عنوان: **استفاده از XQuery - قسمت اول**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۲۵ ۲۲:۴۲
آدرس: www.dotnettips.info
گروه‌ها: NoSQL, SQL Server, xml

XQuery زبانی است که در ترکیب با T-SQL، جهت کار با نوع داده‌ای XML در SQL Server مورد استفاده قرار می‌گیرد. XQuery یک زبان declarative است. عموما زبان‌های برنامه نویسی یا declarative هست و یا imperative. در زبان‌های imperative مانند سی‌شارپ، در هر بار، یک سطر به پردازشگر برای توضیح اعمالی که باید انجام شوند، معرفی خواهد شد. در زبان‌های declarative، توسط زبانی سطح بالا، به پردازشگر عنوان می‌کنیم که قرار است جواب چه چیزی باشد. در این حالت پردازشگر سعی می‌کند تا بهینه‌ترین روش را برای یافتن پاسخ بیابد. SQL و XQuery، هر دو جزو زبان‌های declarative هستند.
XQuery پیاده سازی شده در SQL Server با استانداردهای XQuery 1.0 و XPath 2.0 سازگار است. XQuery برای کار با نودهای مختلف یک سند XML، از XPath استفاده می‌کند. همچنین باید دقت داشت که این زبان به بزرگی و کوچکی حروف حساس است. در آن تمام واژه‌های کلیدی lowercase هستند و تمام متغیرها با علامت $ شروع می‌شوند.

ورودی و خروجی در XQuery

استاندارد XQuery از یک سری توابع ورودی مانند doc برای کار با یک سند و collection برای پردازش چندین سند کمک می‌گیرد. SQL Server از هیچکدام از این توابع پشتیبانی نمی‌کند. در اینجا از XQuery، به کمک متدهای نوع داده‌ای XML استفاده خواهد شد. این متدها شامل موارد ذیل هستند:
- query : یک xml را به عنوان ورودی گرفته و نهایتا یک خروجی XML دیگر را بر می‌گرداند.
- exist : خروجی bit دارد؛ true یا false.
- value : یک خروجی SQL Type را ارائه می‌دهد.
- nodes : خروجی جدولی دارد.
- modify : برای تغییر اطلاعات بکار می‌رود.

این موارد را در طی مثال‌هایی بررسی خواهیم کرد. بنابراین در ادامه نیاز است یک سند XML را که در طی مثال‌های این قسمت مورد استفاده قرار خواهد گرفت، به شرح ذیل مدنظر داشته باشیم:

```
DECLARE @data XML

SET @data =
'<people>
 <person>
  <name>
<givenName>name1</givenName>
<familyName>lname1</familyName>
  </name>
  <age>33</age>
  <height>short</height>
 </person>
 <person>
  <name>
<givenName>name2</givenName>
<familyName>lname2</familyName>
  </name>
  <age>40</age>
  <height>short</height>
 </person>
 <person>
  <name>
<givenName>name3</givenName>
<familyName>lname3</familyName>
  </name>
  <age>30</age>
  <height>medium</height>
 </person>
</people>'
```

در اینجا people در ریشه سند قرار گرفته و سپس سه شخص به مجموعه نودهای آن اضافه شده‌اند.

همانطور که در قسمت قبل نیز ذکر شد، اگر اطلاعات شما در یک فایل XML قرار دارند، نحوه‌ی خواندن آن به شکل یک فیلد XML با کمک openrowset مطابق دستورات زیر خواهد بود:

```
declare @data xml
set @data = (select * from openrowset(bulk 'c:\path\data.xml', single_blob) as x)
```

بررسی متد query

متد query یک XQuery متنی را دریافت کرده، آن‌را بر روی XML ورودی اجرا نموده و سپس یک خروجی XML دیگر را ارائه خواهد داد.
اگر به کتاب‌های استاندارد XQuery مراجعه کنید، به یک چنین کوئری‌هایی خواهید رسید:

```
for $p in doc("data.xml")/people/person
 where $p/age > 30
 return $p/name/givenName/text()
```

همانطور که عنوان شد، متد doc در SQL Server پیاده سازی نشده‌است. بجای آن حداقل از دو روشی که برای مقدار دهی متغیر data عنوان شد، می‌توان استفاده کرد. پس از آن معادل کوئری فوق در SQL Server به نحو ذیل توسط متد query نوشته می‌شود:

```
SELECT @data.query('
 for $p in /people/person
 where $p/age > 30
 return $p/name/givenName/text()
 ')
```

این کوئری givenName تمام اشخاص بالای 30 سال را از سند XML مطرح شده در ابتدای بحث، استخراج می‌کند. خروجی آن نیز یک XML است و اگر آن‌را در SQL Server managment studio اجرا کنید، یک خط آبی زیر نتیجه‌ی آن کشیده می‌شود که بیانگر لینکی است، به محتوای XML حاصل.

بررسی متد value

در ادامه متد value را بررسی خواهیم کرد. در اینجا قصد داریم مقدار سن اولین شخص را نمایش دهیم:

```
SELECT @data.value('/people/person/age', 'int')
```

پارامتر اول متد value یک XQuery است و پارامتر دوم آن، نوع داده‌ای که قرار است بازگشت داده شود. در اینجا اگر اطلاعاتی یافت نشود، نال بازگشت داده خواهد شد.
اگر کوئری فوق را اجرا کنیم با خطای ذیل مواجه خواهیم شد:

```
XQuery [value()]: 'value()' requires a singleton (or empty sequence), found operand of type
'xdt:untypedAtomic *'
```

در اینجا چون از XML Schema استفاده نشده، به untyped Atomic اشاره شده‌است و * پس از آن به zero to many اشاره دارد که برخلاف خروجی zero to one متد value است. این متد، صفر یا حداکثر یک مقدار را باید بازگشت دهد.
برای رفع این مشکل و اشاره به اولین شخص، می‌توان از روش ذیل استفاده کرد:

```
SELECT @data.value('(/people/person/age)[1]', 'int')
```

تولید schema برای سند XML بحث جاری

با استفاده از برنامه Infer.exe مایکروسافت به سادگی می‌توان برای یک سند XML، فایل Schema ایجاد کرد. این برنامه را از اینجا می‌توانید دریافت کنید. پس از آن، اگر فرض کنیم اطلاعات سند XML مثال فوق در فایلی به نام people.xml ذخیره شده‌است، می‌توان schema آن‌را توسط دستور ذیل تولید کرد:

```
Infer.exe people.xml -o schema.xsd
```

people.xml و people.xsd

که نهایتا چنین شکلی را خواهد داشت:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="people">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="person">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="name">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="givenName" type="xs:string" />
                    <xs:element name="familyName" type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="age" type="xs:unsignedByte" />
              <xs:element name="height" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

البته این فایل تولید شده به صورت خودکار، نوع age را unsignedByte تشخیص داده است که در صورت نیاز می‌توان آن‌را به int تبدیل کرد. ولی در کل خروجی آن بسیار با کیفیت و نزدیک به واقعیت است.
این خروجی را که اکنون به صورت یک فایل xsd، در کنار فایل xml معرفی شده به آن می‌توان یافت، با استفاده از openrowset قابل بارگذاری است:

```
declare @schema xml
set @schema = (select * from openrowset(bulk 'c:\path\schema_1.xsd', single_blob) as x)
```

و یا حتی می‌توان یک متغیر از نوع XML را تعریف و سپس محتوای آن را به صورت رشته‌ای در همانجا مقدار دهی کرد.
سپس از این متغیر برای تعریف یک اسکیما کالکشن جدید استفاده خواهیم کرد:

```
CREATE XML SCHEMA COLLECTION poeple_xsd AS @schema
```

در ادامه می‌توان متغیر data را که جهت مقدار دهی سند XML در ابتدای بحث تعریف کردیم، به صورت strongly typed تعریف کنیم:

```
DECLARE @data XML(poeple_xsd)
SET @data = 'مانند قبل با همان محتوایی که در ابتدای بحث عنوان شد'
```

اینبار اگر کوئری ذیل را برای یافتن سن اولین شخص اجرا کنیم:

```
SELECT @data.value('/people/person[1]/age', 'int')
```

خطای واضح‌تری را دریافت خواهیم کرد:

```
XQuery [value()]: 'value()' requires a singleton (or empty sequence), found operand of type
'xs:unsignedByte *'
```

در اینجا xs:unsignedByte بجای xdt:untypedAtomic پیشین گزارش شده‌است.
مشکل کوئری نوشته در اینجا این است که زمانیکه نوع XML تعریف می‌شود، پیش فرض آن content است. یعنی در این حالت چندین root elemnt مجاز هستند. بنابراین person 1 درخواستی، می‌تواند چندین خروجی داشته باشد که در متد value مجاز نیست. این متد، پیش از اجرای کوئری، توسط parser تعیین اعتبار می‌شود و الزاما نیازی نیست تا حتما اجرا شده و سپس مشخص شود که چندین خروجی حاصل آن است.
اینبار تنها کاری که باید برای رفع مشکل گزارش شده انجام شود، تغییر content پیش فرض به document است:

```
DECLARE @data XML(document poeple_xsd)
```

تغییر دیگری نیاز نیست. حتی نیاز نیست از پرانتزها برای مشخص کردن اولین age استفاده کنیم. چون به کمک schema دقیقا مشخص شده‌است که این سند، چه ساختاری دارد و همانند مثال ابتدای بحث، دیگر یک xml untyped نیست.

sequences در XQuery

Sequences بسیار شبیه به آرایه‌ای از آیتم‌ها هستند و منظور مجموعه‌ای از نودها یا مقادیر آن‌ها است. برای مثال به ورودی کوئری‌های XQuery به شکل توالی از یک سند و به خروجی آن‌ها همانند توالی صفر تا چند نود نگاه کنید.

```
DECLARE @x XML
SET @x=''
SELECT @x.query(
'
1,2
(: 1,2 :)
')
```

در مثال فوق یک توالی اصطلاحا دو atomic value را ایجاد کرده‌ایم. این آیتم‌ها با کاما از یکدیگر جدا می‌شوند. همچنین x، پیش از بکارگیری مقدار دهی شده‌است تا null نباشد. عبارتی که بین (: :) قرار می‌گیرد، یک کامنت تفسیر خواهد شد.

همچنین باید دقت داشت که این توالی خطی تفسیر می‌شود.

```
DECLARE @x XML
SET @x=''
SELECT @x.query(
'
for $x in (1,2,3)
for $y in (4,5)
return ($x,$y)
')
```

در اینجا یک جوین کارتزین نوشته شده است، که در آن یک x با یک y جوین خواهد شد. شاید تصور کنید که خروجی آن مجموعه‌ای است با سه عضو که هر عضو آن با دو عضو دیگر جوین می‌شود. اما اگر کوئری فوق را اجرا کنید، یک خروجی خطی را مشاهده خواهید کرد.

به علاوه در SQL Server امکان تعریف Heterogeneous sequences وجود ندارد؛ به عبارتی توالی بین مقادیر و نودها مجاز نیست.
برای مثال اگر کوئری زیر را اجرا کنید:

```
DECLARE @x XML
SET @x=''
```

```
SELECT @x.query(
'
1, <node/>
')
```

با خطای ذیل مواجه خواهید شد:

```
XQuery [query()]: Heterogeneous sequences are not allowed: found 'xs:integer' and
'element(node,xdt:untyped)'
```