عنوان: چرا XML و چرا پشتیبانی توکار از آن در SQL Server
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۲۴ ۱۴:۲۵
آدرس: www.dotnettips.info
گروه‌ها: NoSQL, SQL Server, xml

مقدمه

فیلدهای XML از سال 2005 به امکانات توکار SQL Server اضافه شده‌اند و بسیاری از مزایای دنیای NoSQL را درون SQL Server رابطه‌ای مهیا می‌سازند. برای مثال با تعریف یک فیلد به صورت XML، می‌توان از هر ردیف به ردیفی دیگر، اطلاعات متفاوتی را ذخیره کرد؛ به این ترتیب امکان کار با یک فیلد که می‌تواند اطلاعات یک شیء را قبول کند و در حقیقت امکان تعریف اسکیمای پویا و متغیر را در کنار امکانات یک بانک اطلاعاتی رابطه‌ای که از اسکیمای ثابت پشتیبانی می‌کند، میسر می‌شود.
همچنین SQL Server در این حالت قابلیتی را ارائه می‌دهد که در بسیاری از بانک‌های اطلاعاتی NoSQL میسر نیست. در اینجا در صورت نیاز و لزوم می‌توان اسکیمای کاملا مشخصی را به یک فیلد XML نیز انتساب داد؛ هر چند این مورد اختیاری است و می‌توان یک un typed XML را نیز بکار برد. به علاوه امکانات کوئری گرفتن توکار از این اطلاعات را به کمک XPath ترکیب شده با T-SQL، نیز فراموش نکنید.
بنابراین اگر یکی از اهداف اصلی گرایش شما به سمت دنیای NoSQL، استفاده از امکان تعریف اطلاعاتی با اسکیمای متغیر و پویا است، فیلدهای نوع XML اس کیوال سرور را مدنظر داشته باشید.
یک مثال عملی: فناوری Azure Dev Fabric's Table Storage (نسخه Developer ویندوز Azure که روی ویندوزهای معمولی اجرا می‌شود؛ یک شبیه ساز خانگی) به کمک SQL Server و فیلدهای XML آن طراحی شده است.

چرا XML و چرا پشتیبانی توکار از آن در SQL Server

یک سند XML معمولا بیشتر از یک قطعه داده را در خود نگهداری می‌کند و نوع داده‌ی پیچیده محسوب می‌شود؛ برخلاف داده‌هایی مانند int یا varchar که نوع‌هایی ساده بوده و تنها یک قطعه از اطلاعات خاصی را در خود نگهداری می‌کنند. بنابراین شاید این سؤال مطرح شود که چرا از این نوع داده پیچیده در SQL Server پشتیبانی شده‌است؟
- از سال‌های نسبتا دور، از XML برای انتقال داده‌ها بین سیستم‌ها و سکوهای کاری مختلف استفاده شده‌است.
- استفاده‌ی گسترده‌ای در برنامه‌های تجاری دارد.
- بسیاری از فناوری‌های موجود از آن پشتیبانی می‌کنند.
برای مثال اگر با فناوری‌های مایکروسافتی کار کرده باشید، به طور قطع حداقل در یک یا چند قسمت از آن‌ها، مستقیما از XML استفاده شده‌است.
بنابراین با توجه به اهمیت و گستردگی استفاده از آن، بهتر است پشتیبانی توکاری نیز از آن داخل موتور یک بانک اطلاعاتی، پیاده سازی شده باشد. این مساله سهولت تهیه پشتیبان‌های خودکار، بازیابی آن‌ها و امنیت یکپارچه با SQL Server را به همراه خواهد داشت؛ به همراه تمام زیرساخت‌های مهیای در SQL Server.

روش‌های مختلف ذخیره سازی XML در بانک‌های اطلاعاتی رابطه‌ای

الف) ذخیره سازی متنی

این روش نیاز به نگارش خاصی از SQL Server یا بانک اطلاعاتی الزاما خاصی نداشته و با تمام بانک‌های اطلاعاتی رابطه‌ای سازگار است؛ مثلا از فیلدهای varchar برای ذخیره سازی آن استفاده شود. مشکلی که این روش به همراه خواهد داشت، از دست دادن ارزش یک سند XML و برخورد متنی با آن است. زیرا در این حالت برای تعیین اعتبار آن یا کوئری گرفتن از آن‌ها نیاز است اطلاعات را از بانک اطلاعاتی خارج کرده و در لایه‌ای دیگر از برنامه، کار جستجو پردازش آن‌ها را انجام داد.

ب) تجزیه XML به چندین جدول رابطه‌ای

برای مثال یک سند XML را درنظر بگیرید که دارای اطلاعات شخص و خریدهای او است. می‌توان این سند را به چندین فیلد در چندین جدول مختلف رابطه‌ای تجزیه کرد و سپس با روش‌های متداول کار با بانک‌های اطلاعاتی رابطه‌ای از آن‌ها استفاده نمود.

ج) ذخیره سازی آن‌ها توسط فیلدهای خاص XML
در این حالت با استفاده از فیلدهای ویژه XML می‌توان از فناوری‌های مرتبط با XML تمام و کمال استفاده کرد. برای مثال تهیه کوئری‌های پیچیده داخل همان بانک اطلاعاتی بدون نیاز به تجزیه سند به چندین جدول و یا خارج کردن آن‌ها از بانک اطلاعاتی و جستجوی بر روی آن‌ها در لایه‌ای دیگر از برنامه.

موارد کاربرد XML در SQL Server

کاربردهای مناسب

- اطلاعات، سلسله مراتبی و تو در تو هستند. XQuery و XPath در این موارد بسیار خوب عمل می‌کند.
- ساختار قسمتی از اطلاعات ثابت است و قسمتی از آن خیر. برای نمونه، یک برنامه‌ی فرم ساز را درنظر بگیرید که هر فرم آن هر چند دارای یک سری خواص ثابت مانند نام، گروه و امثال آن است، اما هر کدام دارای فیلدهای تشکیل دهنده متفاوتی نیز می‌باشد. به این ترتیب با استفاده از یک فیلد XML، دیگری نیازی به نگران بودن در مورد نحوه مدیریت اسکیمای متغیر مورد نیاز، نخواهد بود.
نمونه‌ی دیگر آن ذخیره سازی خواص متغیر اشیاء است. هر شیء دارای یک سری خواص ثابت است اما خواص توصیف کننده‌ی آن‌ها از هر رکورد به رکوردی دیگر متفاوت است.

کاربردهای نامناسب

- کل اطلاعات را داخل فیلد XML قرار دادن. هدف از فیلدهای XML قرار دادن یک دیتابیس داخل یک سلول نیست.
- ساختار تعریف شده کاملا مشخص بوده و به این زودی‌ها هم قرار نیست تغییر کند. در این حالت استفاده از قابلیت‌های رابطه‌ای متداول SQL Server مناسب‌تر است.
- قرار دادن اطلاعات باینری بسیار حجیم در سلول‌های XML ایی.

تاریخچه‌ی پشتیبانی از XML در نگارش‌های مختلف SQL Server

الف) SQL Server 2000
در SQL Server 2000 روش (ب) توضیح داده شده در قسمت قبل، پشتیبانی می‌شود. در آن برای تجزیه یک سند XML به معادل رابطه‌ای آن، از تابعی به نام OpenXML استفاده می‌شود و برای تبدیل این اطلاعات به XML از روش for XML ... Select می‌توان کمک گرفت. همچنین تاحدودی مباحث XPath Queries نیز در آن گنجانده شده‌است.

ب) SQL Server 2005
در نگارش 2005 آن، برای اولین بار نوع داده‌ای ویژه XML معرفی گشت به همراه امکان تعریف اسکیمای XML و اعتبارسنجی آن و پشتیبانی از XQuery برای جستجوی سریع بر روی داده‌های XML داخل همان بانک اطلاعاتی، بدون نیاز به استخراج اطلاعات XML و پردازش مجزای آن‌ها در لایه‌ای دیگر از برنامه.

ج) SQL Server 2008 به بعد
در اینجا فاز نگهداری این نوع داده خاص شروع شده و بیشتر شامل یک سری بهبودهای کوچک در کارآیی و نحوه‌ی استفاده از آن‌ها می‌شود.

استفاده از XML با کمک SQLCLR

از SQL Server 2005 به بعد، امکان استفاده از کلیه‌ی امکانات موجود در فضای نام System.Xml دات نت، در SQL Server نیز به کمک SQL CLR مهیا شده‌است. همچنین از SQL Server 2008 به بعد، امکانات فضای نام System.Xml.Linq و مباحث LINQ to XML نیز توسط SQL CLR پشتیبانی می‌شوند.
البته این امکانات در SQL Server 2005 نیز قابل استفاده هستند، اما اسمبلی شما unsafe تلقی می‌شود. پس از آزمایشات و

بررسی کافی، فضای نام مرتبط با LINQ to XML و امکانات آن، به عنوان اسمبلی‌هایی امن و قابل استفاده در SQL Server 2008 به بعد، معرفی شده‌اند.

مزایای وجود فیلد ویژه XML در SQL Server

پس از اینکه فیلدهای XML به صورت یک نوع داده بومی بانک اطلاعاتی SQL Server معرفی شدند، مزایای ذیل بلافاصله در اختیار برنامه نویس‌ها قرار گرفت:
- امکان تعریف آن‌ها به صورت یک ستون جدولی خاصی
- استفاده از آن‌ها به عنوان یک پارامتر رویه‌های ذخیره شده
- امکان تعریف خروجی توابع scalar سفارشی تعریف شده به صورت XML
- امکان تعریف متغیرهای T-SQL از نوع XML

برای مثال در اینجا نحوه‌ی تعریف یک جدول جدید دارای فیلدی از نوع XML را مشاهده می‌کنید:

```
CREATE TABLE xml_tab
(
  id INT,
  xml_col  XML
)
```

- پشتیبانی از فناوری‌های XML ایی مانند اعتبارسنجی اسکیما و نوشتن کوئری‌های پیشرفته با XQuery و XPath.
- امکان تعریف ایندکس‌های XML ایی اضافه شده‌است.

چه نوع XML ایی را می‌توان در فیلدهای XML ذخیره کرد؟

فیلدهای XML امکان ذخیره سازی داده‌های XML خوش فرم را مطابق استاندارد یک XML، دارند. حداکثر اندازه قابل ذخیره سازی در یک فیلد XML دو گیگابایت است.
البته امکانات مهیای در SQL Server در بسیاری از موارد فراتر از استاندارد یک XML هستند. به این معنا که در فیلدهای XML می‌توان Documents و یا Fragments را ذخیره سازی کرد. یک سند XML یا Document حاوی تنها یک ریشه اصلی است؛ اما یک Fragment می‌تواند بیش از یک ریشه اصلی را در خود ذخیره کند. یک مثال:

```
DECLARE @xml_tab TABLE (xml_col XML)
-- document
INSERT @xml_tab VALUES ('<person/>')
-- fragment
INSERT @xml_tab VALUES ('<person/><person/>')
SELECT * FROM @xml_tab
```

مدل داده‌ای XML در SQL Server بر مبنای استانداردهای XQuery و XPath طراحی شده‌است و این استانداردها Fragments را به عنوان یک قطعه داده XML معتبر، قابل پردازش می‌دانند؛ علاوه بر آن مقادیر null و خالی را نیز معتبر می‌دانند. برای مثال عبارات ذیل معتبر هستند:

```
DECLARE @xml_tab TABLE (xml_col XML)
-- text only
INSERT @xml_tab VALUES ('data data data .....')
-- empty string
INSERT @xml_tab VALUES ('')
-- null value
INSERT @xml_tab VALUES (null)
SELECT * FROM @xml_tab
```

همچنین امکان ذخیره سازی یک متن خالی بدون فرمت نیز در اینجا مجاز است. بنابراین به کمک T-SQL می‌توان برای مثال نوع داده varchar و varchar max را به XML تبدیل کرد و برعکس. امکان تبدیل Text و NText (منسوخ شده) نیز به XML وجود دارد ولی

در این حالت خاص، عکس آن، پشتیبانی نمی‌شود.
به علاوه باید دقت داشت که در SQL Server نوع داده‌ای XML برای ذخیره سازی داده‌ها بکار گرفته می‌شود. به این معنا که در اینجا پیشوندهای فضاهای نام XML بی‌معنا هستند.

```
DECLARE @xml_tab TABLE (xml_col XML)
INSERT @xml_tab VALUES ('<doc/>')
INSERT @xml_tab VALUES ('<doc xmlns="http://www.doctors.com"/>')
-- این سه سطر در عمل یکی هستند
INSERT @xml_tab VALUES ('<doc xmlns="http://www.documents.com"/>')
INSERT @xml_tab VALUES ('<dd:doc xmlns:dd="http://www.documents.com"/>')
INSERT @xml_tab VALUES ('<rr:doc xmlns:rr="http://www.documents.com"/>')
SELECT * FROM @xml_tab
```

در این مثال، سه insert آخر در عمل یکی درنظر گرفته می‌شوند.

ذخیره سازی داده‌های XML Encoding

SQL Server امکان ذخیره سازی اطلاعات متنی را به فرمت UFT8، اسکی و غیره، دارد. اما جهت پردازش فیلدهای XML و ذخیره سازی آن‌ها از Collation پیش فرض بانک اطلاعاتی کمک خواهد گرفت. البته ذخیره سازی نهایی آن همیشه با فرمت UCS2 است (یونیکد دو بایتی).

```
DECLARE @xml_tab TABLE  (id INT, xml_col XML)

INSERT INTO @xml_tab
VALUES
  (
5,
N'<?xml version="1.0" encoding="utf-8"?>
<doc1>
  <row name="vahid"></row>
</doc1>
')
```

برای نمونه به مثال فوق دقت کنید. اگر آن‌را اجرا کنید، برنامه با خطای ذیل متوقف خواهد شد:

```
XML parsing: line 1, character 38, unable to switch the encoding
```

علت اینجا است که با قرار دادن N در ابتدای رشته XML ایی در حال ذخیره سازی، آن‌را به صورت یونیکد دوبایتی معرفی کرده‌ایم اما encoding سند در حال ذخیره سازی utf-8 تعریف شده‌است و این‌دو با هم سازگاری ندارند.
برای حل این مشکل باید N ابتدای رشته را حذف کرد. روش دوم، معرفی و استفاده از utf-16 است بجای utf-8 در ویژگی encoding.
همچنین در این حالت اگر encoding را utf-16 معرفی کنیم و ابتدای رشته در حال ذخیره سازی N قرار نگیرد، باز با خطای unable to switch the encoding مواجه خواهیم شد.

نحوه‌ی ذخیره سازی اطلاعات XML ایی در SQL Server

SQL Server فرمت اطلاعات XML وارد شده را حفظ نمی‌کند. برای مثال اگر قطعه کد زیر را اجرا کنید

```
DECLARE @xml_tab TABLE  (id INT, xml_col XML)

INSERT INTO @xml_tab
VALUES
  (
5,
'<?xml version="1.0" encoding="utf-8"?><doc1><row name="vahid"></row></doc1>'
  )

SELECT * FROM @xml_tab
```

خروجی Select انجام شده به صورت زیر است:

```
<doc1>
  <row name="vahid" />
</doc1>
```

اطلاعات و داده نهایی، بدون تغییری از آن قابل استخرج است. اما اصطلاحا lexical integrity آن حفظ نشده و نمی‌شود. بنابراین در اینجا ذکر سطر xml version ضروری نیست و یا برای مثال اگر ویژگی‌ها را توسط " و یا ' مقدار دهی کنید، همیشه توسط " ذخیره خواهد شد.

ذخیره سازی داده‌هایی حاوی کاراکترهای غیرمجاز XML

اطلاعات دنیای واقعی همیشه به همراه اطلاعات تک کلمه‌ای ساده نیست. ممکن است نیاز شود انواع و اقسام حروف و تگ‌ها نیز در این بین به عنوان داده ذخیره شوند. روش حل استاندارد آن بدون نیاز به دستکاری اطلاعات ورودی، استفاده از CDATA است:

```
DECLARE @xml_tab TABLE  (id INT, xml_col XML)

INSERT INTO @xml_tab
VALUES
  (
5,
'<person><![CDATA[ 3 > 2 ]]></person>'
  )

SELECT * FROM @xml_tab
```

در این حالت خروجی select اطلاعات ذخیره شده به صورت زیر خواهد بود:

```
<person> 3 &gt; 2 </person>
```

به صورت خودکار قسمت CDATA پردازش شده و اصطلاحا حروف غیرمجاز XML ایی به صورت خودکار escape شده‌اند.

محدودیت‌های فیلدهای XML

- امکان مقایسه مستقیم را ندارند؛ بجز مقایسه با نال. البته می‌توان XML را تبدیل به مثلا varchar کرد و سپس این داده رشته‌ای را مقایسه نمود. برای مقایسه با null توابع isnull و coalesce نیز قابل بکارگیری هستند.
- order by و group by بر روی این فیلدها پشتیبانی نمی‌شود.
- به عنوان ستون کلید قابل تعریف نیست.
- به صورت منحصربفرد و unique نیز قابل علامتگذاری و تعریف نیست.
- فیلدهای XML نمی‌توانند دارای collate باشند.