XML-implementatie in SQL Server 2005

# XQuery op weg naar standaard

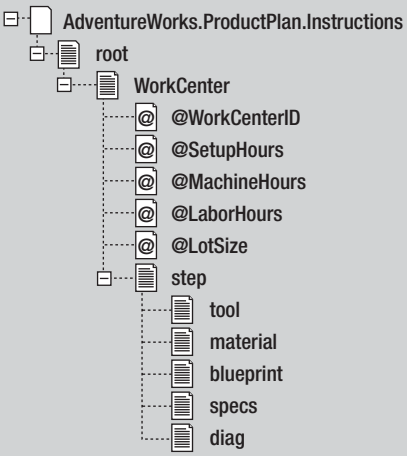
Peter ter Braake

**Ooit was het simpel: XML was voor data-uitwisseling en een database was voor data-opslag. Alleen is met het toenemende gebruik van XML ook een toenemende behoefte ontstaan om XML als XML op te slaan. De vertaalslag van XML naar relationele data en andersom is vaak omslachtig en wellicht onnodig.**

Met XML als native datatype en de integratie van XQuery – de vraagtaal voor XML, zoals SQL dat is voor relationele data – in de query engine, zet Microsoft met SQL Server 2005 (voorheen Yukon) een grote stap richting volledige integratie van XML in de relationele wereld.
Dit artikel is een introductie tot de vraagtaal XQuery en Microsoft's implementatie daarvan in SQL Server 2005. Het World Wide Web Consortium (W3C) werkt aan een standaard voor XQuery. De laatste 'draft' is te vinden op www.w3.org/XML/Query. De ontwikkeling van XQuery loopt parallel met die van XPath 2.0, een W3C-standaard om elementen binnen XML aan te duiden op een manier die te vergelijken is met padaanduidingen van bestanden op de harde schijf.
Omdat XQuery veel gebruik maakt van Xpath-expressies wordt eerst kort ingegaan op Xpath, waarna de integratie van XML en XQuery in de nieuwste versie van SQL Server, die in de eerste helft van 2005 zal uitkomen, aan bod komt. Alle voorbeelden zijn gebaseerd op de eerste beta-versie van SQL Server Yukon.

## XPath

In tegenstelling tot relationele data is XML hiërarchisch opgebouwd. Dat houdt in dat elk element één en hooguit één parent element heeft en er daarmee een eenduidig pad is van de root van het document naar het element onder beschouwing.
Een zogenaamd 'well formed' XML-document heeft precies één root-element. Ontbreekt dit element, maar voldoet het document verder wel aan alle voorwaarden waaraan het moet voldoen om 'well formed' te zijn, dan spreekt men van een XML-fragment. In de rest van dit artikel mag XML-document overal vervangen worden door fragment.
XPath maakt gebruik van de hiërarchische opbouw van XML om collecties van elementen of attributen uit een XML-document te selecteren. Net zoals bij het SQL select statement is het resultaat dus altijd een verzameling. XPath wordt uitgelegd met een aantal voorbeelden die gebaseerd zijn op de Instructions-kolom, te vinden in de ProductPlan-tabel van de AdventureWorks database die bij SQL Server 2005 geleverd wordt. Afbeelding 1 geeft de structuur weer van deze XML-kolom.




**Afbeelding 1:** Structuur van de XML-kolom.

## Informatie

Op www.devx.com/xml/Article/8066 is meer te lezen over XQuery FLWR statements.
Op www.w3schools.com/xpath/default.asp is een korte tutorial en aanvullende voorbeelden over XPath vinden. Tevens vindt u hier informatie over andere aan XML gerelateerde zaken zoals XSLT en XQuery.

Van de onderstaande Xpath-expressies selecteert de eerste het root-element van dit document (en dus het gehele document), terwijl de tweede expressie alle step-elementen uit het document selecteert.

```
/root
/root/WorkCenter/Step
```

De enkele slash (/) refereert aan een absoluut pad binnen het document. De bovenstaande expressie selecteert dus alleen step-elementen die een 'child' zijn van WorkCenter-elementen. Als er binnen het root-element ook step-elementen voorkomen, dan worden die niet geselecteerd. Met de dubbele slash (//) zijn alle elementen te selecteren onafhankelijk van het absolute pad. Zo selecteert de onderstaande expressie alle step-elementen, zowel degene die child zijn van het root-element als degene die child zijn van WorkCenter-elementen.

```
//Step
```

Via het @-teken kunnen attributen worden aangeduid. De onderstaande expressie selecteert het LaborHours-attribuut van alle WorkCenter-elementen. De uitvoer zal dus de verzameling van alle WorkCenter-elementen zijn. Elk element zal een LaborHours-attribuut hebben, maar verder leeg zijn.

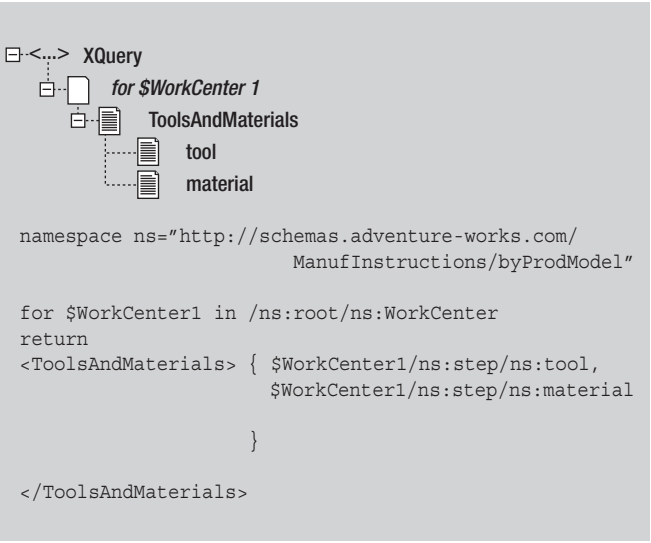
```
/root/WorkCenter/@LaborHours
```

Zoals gesteld leveren Xpath-expressies verzamelingen op. Meestal wordt in dit kader over sets gesproken. De expressies zoals hierboven beschreven leveren alle nodes (elementen) op die voldoen aan het opgegeven pad. In het voorbeeld heeft een WorkCenter-element bijvoorbeeld meer Step-elementen en binnen het document zijn meer WorkCenter-elementen te vinden. Door middel van zogenaamde 'predicates' kan men een filter opgeven, zodat men alleen een subset van elementen terugkrijgt. Dit is te doen door tussen rechte haken een conditie mee te geven, vergelijkbaar met de where clausule uit een SQL statement. Uiteraard moet hierbij rekening worden gehouden met het datatype van het attribuut of het element. Zo levert de eerste van de onderstaande expressies alleen WorkCenter-elementen op waarvoor het attribuut LaborHours een waarde heeft die groter is dan tien. De tweede expressie selecteert het MachineHours-attribuut van alle WorkCenter-elementen waarvoor het attribuut LaborHours een waarde heeft die kleiner is dan vijf.

```
/root/WorkCenter[@LaborHours > 10]
/root/WorkCenter[@LaborHours < 5]/@MachineHours
```

## XQuery

XQuery is gebaseerd op XPath. XQuery maakt gebruik van dezelfde expressies om elementen en attributen binnen een XML-document aan te duiden. Extra is dat XQuery in staat is het resultaat in de vorm van nieuwe XML-elementen terug te geven. Daar waar men met XPath alleen kan werken met elementen uit het oorspronkelijke XML-document, kan men met XQuery nieuwe elementen maken gebaseerd op de oorspronkelijke elementen, door gebruik te maken van het zogenaamde FLWR (flower) statement.




**Afbeelding 2:** XQuery builder, onderdeel van de SQL Server Workbench.

De letters FLWR staan respectievelijk voor de For, Let, Where en Return clausule waaruit het statement is opgebouwd. Met een aantal voorbeelden wordt het FLWR statement uitgelegd. Afbeelding 2 is een weergave van de XQuery builder, een onderdeel van de SQL Server Workbench van SQL Server 2005. De eerste regel van het onderste deel declareert ns als namespace en is eigenlijk niets anders dan een verwijzing naar het XSD-bestand dat de XML beschrijft. Het FLWR statement begint bij de For clausule die vergelijkbaar is met de For Each loop uit Visual Basic. Via het dollar-teken geeft men aan een variabele te maken ($WorkCenter) en één voor één worden alle WorkCenter-elementen hier aan toegekend. Voor elk gevonden element wordt vervolgens het return-gedeelte uitgevoerd. Alles wat na de return tussen accolades staat zal geëvalueerd worden, alles wat buiten de accolades staat wordt letterlijk in het resultaat opgenomen. Het voorbeeld creëert dus nieuwe ToolsAndMaterials-elementen. Elk ToolsAndMaterials-element bevat alle tools- en alle materials-elementen die een child-element zijn van de node waar de variabele $WorkCenter naar verwijst. Net zoals bij een SQL query kan men ook bij een XQuery gebruikmaken van een where clausule. Onderstaande regel is aan bovenstaande XQuery toe te voegen en wel tussen de for clausule en de return clausule in.

```
Where $WorkCenter/@LaborHours > 3
```

Zoals te verwachten is wordt het return-gedeelte met deze toevoeging alleen uitgevoerd voor die WorkCenter-elementen waarvan het attribuut LaborHours een waarde heeft die groter is dan drie. De letter L in het FLWR statement staat voor Let. Door middel van

```
for $klant in document("klanten.xml")//klant
for $order in document("orders.xml")//orders
where $klant/klantnummer = $order/klantnummer
return
   <mijnorder>{
      $order/orderid,
      $klant/klantnaam
      }
   </mijnorder>

for $klant in document("klanten.xml")//klant
return
   <mijnklant>
      {$klant/klantnaam}
      {
      for $order in
                document("orders.xml")//orders
      where $klant/klantnummer =
                           $order/klantnummer
      return {$order/orderid}
      }
   </mijnklant>
```

**Listing 1.**

de let clausule is een variabele te binden aan een sequentie, ofwel aan een serie van elementen die voldoen aan eenzelfde Xpath-expressie. Zo zal de onderstaande XQuery een variabele $stap aanmaken die alle step-elementen bevat.

```
Let $stap := /ns:root/ns:Workcenter/ns:step
```

In tegenstelling tot de for clausule wordt niet één voor één door deze elementen heengelopen, maar $stap bevat ze allemaal tegelijk. Helaas wordt dit gedeelte van XQuery nog niet ondersteund in SQL Server 2005.
Een optie die wel terug te vinden is in SQL Server 2005 beta 1 is wat 'computed constructor syntax' genoemd wordt. In het voorbeeld is een element ToolsAndMaterials gemaakt. Men kan dit element ook attributen meegeven en de attribuutwaarde kan berekend worden door tussen accoladen een Xpath-expressie mee te geven. De onderstaande regel zou bijvoorbeeld een ToolsAndMaterial-element maken met een attribuut uren, dat de waarde krijgt van het LaborHours-attribuut van een WorkCenter-element:

```
<ToolsAndMaterial uren="{$WorkCenter/@LaborHours}">
```

## Joins

Een belangrijk deel van de kracht van SQL komt voort uit de mogelijkheid om joins te kunnen schrijven. Relationele databases zijn (in meer of mindere mate) genormaliseerd, wat dwingt om in query's informatie uit meer tabellen te combineren tot één resultaat. Met XML kan men tegen een vergelijkbare situatie aanlopen die dan om een vergelijkbare oplossing vraagt.
Stel, men heeft een XML-bestand 'klanten.xml' en een apart bestand 'orders.xml', en in beide bestanden vindt men een element klantnummer terug. De eerste XQuery in listing 1 (zie kader) laat een join tussen deze beide bestanden zien zoals men die met SQL ook zou verwachten. Er gebeurt hier een aantal nieuwe dingen. Ten eerste is in de for clausule het document keyword ingebed in een Xpath-expressie. In dit voorbeeld wordt een XML-bestand geopend in plaats van de data uit de database te halen.
Tevens is te zien dat de for clausule twee keer voorkomt waarmee men dus twee XML-bestanden opent. Aan twee verschillende variabelen ($klant en $order) worden elementen toegekend uit de respectieve bestanden. Twee for clausules in een XQuery is vergelijkbaar met twee tabellen in de from clausule van een SQL statement, die slechts door een komma worden gescheiden. Dat houdt in dat er een cartesisch product gevormd wordt: elk klant-element uit het klantenbestand wordt gemengd met elke order uit het orderbestand. De where clausule wordt gebruikt om bij elkaar horende elementen te vinden op basis van klantnummer. Dit is precies zoals men volgens de ANSI SQL 89 standaard-joins moet schrijven in SQL.

De zojuist beschreven join is vergelijkbaar met een inner join. Ook een outer join valt met XQuery te maken hoewel er geen join keyword of iets dergelijks ter beschikking staat. Wat men wel mag doen is XQuery's nesten. De tweede XQuery in listing 1 laat hier een voorbeeld van zien. Voor elke klant die in het klantenbestand gevonden wordt, worden alle orders uit het orderbestand gehaald, waarvoor geldt dat het klantnummer gelijk is aan die van de klant. Elk mijnklant-element in het resultaat van deze Xquery, bevat een klantnaam-element en nul of meer orderid-elementen.

## Integratie in SQL Server

Om te zien hoe het bovenstaande te gebruiken is in een relationele omgeving, en met name hoe Microsoft XQuery heeft geïntegreerd in SQL Server 2005, wordt eerst het XML-datatype bekeken dat in SQL Server 2005 geïntroduceerd gaat worden. XML is in SQL Server 2005 een native datatype. Dit houdt in dat men variabelen en kolommen kan maken die XML opslaan. Microsoft onderkent twee varianten, typed en untyped. Het verschil tussen beide is dat bij typed XML een XSD-schema aan de variabele of kolom wordt gekoppeld. SQL Server zal, alsof het een check constraint betreft, waken dat de data die in de typed kolom gezet wordt, voldoen aan de gestelde definitie. Naast dit voordeel gaat een typed XML-kolom ook efficiënter om met het geheugen, omdat de XML niet als een lange string wordt opgeslagen, terwijl dat met untyped XML wel gebeurt. SQL Server maakt gebruik van de kennis die in de XSD-definitie zit om de data op te slaan als 'gewone' datatypes.

Het XML-datatype kent vier methods waarmee men de opgeslagen XML kan manipuleren. Deze methods zijn query, value, exists en modify en worden gebruikt om de daadwerkelijke integratie tussen SQL en XQuery tot stand te brengen. De query, value en exists method verwachten alle drie een XQuery statement als argument. Men kan een kolom van het type XML selecteren en van deze kolom de query method aanroepen. Als argument wordt een XQuery statement gegeven mee waarmee men kan definiëren welk resultaat men verwacht.
Listing 2 (zie kader) laat twee voorbeelden zien. In het eerste voorbeeld komt de XQuery terug die eerder bij het bespreken van het FLWR statement is gezien. Ditmaal is het XQuery statement echter ingebed in een SQL statement. In dit voorbeeld wordt de instructions-kolom geselecteerd. Het volledige XML-document dat in die kolom te vinden is, is echter niet interesant, een eigen ToolsAndMaterials-element moet gemaakt worden. Dit kan door van de instructions-kolom de query method aan te roepen (door middel van twee dubbele punten) en als parameter een volledig XQuery statement mee te geven.

Tevens wordt in de where clausule gebruik gemaakt van de exist method van de XML-kolom. Deze method controleert slechts of het XQuery statement dat als parameter wordt meegegeven resultaat oplevert. Voor elk record waarvoor geldt dat er geen tool-elementen zitten in de instructions-kolom, geeft exist nul (false) terug. Met andere woorden, het resultaat van de query bestaat uit alleen die records waarvoor wel tools beschreven zijn in de Instructions-kolom. Dit zijn zes records van de negen die in de ProductPlan-tabel zitten.
De tweede query haalt een ProductModelID op uit de ProductPlan-tabel en tevens van de bijbehorende instructie-kolom de som van alle LaborHours-attributen. Het gewenste resultaat is in dit geval geen XML maar gewoon een scalar-waarde. Daarom wordt niet de query method gebruikt zoals in het eerste voorbeeld, maar de value method die een enkelvoudige waarde converteert naar een SQL datatype. In het voorbeeld wordt met een XQuery statement de som berekend van alle LaborHours en de value method geeft dat terug als een kolom met datatype numeric.

Tot slot is er nog de modify method. Met deze functionaliteit loopt Microsoft vooruit op de standaard waar de W3C aan werkt.
Via modify kan men insert, update en delete statements uitvoeren binnen een XML-kolom. Wederom wordt via Xpath-expressies aangegeven welke attributen of elementen men wil veranderen. Net zoals bij SQL is de where clausule uitermate belangrijk om aan te geven welke records aangepast moeten worden. Zie voor voorbeelden de Books Online (BOL) van SQL Server.

## Conclusie

SQL Server heeft een grote stap gezet met betrekking tot de integratie van XML en de traditionele relationele database.
In SQL Server 2000 zijn er wel mogelijkheden om met XML te werken, met name via OPENXML en de FOR XML clausule,

```
select instructions::query('
namespace ns="http://schemas.adventure-
      works.com/ManufInstructions/ByProdModel"

for $WorkCenter in /ns:root/ns:WorkCenter
return
   <ToolsAndMaterials> {
      $WorkCenter/ns:step/ns:tool,
      $WorkCenter/ns:step/ns:material
   }
   </ToolsAndMaterials>
')

from ProductPlan
where instructions::exist('
namespace ns="http://schemas.adventure-
      works.com/ManufInstructions/ByProdModel"

/ns:root/ns:WorkCenter/ns:step/ns:tool
') = 1

select productmodelid, instructions::value(
'namespace ns="http://schemas.adventure-
      works.com/ManufInstructions/ByProdModel"
( sum(/ns:root/ns:WorkCenter/@LaborHours) )'
, 'numeric'
)
from ProductPlan
```

**Listing 2.**

maar XML opslaan kan alleen in tekstkolommen. Dat zijn tot 2 Gigabyte grote velden die lastig te manipuleren zijn en die bovendien geen enkele weet hebben van wat XML is. Het alternatief is om de XML te parsen en te vertalen naar relationele data. Dat is inefficiënt als de gegevens weer als XML opgevraagd worden, maar levert wel de mogelijkheid om de kracht van SQL te gebruiken bij het opvragen van de gegevens.
In SQL Server 2005 kan men XML opslaan en via een XSD-schema kan SQL Server controleren of de opgeslagen data voldoen aan de definitie. Bovendien gebruikt SQL Server het XSD-schema om de XML efficiënt op te slaan. Naast het verminderde geheugengebruik zorgt dat ook voor de mogelijkheid van indexen op XML-kolommen. Door de uitbreiding van de query engine met XQuery kan men de opgeslagen XML benaderen en manipuleren zonder grote hoeveelheden code te schrijven. Vooral voor het ontwikkelen van applicaties die XML versturen, zoals web services, kan deze integratie erg handig zijn.

**Peter ter Braake** (pbraake@computrain.nl) is productspecialist SQL Server bij Compu'Train.