified. Location: /:Customers[1]/:Customer[1]/:Contact[1]

Here is the correct structure. Note that the XML value is EXACTLY in the same
order as defined
in the SCHEMA.
*/

DECLARE @cust AS XML(CustomerSchema)
SET @cust = '
<Customers>
    <Customer CustomerNumber="A001">
        <Name />
        <Address />
        <Contact />
    </Customer>
</Customers>
'
```

Let us move to the final step and complete the *schema*. [Code]

```

DROP XML SCHEMA COLLECTION CustomerSchema
GO

/*
    Now let us enhance the schema further and add all the
    sub elements that we need under "Contact", "Name" and "Address"
*/
CREATE XML SCHEMA COLLECTION CustomerSchema AS '
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="Customers">
    <complexType>
      <sequence>
        <element name="Customer" minOccurs="0">
          <complexType>
            <sequence>
              <element name="Name" minOccurs="1" maxOccurs="1" >
                <complexType>
                  <sequence>
                    <element name="FirstName" type="string" />
                    <element name="MiddleName" type="string" />
                    <element name="LastName" type="string" />
                  </sequence>
                </complexType>
              </element>
              <element name="Address" minOccurs="1" maxOccurs="1" >
              <complexType>
                <sequence>
                  <element name="Street" type="string" />
                  <element name="City" type="string" />
                  <element name="State" type="string" />
                  <element name="Zip" type="string" />
                </sequence>
              </complexType>
            </element>
            <element name="Contact" minOccurs="1" maxOccurs="1" >
              <complexType>
                <sequence>
                  <element name="Phone" type="string" />
                  <element name="Fax" type="string" />
                  <element name="Email" type="string" />
                </sequence>
              </complexType>
            </element>
            </sequence>
            <attribute name="CustomerNumber" type="string" use="required" />
```

```
47            </complexType>
48          </element>
49        </sequence>
50      </complexType>
51    </element>
52 </schema>
53 '
54 GO
55
56 /*
57 Let us test our SCHEMA and see if the SCHEMA validator accepts our value.
58 */
59
60 DECLARE @cust AS XML(CustomerSchema)
61 SET @cust = '
62 <Customers>
63   <Customer CustomerNumber="A001">
64     <Name>
65       <FirstName>Jacob</FirstName>
66       <MiddleName>V</MiddleName>
67       <LastName>Sebastian</LastName>
68     </Name>
69     <Address>
70       <Street>401, Time Square</Street>
71       <City>Ahmedabad</City>
72       <State>Gujarat</State>
73       <Zip>380006</Zip>
74     </Address>
75     <Contact>
76       <Phone>999 999 9999</Phone>
77       <Fax>888 888 8888</Fax>
78       <Email>jacob@dotnetquest.com</Email>
79     </Contact>
80   </Customer>
81 </Customers>
82 '
83
84 /*
85 CHEERS! The schema is ready!
86 */
```

Well, we are done. We worked so hard. It is time to go for a coffee. When you are back, you can download the *SCHEMA* that we just created here.

Conclusions

This article does not provide a full view of *XML* schemas. It demonstrates the basic usage scenarios. The primary purpose of this article is to introduce the basics of *SCHEMAS* and make it familiar to the developers around who want to start working with *TYPED XML*.