XQuery: nieuwe W3C-standaard

Gereedschap voor XML-developers

Sinds release 9.2 is XML-technologie zoals XPath, XSLT en XMLSchema beschikbaar op het Oracle-platform. In Database 10g Release 2 implementeert Oracle een nieuwe W3C-standaard: XQuery. In dit artikel beschrijft Marc Vahsen aan de hand van een aantal voorbeelden wat er mogelijk is met de query-taal XQuery, welke voorzieningen de database hiervoor heeft en wat de relatie is met bestaande technologieën zoals XSLT, XPath en SQL.

XQuery is een query-taal waarmee een XML bron bevraagd kan worden middels een expressie, en waarbij het antwoord (meestal) bestaat uit XML fragmenten. Wie deze definitie leest zal zich wellicht afvragen of XQuery wel iets toevoegt aan de tools die we momenteel al hebben. Het schrijven van een query op een XML-bron kunnen we eigenlijk al (via XPath) en het resultaat transformeren naar een ander, nieuw XML-document is ook mogelijk (via XSLT). Zal XQuery dan wel in staat zijn marktaandeel te veroveren? Alvorens deze vraag te beantwoorden zal ik aan de hand van een aantal voorbeelden duidelijk maken hoe XQuery werkt, later zal ik deze vraag beantwoorden.

XQuery op het Oracle platform

Momenteel is XQuery nog geen standaard (in W3C termen: 'recommendation') maar het is een working draft die vrijwel uitgekristalliseerd is. De verwachting is dat de huidige versie niet sterk meer zal veranderen alvorens het een standaard wordt. Daarom heeft een aantal fabrikanten reeds besloten om al een implementatie te maken. Momenteel zijn er al meer dan veertig implementaties. Zo kun je XQuery expressies momenteel uitvoeren in Saxon8 (Java-platform), in de Oracle-database (sinds versie 10g Release 2) en in een aantal ontwikkeltools, bijvoorbeeld XMLSpy (2005 en 2006) en Stylus. Zodra XQuery de recommendation-status bereikt zullen eventuele wijzigingen door Oracle snel worden doorgevoerd. Naast de XQuery-standaard heeft Oracle een aantal uitbreidingen geïmplementeerd die door de SQL/XML standaard worden voorgeschreven.

In dit artikel zullen acht voorbeelden worden uitgewerkt in drie richtingen:

1. van XML naar XML
2. van XML naar Relationeel
3. van Relationeel naar XML

Van XML (bron) naar XML (doel)

Hiervoor konden we tot nu toe bijvoorbeeld gebruik maken van XSLT. XQuery doet iets soortgelijks: laat op een XML bronbestand een transformatie los die weer XML oplevert. Zoals een SQL statement een vaste structuur heeft:

SELECT ... FROM ... WHERE ... ORDER BY

zo geldt dat ook voor een XQuery-expressie:

FOR ... LET ... WHERE ... ORDER BY ... RETURN ...

Door de uitspraak van de eerste letters FLWOR worden deze XQuery expressies vaak 'flower' expressies genoemd. De For en Let clause kunnen ook in omgekeerde volgorde voorkomen.

Voorbeeld 1. Eenvoudige FLWOR expressie
Het bron bestand office.xml bevat:

```
<offices>
   <office id="1">
      <city>Costa Mesa</city>
   </office>
   <office id="2">
      <city>El Segundo</city>
   </office>
   <office id="3">
      <city>Rafael</city>
   </office>
   <office id="4">
      <city>Rocklin</city>
   </office>
</offices>
```

We voeren hierop de volgende flower expressie uit:

```
let      $doc := doc('offices.xml')
for      $o in $doc/offices/office
where    $o/@id > 2
order by $o/city
return   <location id="{$o/@id}">
           {$o/city/text()}
         </location>
```

Dit geeft als output

```
<location id="4">Rocklin</location>
<location id="3">San Rafael</location>
```

SQL	XQuery
select	return
from	for
where	where
order by	order by
declare	let

Tabel 1.

De overeenkomst met SQL is sprekend (zie tabel 1). In iedere component kun je gebruik maken van een xpath expressie en de for / let keywords mogen (voor de where clause) willekeurig vaak voorkomen.
Het valt op dat de uitkomst van dit eerste statement geen valid XML is maar een zogenaamd XML-fragment. Eigenlijk is XQuery nog diverser van aard: een expressie levert altijd een *sequence* van resultaten op, zoals bijvoorbeeld blijkt uit de volgende eenvoudige XQuery:

```
for $i in 1 to 10 return $i
```

geeft als antwoord

```
1 2 3 4 5 6 7 8 9 10
```

Om van een sequence van XML nodes weer valid XML te maken, kun je XQuery-expressies (willekeurig diep) nesten:

```
<offices>{
    for      $o in doc('offices.xml')/offices/office[@id>2]
    order by $o/city
    return   <location id="{$o/@id}">
               {$o/city/text()}
             </location>
 }</offices>
```

resultaat:

```
<offices>
   <location id="4">Rocklin</location>
   <location id="3">San Rafael</location>
</offices>
```

XQuery in Oracle

Bovenstaand voorbeeld kan uitgevoerd worden op het Java-platform (bijvoorbeeld met Saxon8), of in XML development environments zoals XMLSpy of Stylus. In het volgende voorbeeld kijken we naar de manier waarop een XQuery-expressie in Oracle kan worden geschreven. Hiervoor biedt Oracle drie mogelijkheden:

1. met SQL/XML functie xmlquery():

```
select xmlquery('[xquery expressie]' returning content) from dual;
```

2. in sqlplus met het commando

```
xquery [xquery expressie]
```

3. met SQL/XML functie xmltable():

```
select * from xmltable('[xquery expressie]') from dual;
```

Voorbeeld 2. Een XQuery op XDB repository resource
Je kunt op verschillende manieren de bron-XML binnen een XQuery aanduiden. We kunnen bijvoorbeeld een XML-file in de Oracle XDB Repository plaatsen (zie een artikel hierover in Optimize nr. 5/2004).

Plaats de XML- bron uit voorbeeld 1 in de volgende XDB repository file:

```
/public/xquery/offices.xml
```

XQuery in sqlplus:

```
SQL> select xmlquery(
  2             'for    $o in doc("/public/xquery/offices.xml")//office
  3              where  $o/@id > 2
  4              return <location id="{$o/@id}">
  5                        {$o/city/text()}
  6                      </location>
  7             '
  8             returning content
  9          ) xml
 10  from dual;

XML
--------------------------------------------------------------------------
--
<location id="3">San Rafael</location><location id="4">Rocklin</locati-
on>
```

```
1 row selected.

SQL>
```

Voorbeeld 3.Query op XDB repository folder
Naast het gebruiken van een XDB repository file, kan ook een XDB repository folder worden gebruikt. Oracle doorzoekt dan alle XML files in deze folder en subfolders.
Vervang hiertoe in het bovenstaande voorbeeld doc(...) door collection(...):

```
doc("/public/xquery/offices.xml")//office
```

door

```
collection("/public/xquery")//office
```

Voorbeeld 4. Query direct vanuit sqlplus
De syntax van een XQuery die je direct in sqlplus uitvoert is een stuk gemakkelijker, maar is natuurlijk wel beperkt bruikbaar. Onderstaand statement werkt alleen vanuit een 10g sqlplus client:

```
SQL> xquery
  2  for $o in doc("/public/xquery/offices.xml")//office
  3  return $o
  4  /

Result Sequence
-----------------------------------------------------------
<office id="1"><city>Costa Mesa</city></office>
<office id="2"><city>El Segundo</city></office>
<office id="3"><city>San Rafael</city></office>
<office id="4"><city>Rocklin</city></office>

4 rows selected.

SQL>
```

Wat opvalt is dat het resultaat hier gesplitst wordt over meerdere rijen. Het is dus niet één sequence van vier XML-nodes, maar vier rijen met één XML-node. In een later voorbeeld zullen we zien dat ook xmltable() dit doet.

Het volgende voorbeeld laat zien hoe je een xmltype kolom van een Oracle-tabel kunt gebruiken als XML-bron.

Voorbeeld 5. XQuery op xmltype kolom
Eerst creëren we een tabel en voegen hieraan XML-inhoud toe:

```
create table ACME_OFFICES_XML
( id        number(10,0)    not null
, xmldoc    xmltype         not null
)
/
```

```
SQL> insert into acme_offices_xml (id, xmldoc)
  2  values
  3  ( 100
  4  , xdburitype('/public/xquery/offices.xml').getxml()
  5  )
  6  /

1 row created.
```

De XQuery luidt nu:

```
select x.id
,      xmlquery('for $o in /offices/office
                 where $o/@id > 2
                 return $o/city'
           passing x.xmldoc
           returning content
        ) cities_xml
from   acme_offices_xml x
/

     ID CITIES_XML
------- ------------------------------------------------
    100 <city>San Rafael</city><city>Rocklin</city>

SQL>
```

De xmltype kolom mag door deze constructie als default context worden gebruikt, waardoor in de for clause een direct xpath vanaf de context root gebruikt kan worden: /offices/office.

Voorbeeld 6. Serialiseer XQuery-resultaat in meerdere rijen
De derde variant hoe je in Oracle een XQuery kunt uitvoeren gebruikt de xmltable() SQL/XML-functie:

```
SQL> select t.column_value
  2  from   xmltable(
  3             'for $o in doc("/public/xquery/offices.xml")//office
  4              return $o
  5             '
  6          ) t
  7  ;

COLUMN_VALUE
--------------------------------------------------------------------------------
-
<office id="1"><city>Costa Mesa</city></office>
<office id="2"><city>El Segundo</city></office>
<office id="3"><city>San Rafael</city></office>
<office id="4"><city>Rocklin</city></office>

4 rows selected.

SQL>
```

Door het opsplitsen van het query-resultaat in meerdere rijen, blijkt deze constructie erg handig te zijn bij het verwerken van het resultaat van een XQuery in bijvoorbeeld PL/SQL of Java-code. Dit zien we ook in het volgende voorbeeld, waarin we

Advertentie

kijken naar de mogelijkheid om xml content naar relationele content over te zetten.

Van XML(bron) naar relationeel (doel)

Bij de voorgaande voorbeelden was het resultaat van de XQuery steeds (een sequence van) XML nodes. Daardoor ligt de vergelijking met de mogelijkheden van XSLT voor de hand. We zullen nu zien dat XQuery – in tegenstelling tot XSLT – ook een relationeel resultaat kan produceren uit een xml resource. Dit blijkt erg krachtig te zijn in de (relationele of procedurele) verwerking van xml.

Voorbeeld 7. Van XML naar relationele data met xmltable()

```
SQL> select office.id
  2  ,       office.city
  3  from    xmltable(
  4              'for $o in doc("/public/xquery/offices.xml")//office
  5               return $o
  6              '
  7              columns id     number        path '@id'
  8              ,       city   varchar2(30) path 'city'
  9          ) office
 10  where office.id > 2
 11  ;

     ID CITY
------- ----------------------
      3 San Rafael
      4 Rocklin

2 rows selected.

SQL>
```

Je ziet dat hierbij het resultaat automatisch 'geparsed' is naar relationeel formaat. Om dit voor elkaar te krijgen zonder XQuery moet je in Oracle je toevlucht nemen tot bijvoorbeeld de dbms_xmldom package, en die is door zijn lastige API nogal moeizaam in het gebruik. Om het bovenstaande resultaat te bereiken met de DOM API is een veelvoud van het aantal regels code nodig.

Van Relationeel (bron) naar XML (doel)

De Oracle-server beschikt al een tijdje over SQL/XML-functies om relationele data over te zetten naar xmltype-data (zie Optimize nr. 3/2004). Denk aan de functies xmlelement(), xmlattribute() en xmlsequence().
Met XQuery komt daar een nieuwe mogelijkheid bij, door gebruik te maken van de ora:view() functie in een XQuery-expressie. Deze functie zet een tabel (of view) source automatisch om naar canonical XML-formaat, waardoor kolommen met xpath-expressie /ROW/<column_name> uitgelezen kunnen worden.

Voorbeeld 8. Relationele data naar XML met ora:view()

```
SQL> select xmlquery(
  2             'for $o in ora:view("acme_offices")/ROW
  3              where $o/ID > 2
  4              return $o/NAME
  5             '
  6             returning content
  7          ) xml
  8  from dual
  9  /

XML
------------------------------------------------------------
<NAME>San Rafael</NAME><NAME>Rocklin</NAME>

1 row selected.

SQL>
```

Om verschillende XML-resources aan elkaar te verbinden kun je binnen een XQuery een join schrijven:

```
SQL> select xmlquery(
  2             'for $o           in ora:view("acme_offices")/ROW
  3              for $oList       in ora:view("acme_offices_lists")/ROW
  4              for $oListEntry in ora:view("acme_offices_list_
entries")/ROW
  5              where $o/ID      > 2
  6              and   $o/ID      = $oListEntry/OFFICE_ID
  7              and   $oList/ID = $oListEntry/OFFICES_LIST_ID
  8              return <office name="{$o/NAME}"
  9                             listName="{$oList/DESCRIPTION}"
 10                     />
 11             '
 12             returning content
 13          ) xml
 14  from dual
 15  /

XML
------------------------------------------------------------------
<office name="San Rafael" listName="California offices"></office>
<office name="Rocklin" listName="California offices"></office>

1 row selected.

SQL>
```

Hierboven worden drie ora:view() resources aan elkaar gejoined, maar het zou evengoed een join mogen zijn tussen een XDB repository file, een ora:view() en een xmltype kolom. Gelukkig heeft Oracle de optimalisatie van XQuery-statements goed geregeld: bovenstaande query maakt onder water gebruik van aanwezige indexen en constraints. Het explain plan pad voor deze XQuery is vergelijkbaar met een query waarin de drie tabellen relationeel aan elkaar worden gejoined.

Voorbeeld 9. XML URI als resource voor XQuery
Erg handig is dat je ook een willekeurige URI die XML oplevert kunt gebruiken als basis voor je XQuery. Denk bijvoorbeeld aan een webservice of – zoals in onderstaand voorbeeld – een RSS feed:

```
SQL> select xmlquery(
  2             'for $i in $xml
  3              return $i//item/title
  4             '
  5             passing xmlparse (
  6                         document
  7                         httpuritype('http://www.nu.nl'
  8                            ||'/deeplink_rss2/index.jsp?r=Internet'
  9                         ).getclob()
 10                    ) as "xml"
 11             returning content
 12          ).extract('/')  internetnieuws
 13  from dual;

INTERNETNIEUWS
--------------------------------------------------------------------
<title>Tweederde jongeren helpt ouders op internet</title>
<title>Meldpunt dode vogels op internet</title>
<title>CBP: registratie domeinnaam moet anoniem kunnen</title>
<title>Website smurfen mag niet meer</title>

1 row selected.

SQL>
```

Wil je dit resultaat in meerdere rijen terug zien, gebruik dan de xmltable()-constructie in plaats van xmlquery(). Alvorens we de balans opmaken volgen hieronder nog enkele wetenswaardigheden van XQuery:

Datatypes in XQuery

XQuery is – in tegenstelling tot XSLT – 'strongly typed'. Dit wil zeggen dat je beschikt over een heel scala aan datatypen. Deze typering is gebaseerd op de typering binnen XMLSchema. Voorbeelden van atomaire datatypen zijn xs:integer, xs:double. Ook nodes zijn getypeerd, zo heeft een attribuut het attribute() type, eneen element het element() type.
Door deze typering kun je in XQuery veel gemakkelijker berekeningen uitvoeren dan in XSLT. Ter vergelijking: wat zou je van PL/SQL vinden als je als datatype alleen varchar2 tot je beschikking had?
Soms loop je bij de typering tegen implementatiespecifieke zaken op. In de volgende query verliest Oracle bijvoorbeeld het type van het attribuut id:

```
SQL> select xmlquery(
  2             'for $o in doc("/public/xquery/offices.xml")//office
  3              return <x>{$o/@id}</x>
  4             '
  5             returning content
  6           )  xml
  7  from dual;

XML
--------------------------------------------------------------------
<x>1</x><x>2</x><x>3</x><x>4</x>

1 row selected.

SQL>
```

Terwijl Saxon8 (en XMLSpy) dit wel vasthouden:

```
for $o in doc('offices.xml')/offices/office
return <x>{$o/@id}</x>

<x id="1"/><x id="2"/><x id="3"/><x id="4"/>
```

XQuery functies

XQuery beschikt over meer dan honderd functies die je in je expressies kunt gebruiken. Deze functies varieren van de bekende XPath 1.0 functies zoals count() en starts-with() tot veel uitgebreidere en exotische functies.

Interessant zijn de functies replace() en matches() die we zo hebben moeten missen in XSLT 1.0.

Deze replace() en matches() functies werken op basis van reguliere expressies. In een Oracle XML-query moet je de implementatie-specifieke variant van Oracle gebruiken: ora:replace() en ora:matches. Deze functies zijn overigens sinds database versie 10 release 1 ook beschikbaar in 'plain SQL':

```
SQL> select regexp_replace('abcde', '([ab][cd])+', 'x') from dual;

REGE
----
axde
```

Naast de standaardfuncties kun je in XQuery ook je eigen functies schrijven. Die declareer je direct voorafgaand aan de XQuery-expressie in de zogenaamde XQuery-proloog.

Query is modulair van opzet

Bij het schrijven van een XQuery-expressie kun je tussentijdse resultaten opslaan in een variabele. Door de typering binnen XQuery kan dat net zo gemakkelijk een string als een compleet stuk XML zijn. Een XSLT-variabele kan met de constructie

```
<xsl:variable name="[name]">[value]</variable>
```

alleen maar een string value bevatten.

Conclusie

XQuery zal XSLT, XPath of SQL niet gaan vervangen, maar daar wel een belangrijke aanvulling op vormen. XSLT zal in de toekomst gebruikt blijven worden voor transformaties van XML naar XML of naar XHTML. XSLT heeft een aantal tekortkomingen – processing van source nodes kan alleen in documentvolgorde, de mogelijkheid tot het schrijven van eigen functies is beperkt, de taal is niet getypeerd, maar het concept blijft krachtig uitstekend toepasbaar voor een groot aantal toepassingen.
In de automatiseringswereld, waar steeds meer informatie beschikbaar komt in XML-formaat is echter wel degelijk een plaats voor XQuery. De bovenstaande voorbeelden tonen wel aan dat sommigen hiervan moeilijk (of helemaal niet) met alleen XSLT te realiseren zijn. Toekomstige ontwikkelingen maken die overtuiging alleen maar sterker. Denk bijvoorbeeld aan 'Virtual XML Garden', een initiatief dat IBM momenteel ontwikkelt. Hiermee beschik je over virtuele XML-views (XQuery-functies) over gestructureerde data zoals een zip file. De Oracle implementatie van XQuery lijkt prima bruikbaar te zijn in de praktijk en biedt krachtige nieuwe hulpmiddelen. Daarnaast brengt 10gR2 nog meer goed nieuws, bijvoorbeeld nieuwe SQL/XML-functies insertchildxml() en insertxmlbefore(). Vandaar de oproep aan alle XML-developers: stap op een DBA af en regel de migratie naar RDBMS 10g Release 2!

Marc Vahsen is als Senior Consultant werkzaam bij Cumquat Information Technology en kan bereikt worden via marc@cumquat.nl.

NIEUWS

Oracle breidt supply chain suite uit

Oracle presenteert zijn vernieuwde versie van Oracle Advanced Planning en Scheduling (APS) en integreert hierin Strategic Network Optimization en Production Scheduling van de PeopleSoft Supply Chain Planning-suite. De nieuwe versie combineert de voordelen van zowel de PeopleSoft- als Oracle-applicaties. Hierdoor wordt de steeds complexere leveranciersketen inzichtelijker en worden toeleveranciers en productiebronnen optimaler benut. Bovendien stelt het de gebruikers in staat om de risico's, die verbonden zijn met de leveranciersketen, zoveel mogelijk terug te dringen.

Oracle introduceert Oracle TimesTen In-memory Database 6.0

Oracle presenteert Oracle TimesTen In-Memory Database Release 6.0. Dit is de eerste product-upgrade na de acquisitie van TimesTen in juni van dit jaar. Deze versie biedt enorme verbeteringen qua opslagcapaciteit en beschikbaarheid, een groter databasegeheugen en meer data-opslag, sterkere integratie met Oracle-producten en bredere ondersteuning van standaarden als SQL en Java. Met deze nieuwe versie is Oracle eigenaar van een end-to-end datamanagement oplossing die zowel real-time, embedded en front-office-systemen omvat, als grootschalige enterprise databases en datawarehouses.

Benchmarks tonen recordprestaties Oracle 10g Release 2

Opnieuw heeft Oracle records gebroken in een TPC-H 300 gigabyte (GB) benchmark voor Oracle Database 10g Release 2 en Oracle Real Application Clusters op Red Hat Enterprise Linux. Oracle heeft daarmee zijn eigen benchmark-resultaat van afgelopen september overtroffen. Daarmee wordt de lat voor datawarehousing-performance nog hoger gelegd. Oracle Database 10g Release 2 en Oracle Real Application Clusters draaiden op een acht node HP BladeSystem cluster van ProLiant BL25p server blades, uitgerust met een AMD Opteron 2.6 GHz dual-core processors en Red Hat Enterprise Linux versie 4, en bereikten een wereldrecord-performance van 18,725.9 QphH@300GB en een prijs-performance ratio van $27.97/QphH@300GB.
Sinds de release van de database in januari 2004 heeft Oracle 10g indrukwekkende prestaties geleverd op het gebied van performance en schaalbaarheid.
Ook in een recente TPC-H 10 Terabyte benchmark voor Oracle Database 10g Release 2 werd een record gebroken. Met deze prestatie is Oracle nu recordhouder voor de drie belangrijkste schaal-factoren in databaseperformance. Oracle heeft vier van de vijf TPC-H performance records in handen: TPC-H 300 Gigabyte, TPC-H 1 TB, TPC-H 3 TB en TPC-H 10 TB.
Oracle Database 10g Release 2 met Automatic Storage Management bereikte een performance van 108,099.7/QphH@10000GB met een prijs-performance ratio van $53.80/QphH@10000GB, op een enkele Sun Fire E25K server met 72 UltraSPARC IV+ 1.5 GHz processoren en een Sun Solaris 10 besturingssysteem.