Zoeken in boomstructuren

Een nieuw soort query taal voor het ondervragen en manipuleren van XML gegevens

Hans C. Arents

Naarmate meer en meer gegevens in XML formaat opgeslagen of uitgewisseld worden neemt de behoefte toe aan een bijbehorende query taal die toelaat van deze gegevens op de gepaste manier te ondervragen. XML vormt echter een uitdaging, omdat noch het relationele noch het object-georiënteerde data model hierop van toepassing zijn en er dus een totaal nieuw soort query taal dient ontwikkeld te worden. In dit artikel bespreken we de verschillende pogingen tot nu toe, bekijken we in detail de momenteel meest gebruikte XML query taal XQL, en gaan we ook na in hoeverre XQL reeds door de bestaande XML data servers ondersteund wordt.

Waarom een aparte query taal voor XML?

De grote doorbraak in het gebruik van relationele databanken is er pas gekomen toen het duidelijk werd welke het onderliggende data model was (relationele algebra). Op basis daarvan kon immers de SQL query taal ontworpen worden die toeliet van relationele gegevens op een betrouwbare en volledige manier te ondervragen. Sindsdien is SQL een algemeen aanvaarde en gestandaardiseerde query taal geworden. Gelet op het groeiend succes van XML als data formaat voor de opslag en uitwisseling van gegevens bestaat er momenteel een acute nood aan een query taal die dezelfde rol kan spelen als SQL. Het probleem is echter dat XML op een volledig ander data model gebaseerd is. Een XML document is in essentie een hiërarchische boom van elementen, attributen en inhoud, of nauwkeuriger gezegd een woud van boomstructuren. Wat precies het beste wiskundige formalisme is om dit semi-gestructureerde data model adequaat te beschrijven is nog niet volledig duidelijk (favoriete kandidaat tot nu toe zijn zgn. forest automata). Het is echter wel al duidelijk dat aan een XML document vragen dienen gesteld te kunnen worden die door een (object-)relationele of object-georiënteerde query taal niet eens kúnnen gesteld worden. Vragen naar de wijze waarop bepaalde elementen in andere elementen genest zijn (hiërarchie), naar de wijze waarop bepaalde elementen op elkaar volgen (opeenvolging), naar de inhoud van elementen en attributen, enz. De informatie inhoud van een XML document bestaat immers niet alleen uit de eigenlijke inhoud van dat document, maar ook uit de relaties tussen die inhoud zoals uitgedrukt door de structuur van het document. Vandaar de nood aan een aparte query taal die zowel inhoud als structuur kan aanspreken.

Historiek achter de XQL query taal

Al kort na de aanvaarding van XML als officiële W3C (**W**orld **W**ide **W**eb **C**onsortium) standaard in februari 1998 begon men al een XML ondervragingstaal te ontwerpen. De eerste poging, XML-QL, benaderde de problematiek vanuit een zuiver databank perspectief en had een opvallende SQL “look and feel”. De XML-QL taal had echter een aantal gebreken, als gevolg van een te sterk data-georiënteerde kijk op XML gegevens. Om te vermijden dat er een wirwar aan XML ondervragingstalen zou ontstaan organiseerde het W3C in december 1998 een XML Query Languages Workshop, QL'98, om de violen op elkaar af te stemmen. De bijdrage die daar op de meeste bijval mocht rekenen was het XQL (**X**ML **Q**uery **L**anguage) voorstel, mede-ondersteund door Microsoft. Hoewel ook deze XQL’98 taal een aantal gebreken vertoonde, ditmaal als gevolg van een te sterk document-georiënteerde kijk op XML gegevens, werd deze taal toch door alle XML data server vendors gekozen als XML query taal. Ondertussen is er ook al een voorstel voor een nieuwere versie, XQL’99, die tracht de sterke punten van XML-QL en XQL’98 te combineren. XQL is dus géén officiële W3C standaard, en sinds 1998 is het W3C eigenlijk nog niet veel verder geraakt dan het oprichten van een XML Query Working Group, die pas recentelijk een XML Query Requirements document wist te produceren. Een echte standaard XML Query taal ligt dus zeker nog meer dan één jaar in het verschiet. Opmerkelijk aan dit Requirements document is overigens dat het enkel spreekt over de gewenste query mogelijkheden van een toekomstige XML Query taal, en niets zegt over de gewenste add/update/delete mogelijkheden. M.a.w. het W3C beschouwt een XML Query taal nog louter als een XML data *ondervragings*taal, en nog niet als een volwaardige XML data *manipulatie*taal. Vandaar dat sommige XML data server vendors aan XQL hun eigen, produkt-specifieke add/update/delete extensies toegevoegd hebben.

Wat zijn de kenmerken van XQL?

De syntactische en functionele verschillen tussen XQL en SQL zijn een gevolg van de fundamentele verschillen tussen het soort data (relationele en XML) waarvoor deze beide query talen ontworpen werden. Om dit goed te begrijpen loont het de moeite om te kijken welk antwoord beide soorten data geven op de volgende vijf vragen:

- Wat is een databank eigenlijk?
- Wat is het onderliggende data model?
- Wat is de input voor een query?
- Waarop kan er gezocht worden?
- Wat is het resultaat van een query?

De volgende tabel geeft een overzicht van de antwoorden:

Relationele data	**XML data**
De databank is een verzameling tabellen.	De databank is een verzameling XML documenten.
Het onderliggend data model is gebaseerd op relationele algebra.	Het onderliggend data model is gebaseerd op hiërarchische boomstructuren.
De input voor een query is één of meerdere tabellen.	De input voor een query is één of meerdere XML documenten.
Er kan gezocht worden op inhoud.	Er kan gezocht worden op hiërarchie en opeenvolging (van elementen) en op inhoud (van elementen en attributen).
Het resultaat van een query is een tabel die op zijn beurt mag gebruikt worden voor verdere queries.	Het resultaat van een query is een XML document dat op zijn beurt mag gebruikt worden voor verdere queries.

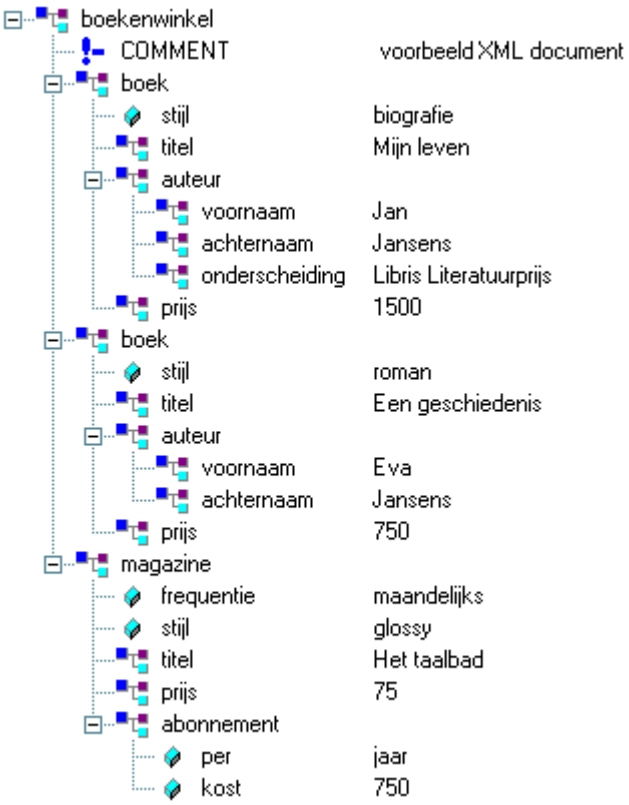
XQL moet dus kunnen toegrijpen op de hiërarchische boomstructuur van een XML document, en zinvolle vragen stellen met betrekking tot hiërarchie, opeenvolging en inhoud (of een combinatie hiervan). Daartoe beschouwt XQL een XML document (zie Figuur 1) als een boom met één enkele wortel knop, en met knoppen voor ieder element, attribuut, tekst string, commentaar, enz. in de XML gegevens (zie Figuur 2). Op die manier is XQL in staat om elementen uit een XML document te selecteren door te testen of die elementen aan bepaalde structuur- of inhoudsvereisten voldoen. De verzameling elementen die als resultaat van een XQL query teruggegeven worden hebben dezelfde hiërarchie en volgorde zoals ze die hadden in het ondervraagde XML document. Van elk element in die verzameling resultaatelementen wordt telkens de volledige inhoud teruggegeven, m.a.w. het element zelf (met zijn attributen en inhoud) en alle subelementen van dat element.

Wat is er mogelijk in XQL?

Net zoals SQL is XQL is een declaratieve taal, geen procedurale taal. U zegt wat u wenst te vinden, en de XQL query engine voert die query zo goed mogelijk voor u uit. Nergens in de XQL standaard wordt vastgelegd hoe XML documenten het best dienen opgeslagen te worden, laat staan hoe ze het best dienen geïndexeerd te worden om optimaal ondervraagd te kunnen worden. Om u een idee te geven van welk soort vragen u kan stellen in XQL geven we u een kort overzicht van de belangrijkste query mogelijkheden.

```
<?xml version="1.0" ?>
- <boekenwinkel>
    <!-- voorbeeld XML document -->
  - <boek stijl="biografie">
      <titel>Mijn leven</titel>
    - <auteur>
        <voornaam>Jan</voornaam>
        <achternaam>Jansens</achternaam>
        <onderscheiding>Libris Literatuurprijs</onderscheiding>
      </auteur>
      <prijs>1500</prijs>
    </boek>
  - <boek stijl="roman">
      <titel>Een geschiedenis</titel>
    - <auteur>
        <voornaam>Eva</voornaam>
        <achternaam>Jansens</achternaam>
      </auteur>
      <prijs>750</prijs>
    </boek>
  - <magazine stijl="glossy" frequentie="maandelijks">
      <titel>Het taalbad</titel>
      <prijs>75</prijs>
      <abonnement kost="750" per="jaar" />
    </magazine>
  </boekenwinkel>
```

Figuur 1: Een voorbeeld XML document.



Figuur 2: De boomstructuur van het voorbeeld XML document.

Lopen doorheen de boomstructuur

Zoals gezegd bekijkt XQL de te ondervragen XML gegevens als een boomstructuur. Het resultaat van een XQL query hangt dan ook af van in welke knop in die boomstructuur de query gesteld wordt, en over welke verzameling knoppen heen u uw query wenst uit te voeren. U dient dus eerst te bepalen vanaf waar in de boom (de zgn. *context knop*) de query dient geëvalueerd te worden, en vervolgens dient u te bepalen of u enkel over de

onmiddellijke kinderen of over alle afstammelingen (op willekeurige diepte) heen wenst te lopen. Dit gebeurt aan de hand van zgn. *context operatoren* (/, //, ., and ..), waarmee u een *pad* doorheen de boom vastlegt.

Voorbeeld: geef alle “prijs” elementen die onmiddellijke kinderen zijn van het element waarin ik me bevind

```
./prijs
```

wat kan afgekort worden tot:

```
prijs
```

Voorbeeld: geef alle “auteur” elementen die een willekeurige afstammeling zijn van de vader van het element waarin ik me bevind

```
..//auteur
```

Voorbeeld: geef alle “voornaam” elementen die onmiddellijke kinderen zijn van het “auteur” element

```
auteur/voornaam
```

Voorbeeld: geef alle “titel” elementen die een willekeurige afstammeling zijn van het “boekenwinkel” element

```
boekenwinkel//titel
```

Vinden van elementen

De *wortel* knop is de bovenste knop van de boomstructuur. Het wortel element is het element dat alle andere elementen in het XML document bevat. Als u wil dat uw query in dit wortel element start dan dient de query te beginnen met een “/”, het symbool voor het wortel element.

Voorbeeld: geef het wortel element van het voorbeeld document

```
/boekenwinkel
```

Voorbeeld: geef alle “auteur” elementen in het voorbeeld document

```
//auteur
```

Opgelet: had u hier

```
/auteur
```

gebruikt, dan had u natuurlijk niks teruggevonden, vermits de “auteur” elementen geen onmiddellijke kinderen zijn van het wortel element.

U kan ook een wildcard gebruiken om bepaalde elementen terug te vinden.

Voorbeeld: geef alle elementen die onmiddellijke kinderen zijn van het “magazine” element

```
magazine/*
```

Vinden van attributen

XQL beschouwt de attributen van een element ook als knoppen van de document boomstructuur, dus kan u dezelfde pad syntax gebruiken als voor elementen om die attributen terug te vinden. Om het onderscheid te maken tussen elementen en attributen dient de naam van een attribuut wel voorafgegaan te worden door @.

Voorbeeld: geef het attribuut “stijl” van het element waarin ik me bevind

```
./@stijl
```

wat kan afgekort worden tot:

```
@stijl
```

Voorbeeld: geef het attribuut “kost” van het “abonnement” element

```
abonnement/@kost
```

U kan eveneens een wildcard gebruiken om bepaalde attributen terug te vinden.

Voorbeeld: geef alle attributen van het “abonnement” element

```
abonnement/@*
```

Filteren van elementen

U kan elementen uit een verzameling elementen wegfilteren door er een *filter* subquery op toe te passen. Een dergelijke filter subquery moet een Booleaanse waarde als resultaat geven, en wordt gebruikt om elk element uit de verzameling elementen te testen. Elk element dat niet voldoet wordt dan weggelaten uit de verzameling.

Voorbeeld: geef alle elementen die het attribuut “stijl” hebben

```
//*[@stijl]
```

Voorbeeld: geef alle “boek” elementen die tenminste één “auteur” element als onmiddellijk kind hebben

```
//boek[auteur]
```

Voorbeeld: geef de titel van alle boeken waarvan de auteur tenminste één onderscheiding behaald heeft

```
//boek[auteur/onderscheiding]/titel
```

In zo’n filter subquery kan ook gekeken worden naar de waarde van een element of een attribuut, of kunnen er Booleaanse operatoren gebruikt worden.

Voorbeeld: geef de titel van alle boeken waarvan de auteur als achternaam Jansens heeft

```
//boek[auteur/achternaam=’Jansens’]/titel
```

Voorbeeld: geef de voornaam van de auteur van alle romans met een prijs lager dan 1000

```
//boek[@stijl=’roman’ and prijs < 1000]/auteur/voornaam/
```

Selecteren van elementen

Het is ook mogelijk een element of een reeks elementen uit een verzameling resultaatelementen te selecteren door deze via hun volgnummers aan te spreken. De elementen in zo’n resultaat worden genummerd vanaf 1, geteld in dezelfde volgorde als waarin ze oorspronkelijk in het XML document voorkomen.

Voorbeeld: geef het eerste boek in de boekenwinkel

```
//boek[1]
```

Voorbeeld: geef de tweede auteur met als achternaam Jansens

```
//auteur[achternaam=’Jansens’][2]
```

Voorbeeld: geef het voorlaatste en het laatste prijs element in het voorbeeld document

```
//prijs[-2 to -1]
```

Zoeken naar tekst

Eigenaardig zijn de zoekmogelijkheden van XQL het minst sterk ontwikkeld als het op zoeken van tekst inhoud in een element of een attribuut van een element aankomt.

Voorbeeld: geef het “titel” element van het boek dat als titel de string “Mijn leven” heeft

```
//boek/titel[.='Mijn leven']
```

Merk op dat de tekst string perfect moet matchen. Pas in XQL’99 wordt de mogelijkheid voorzien om ook “*” en “?” wildcards te mogen gebruiken, en zowel case-insensitive als case-sensitive te zoeken.

Unie en intersectie

De unie operator laat toe van twee verzamelingen elementen samen te nemen, de intersectie operator laat toe van de doorsnede te vinden tussen twee verzamelingen elementen.

Voorbeeld: geef alle boeken en magazines in de boekenwinkel

```
/boekenwinkel/(boek union magazine)
```

Voorbeeld: geef alle boeken waarvan de auteur als achternaam “Jansens” heeft en met een prijs hoger dan 1000

```
//boek[auteur/achternaam="Jansens"] intersect //boek[prijs > 1000]
```

Merk op dat zo’n unie of intersectie enkel mogelijk is binnen éénzelfde XQL query. Het is niet mogelijk om een unie of een intersectie te nemen van de verzamelingen resultaatelementen van twee XQL queries. De mogelijkheid om een join uit te voeren tussen twee verzamelingen resultaatelementen is pas voorzien in XQL’99.

Welke tools ondersteunen nu reeds XQL?

In een vorig artikel hebben we reeds gezien hoe een aantal XML data servers reeds XQL ondersteunen, hoewel het dus nog steeds geen echte standaard is. De nood aan een bruikbare XML query taal is echter zo groot dat o.a. eXcelon Corp. in de *eXcelon* XML data server en Software AG in de *Tamino* XML data server besloten hebben om dan maar (een gedeelte van) XQL’98 te ondersteunen. Tamino beperkt zich strikt tot de ondervraging van XML gegevens, eXcelon ondersteunt ook de add/update/delete van XML gegevens.

Software AG’s *Tamino* ondersteunt slechts een vrij beperkt gedeelte van de voorziene XQL’98 mogelijkheden, maar biedt wel een aantal interessante bijkomende mogelijkheden. Wat ontbreekt is o.a. de mogelijkheid om de unie of de intersectie van twee verzamelingen elementen te nemen, of om een element te selecteren via zijn volgnummer. Wat er bijkomt is o.a. de mogelijkheid om wildcards en nabijheidsoperatoren te gebruiken bij het zoeken naar tekst, of om de resultaten van een query te sorteren naar de waarde van een element. De belangrijkste bijkomende functionaliteit is wel de ondersteuning van *cursoring*, dwz. de resultaten van een XQL query worden niet allemaal tegelijkertijd teruggegeven, maar in door de ontwikkelaar zelf te definiëren resultaatblokken van een bepaalde grootte. In het XQL’98 voorstel is deze functionaliteit eigenaardig genoeg niet voorzien, terwijl ze toch heel nuttig is bij het raadplegen van grote hoeveelheden XML gegevens. Tamino blijft wel de XQL filosofie trouw, en biedt geen enkele ondersteuning voor de rechtstreekse add/update/delete van XML gegevens via XQL. Als u XML gegevens opgeslagen in Tamino wenst te wijzigen dient u die er eerst uit te halen, ze via een extern programma te wijzigen, en ze vervolgens terug op te slaan.

De XML query mogelijkheden in eXcelon Corp.’s *eXcelon* omvatten nagenoeg alles van XQL’98. Wat hierin o.a. wel ontbreekt is de mogelijkheid om de resultaten van twee XQL queries te combineren, of om de resultaten van een query te sorteren naar de waarde van een element. Wat er aan toegevoegd werd is een aantal eXcelon-specifieke functies die toelaten van te zoeken binnen de tekst van een element of binnen een document dat meerdere andere documenten overkoepelt. eXcelon kent immers de notie van “gebundelde” XML documenten, die toelaten van een XML query over meerdere documenten heen te laten uitvoeren, iets wat eigenaardig genoeg ook niet voorzien is in het XQL’98 voorstel. eXcelon heeft zich ook niet willen beperken tot het opvragen van XML gegevens alleen, maar ondersteunt ook een zelfgedefinieerde XML syntax die toelaat van add/update/delete bewerkingen op de opgeslagen XML gegevens uit te voeren. In combinatie met de gepaste XQL instructies is het hiermee mogelijk om door de resultaten van een XQL query te lopen en XML elementen of attributen van XML elementen aan te maken, te wijzigen of te verwijderen.

Naast deze XML data servers zijn er nog maar een paar andere produkten die XQL ondersteunen. Zo ondersteunt de XML parser ingebouwd in Microsoft’s Internet Explorer 5.0 nagenoeg alle XQL’98 functionaliteiten. Via de DOM API kan men vanuit een VBScript of JavaScript een XQL query loslaten op een ingelezen XML document en vervolgens verderwerken met de resultaten van die query. Er is ook nog de *Infonyte XQL Suite* van Globit, een commerciële versie van een experimentele XQL engine ontwikkeld door het GMD-IPSI (Institute for Integrated Publication and Information Systems). Deze XQL Suite ondersteunt de volledige XQL’98 syntax, en voegt daar bovendien nog multi-document queries en user-defined XQL functies aan toe. De produkten van de klassieke relationele databank vendors (Oracle, Microsoft, enz.) ondersteunen begrijpelijkerwijs geen XQL: hier dient u nog steeds gewone SQL te gebruiken om XML gegevens uit de databank te halen, en vervolgens zelf programma’s te schrijven om via de DOM of SAX in die XML gegevens te gaan zoeken.

Conclusie

Zolang als een goed wiskundig data model voor XML ontbreekt blijft het moeilijk om een bijbehorende query taal te ontwerpen. Het is dan ook duidelijk dat de ontwikkeling van een standaard XML query taal nog wel wat voeten in de aarde zal hebben. XQL vormt een verdienstelijke eerste poging, maar is te sterk gericht op het louter raadplegen van XML gegevens, en te weinig op het manipuleren van die gegevens. Het feit dat XQL al met veel

succes in de huidige generatie XML data servers gebruikt wordt toont echter wel aan dat er nu reeds bedrijfszeker met XML kan gewerkt worden als formaat voor de opslag en de uitwisseling van gegevens.

Hans C. Arents (hca@itworks.be) is een onafhankelijk XML technologie adviseur bij I.T. Works, de leidende Belgische organisator van produkt- en leveranciersonafhankelijke IT seminaries. U kan meer over hem te weten komen op http://www.arents.be/.

De standaarden relevant voor de XML query taal:

XML (E**x**tensible **M**arkup **L**anguage): een bewust vereenvoudigde versie van SGML (Standard Generalized Markup Language) die toelaat van platform- en programmeertaal onafhankelijke markup talen te definiëren waarmee men zowel de inhoud als de structuur van dokumenten en data eenduidig kan vastleggen.

XQL (E**x**tensible **Q**uery **L**anguage): een query taal die toelaat om zowel de structuur als de inhoud van XML gegevens te ondervragen.

DOM (**D**ocument **O**bject **M**odel): een platform- en programmeertaal onafhankelijke API om XML gegevens te manipuleren. Het XML bestand wordt in zijn geheel ingelezen, als een boomstructuur van objecten voorgesteld, en kan vervolgens qua inhoud en structuur gemanipuleerd worden vanuit een programma of script.

SAX (**S**imple **A**PI for **X**ML): een platform- en programmeertaal onafhankelijke API om XML gegevens te parsen. Het XML bestand wordt incrementeel ingelezen, en telkens elementen en attributen in de input stream voorbijkomen worden events gesignaleerd aan een programma of script dat hierop dan kan reageren.

Informatie op het Internet:

http://metalab.unc.edu/xql/ (XQL FAQ)

http://metalab.unc.edu/xql/xql-proposal.html (XQL'99)

http://www.w3.org/TandS/QL/QL98/pp/xql.html (XQL'98)

http://www.w3.org/TR/1998/NOTE-xml-ql-19980819 (XML-QL)

http://www.w3.org/TR/xmlquery-req (XML Query Requirements document)

http://www.w3.org/TandS/QL/QL98/ (QL'98 - The Query Languages Workshop)

http://www.softwareag.com/tamino/ (Tamino XML data server)

http://www.exceloncorp.com/ (eXcelon XML data server)

http://www.globit.com/infonyte.htm (Infonyte XQL Suite)