Voorproefje XQuery stemt gematigd optimistisch

De definitie van XQuery, de toekomstige standaard XML query taal, vordert langzaam maar zeker. Hoever staan we momenteel?

Hans C. Arents

Er is de laatste maanden heel wat belangrijke vooruitgang geboekt in de definitie van XQuery, de toekomstige standaard XML query taal. Na het definiëren van een XML query data model en XML query algebra is men nu toe aan het vastleggen van de syntax van de XML query taal zelf. Alhoewel deze syntax nog niet definitief vastligt, is het toch al mogelijk een eerste voorsmaakje te geven van de mogelijkheden van deze nieuwe XML query taal.

Waar ligt de uitdaging?

In een vorig artikel “Zoeken in boomstructuren” (zie Database Magazine, nummer 4, juni 2000) hebben we reeds verduidelijkt dat het een hele uitdaging is om een goede query taal te ontwikkelen voor XML. XML is immers een uiterst veelzijdige markup taal, die niet alleen geschikt is voor het vastleggen van de struktuur en de betekenis van (semi-)gestruktureerde documenten, maar ook voor het voorstellen en uitwisselen van een grote verscheidenheid aan data. Dit betekent dat een XML query taal in staat moet zijn om documenten in XML content management systemen te ondervragen, om gegevens in XML data servers te manipuleren, en zelfs om queries over relationele of object-georiënteerde databanken uit te voeren die resulteren in XML resultaten. De XQL (E**x**tensible **Q**uery **L**anguage) taal die momenteel in de meeste XML tools ondersteund wordt kan enkel maar beschouwd worden als een eerste poging tot het ontwikkelen van zo’n universele XML query taal. Zoals we in het vorig artikel al aantoonden is XQL immers niets meer dan een XML data *ondervragings*taal, en zeker nog geen volwaardige XML data *manipulatie*taal. Het W3C (**W**orld **W**ide **W**eb **C**onsortium), de non-profit organisatie die instaat voor het vastleggen van alle cruciale Web standaarden, heeft het afgelopen jaar heel wat denkwerk verricht om te komen tot een échte standaard XML query taal, genaamd **XQuery**. Vermits een eerste publieke versie van deze nieuwe W3C standaard op 15 februari verschenen is loont het de moeite om XQuery nu al eens van nabij te bekijken.

De definitie van een XML query taal

Het definiëren van een standaard XML query taal is de verantwoordelijkheid van de XML Query Working Group van het W3C. Deze Working Group publiceerde een eerste versie van het XML Query Requirements document in januari 2000, gevolgd door een aangepaste versie in augustus van datzelfde jaar. In dat document worden de doelstellingen verduidelijkt die de Working Group zichzelf gesteld heeft: eerst het definiëren van een XML query data model, vervolgens het definiëren van een formele query algebra die op dat data model van toepassing is, en tenslotte het vastleggen van een XML query syntax gebaseerd op die query algebra. Door de query taal te bouwen op een duidelijk gedefinieerd query data model moet het immers mogelijk zijn om zonder dubbelzinnigheid te kunnen spreken over het soort XML structuren en XML data inhoud die kunnen bevraagd worden. En door te werken met een formeel gespecifieerde query algebra moet het mogelijk zijn om queries op een éénduidige manier uit te voeren en waar nodig te optimaliseren. In het document worden ook de ontwerpdoelstellingen voor de XML query taal zelf vastgelegd (o.a. dat de query taal een declaratieve taal dient te zijn, en naast een gewone syntax ook nog een XML syntax dient te hebben), en wordt via een aantal gedetailleerde Use Cases (o.a. het raadplegen van XML documenten, het raadplegen van relationele data, het uitvoeren van een full-text search) duidelijk gemaakt waar de taal precies moet toe in staat zijn. Doel is uiteindelijk te komen tot een XML query taal die toelaat van queries uit te voeren op één enkel XML document of een verzameling van XML documenten. Deze queries moeten volledige documenten kunnen selecteren (of deelboomstructuren van documenten) die voldoen aan bepaalde eisen op het gebied van document inhoud én structuur, en moeten nieuwe documenten (of deelboomstructuren van documenten) kunnen aanmaken op basis van wat zonet geselecteerd werd.

Het XML query data model

De bestaande data modellen voor semi-gestructureerde data ondersteunen niet volledig de specifieke eigenschappen van XML, meer bepaald de hiërarchische aard van XML data. Vandaar dat het XML query data model gebaseerd is op de data modellen voor boomstructuren. De twee bekendste mathematische voorstellingen van boomstructuren zijn het *model uit de grafentheorie* en het *boom-constructie model*. In de grafentheorie wordt een boom gemodelleerd als een tuple (N, E, r), waarin N een verzameling boomknoppen is, E een one-to-many relatie van bogen van ouderknoppen naar kindknoppen, en r de speciale wortel (“root”) knop van de boom. In het boom-

constructie model wordt een boom gemodelleerd als de recursieve toepassing van een boom-creatie functie op een verzameling deelbomen. In beide modellen kunnen in de boom gegevens geassocieerd worden met de knoppen in de boom ("node-labeled"), met de bogen ("edge-labeled"), of met beide. Het XML query data model is een *node-labeled, boom-constructie model*, met als bijkomende verfijning de notie van "knop identiteit" (een knop in de boom heeft een unieke identifier). Het toevoegen van knop identiteit maakt het eenvoudiger om belangrijke XML concepten zoals gekoppelde ID/IDREF attribuutwaarden of XPointer hyperlink adressen te modelleren. Het zou te ver leiden om hier dieper op dit data model in te gaan, maar u dient wel te beseffen dat dit XML query data model gedefinieerd werd met als ruimer doel het te laten uitgroeien tot het gemeenschappelijke data model voor alle XML data (dus later ook te gebruiken in de XML Schema modelleringstaal, in de XSL stylesheet taal, enz.).

De XML query algebra

In de databank wereld is het gangbaar om te eisen dat een query taal gedefinieerd wordt aan de hand van een formele query algebra (zo is bvb. SQL gebaseerd op relationele algebra). Er zijn immers twee goede redenen om eerst een formele query algebra te definiëren voor een query taal, alvorens over te gaan tot de eigenlijke definitie van de syntax van die query taal. Een eerste reden is dat een query algebra ons toelaat van een éénduidige semantiek voor de query taal vast te leggen, zodat we zeker kunnen zijn dat de operaties die we kunnen uitvoeren in de query taal goed gedefinieerd zijn. Een tweede reden is dat een query algebra een aantal wetten moet bevatten, die ons zullen toelaten om aan query optimalisatie te doen. Om aan deze vereisten te voldoen introduceert de XML query algebra een aantal operaties die kunnen uitgevoerd worden op XML data, o.a.

- **projectie**: het vastpakken van XML elementen in een XML boomstructuur
- **iteratie**: het lopen doorheen een lijst van XML elementen in een XML boomstructuur
- **selectie**: het selecteren van de XML elementen in een XML boomstructuur die aan bepaalde eisen voldoen
- **sorteren en aggregeren**: het rangschikken naar volgorde of het samen beschouwen van XML elementen
- **joining, groeperen en herstructureren**: het samennemen en het omvormen van XML boomstructuren

Op basis van deze operaties worden dan een aantal wetten i.v.m. de structuur en de betekenis van queries in deze algebra gedefinieerd, o.a. over de gelijkwaardigheid van bepaalde query uitdrukkingen of de gelijkwaardigheid van bepaalde query resultaten. Deze equivalentie wetten laten toe van bepaalde XML query operaties om te zetten in eenvoudiger operaties, of bepaalde XML query resultaten om te zetten in gelijkwaardige resultaten.

De XML query syntax

Na het definiëren van het XML query data model en de XML query algebra is de XML Query Working Group dan meteen begonnen met het vastleggen van de syntax voor **XQuery**, de XML query taal die op dit data model en deze algebra gebaseerd zal zijn. Hierbij heeft de Working Group heel wat inspiratie gezocht bij *Quilt*, een recent voorgestelde XML query taal die de beste ideeën combineert van alle voorstellen die tot nu toe geformuleerd werden (vandaar de naam "Quilt", lappendeken). Zo gebruikt Quilt o.a. de XPath zoeksyntax van XQL, leent het van SQL het idee dat een XML query uitdrukking uit een vast patroon van query clauses dient te bestaan (zoals het SELECT-FROM-WHERE patroon in SQL), en steunt het net zoals OQL (Object Query Language) op de notie dat een XML query taal een functionele taal dient te zijn, waarin verschillende soorten query expressies kunnen bestaan die onbeperkt in elkaar mogen genest worden. Maar bovenal is Quilt gebaseerd op een diep begrip van de fundamenten van XML, met name de noties van *hiërarchie* (de wijze waarop bepaalde elementen in andere elementen genest zijn), *opeenvolging* (de wijze waarop bepaalde elementen op elkaar volgen) en *verwijzing* (de wijze waarop bepaalde elementen via gekoppelde ID/IDREF attribuutwaarden naar elkaar kunnen verwijzen). Bijgevolg is XQuery net zoals Quilt in staat om queries te formuleren die gebaseerd zijn op de structuur van een XML document, en om hieruit query resultaten te produceren die ofwel de oorspronkelijke documentstructuur behouden ofwel een nieuwe structuur genereren. XQuery kan werken op "platte" structuren, zoals de rijen van een relationele databank, en vervolgens in het query resultaat XML hiërarchieën genereren gebaseerd op de inhoud van deze structuren. Het kan ook queries formuleren die werken op "niet-platte structuren" (gebaseerd op de ouder/kind relaties tussen bepaalde elementen of de volgorde van bepaalde elementen in een XML document), en het kan deze structuren behouden of volledig nieuwe structuren genereren in het query resultaat.

Een voorsmaakje van XQuery

Om u een eerste voorsmaakje te geven van wat er allemaal met de toekomstige XQuery taal mogelijk zal zijn volgen hier een aantal (eenvoudige) voorbeelden van XQuery queries. Om deze voorbeelden te kunnen begrijpen

gaan we er wel van uit dat u vertrouwd bent met de XPath zoeksyntax gebruikt door XQL (zie het vorig artikel of het XPath standaard document). In de voorbeelden worden de volgende XPath symbolen gebruikt:

Symbool	Betekenis
/	De wortel knop, of de separator tussen de verschillende stappen in het zoekpad
//	De afstammelingen van de knop (op een willekeurige diepte in de boomstructuur)
[...]	Een Boleaanse expressie om te filteren in de resultaatknoppen van een bepaalde stap
[n]	Een afkorting voor de Boleaanse expressie die de *n*de knop uit een lijst van knoppen filtert

Terugvinden van knoppen

De eenvoudigste XQuery uitdrukkingen hebben betrekking op het terugvinden van bepaalde knoppen in de XML boomstructuur. Hiertoe gebruikt XQuery de vertrouwde XPath zoekpaden, die bestaan uit een aantal stappen doorheen de boomstructuur. Iedere stap stelt een beweging in een bepaalde richting doorheen de boomstructuur voor, en iedere stap kan één of meerdere predikaten (Booleaanse expressies) omvatten om de knoppen weg te filteren die niet aan een bepaalde vereiste voldoen. Het resultaat van iedere stap is een lijst van resultaatknoppen die dienst doet als het startpunt voor de volgende stap in het zoekpad.

Voorbeeld: in het tweede hoofdstuk van het document genaamd “zoo.xml”, vind alle figuren die de ondertitel “Boomkikkers” hebben.

```
document("zoo.xml")/hoofdstuk[2]//figuur[ondertitel = "Boomkikkers"]
```

Voorbeeld: vind alle figuren in de hoofdstukken 2 tot 5 van het document genaamd “zoo.xml”.

```
document("zoo.xml")/hoofdstuk[RANGE 2 TO 5]//figuur
```

Maken van elementen

Naast het terugvinden van knoppen bevat XQuery ook de mogelijkheid tot het aanmaken van nieuwe elementen. Dit gebeurt met behulp van een zgn. *element constructor*. Een element constructor bestaat uit een begin en een eind tag voor het element, waarbinnen er optioneel een lijst van bijkomende XQuery expressies komt die zorgen voor de inhoud van het element De begin tag kan ook de waarden van één of meerdere attributen vastleggen.

Voorbeeld: genereer een nieuw <werknemer> element dat een "wnid" attribuut bevat en de geneste elementen <naam> en <job>. De waarden van het attribuut en van de geneste elementen worden bepaald door XQuery variabelen, waarvan de waarde in het voorafgaande gedeelte van de query vastgelegd werd.

```
<werknemer wnid = $id>
   <naam> $n </naam> ,
   <job> $j </job>
</werknemer>
```

Voorbeeld: genereer een nieuw element met een uit het voorafgaande gedeelte van de query afgeleide naam “tagnaam”, dat de geneste elementen <beschrijving> en <prijs> bevat.

```
<$tagnaam>
   <beschrijving> $b </beschrijving> ,
   <prijs> $p </prijs>
</$tagnaam>
```

Maken van boomstructuren

Om een eigenlijke XQuery query te kunnen uitvoeren dienen we een *FLWR* (uitgesproken als "flower") expressie op te bouwen aan de hand van FOR, LET, WHERE en RETURN clauses. Net zoals in een SQL query (waar we de SELECT, FROM en WHERE clauses dienen te gebruiken) moeten deze clauses in een welbepaalde volgorde optreden. Zo’n FLWR expressie zorgt ervoor dat de waarde van één of meerdere XQuery variabelen bepaald wordt, en dat deze waarden vervolgens gebruikt worden om het query resultaat te genereren (in het algemeen een geordend woud van knoppen). De FOR clause wordt gebruikt voor iteratie, de LET clause wordt gebruikt om de waarde van één of meerdere variabelen vast te leggen, de WHERE clause wordt gebruikt om te filteren in de

lijst van knoppen die het resultaat zijn van de FOR clause, en de RETURN clause wordt gebruikt om het uiteindelijke resultaat van de query te genereren (één knop, een geordend woud van knoppen, of een bepaalde waarde).

Om dit alles te verduidelijken zullen we een aantal eenvoudige voorbeelden van FLWR expressies bekijken, gericht op het ondervragen van een XML document met als naam "`bib.xml`" dat een lijst van `<boek>` elementen bevat. Ieder `<boek>` element bevat op zijn beurt een `<titel>` element, één of meerdere `<auteur>` elementen, een `<uitgever>` element, een `<jaar>` element, en een `<prijs>` element. Een grafische voorstelling van het document model (in XML Schema syntax) voor dit "bib.xml" document vindt u in figuur 1.

Voorbeeld: geef de lijst van de titels van de boeken gepubliceerd door Morgan Kaufmann in 1998.

```
FOR $b IN document("bib.xml")//boek
WHERE $b/uitgever = "Morgan Kaufmann"
AND $b/jaar = "1998"
RETURN $b/titel
```

Voorbeeld: geef de lijst van alle uitgevers, met voor iedere uitgever zijn naam en de gemiddelde prijs van al de boeken die hij uitgeeft.

```
FOR $u IN distinct(document("bib.xml")//uitgever)
LET $g := avg(document("bib.xml")/boek[uitgever = $u]/prijs)
RETURN
   <uitgever>
      <naam> $u/text() </naam> ,
      <gemiddelde_prijs> $g </gemiddelde_prijs>
   </uitgever>
```

Merk het gebruik op van de `distinct` functie in de FOR clause, met als doel het verwijderen van eventuele dubbels uit de lijst van uitgevers die in het document kunnen gevonden worden. Let tevens op het gebruik van de `text` functie in de RETURN clause: we willen immers niet het `<uitgever>` element zelf terugkrijgen als inhoud voor het `<naam>` element, maar enkel de tekstinhoud van dit `<uitgever>` element.

Voorbeeld: geef de lijst van de uitgevers die meer dan 100 boeken gepubliceerd hebben.

```
<grote_uitgevers>
   FOR $u IN distinct(document("bib.xml")//uitgever)
   LET $b := document("bib.xml")/boek[uitgever = $u]
   WHERE count($b) > 100
   RETURN $u
</grote_uitgevers>
```

Merk op hoe hier de FLWR expressie binnenin de element constructor zit die dient om het resultaat te genereren.

FLWR expressies kunnen ook gebruikt worden om transformaties in de structuur (en de inhoud) van documenten door te voeren, zoals geïllustreerd wordt in de volgende query, waarin een bestaande hiërarchie in het document omgekeerd wordt. Dit voorbeeld illustreert ook hoe FLWR expressies binnenin elkaar kunnen genest worden.

Voorbeeld: keer de structuur van het document om zodat i.p.v. dat ieder book een lijst van auteurs bevat, ieder afzonderlijk auteur een lijst bevat van de titels van de boeken van die auteur.

```
<auteur_lijst>
   FOR $a IN distinct(document("bib.xml")//auteur)
   RETURN
      <auteur>
         <naam> $a/text() </naam> ,
         FOR $b IN document("bib.xml")//boek[auteur = $a]
         RETURN $b/titel
      </auteur>
</auteur_lijst>
```

Wat dient er nog te gebeuren?

Zoals uit de voorgaande voorbeelden duidelijk wordt zal XQuery een heel stuk verder gaan dan XQL. Waar we in XQL enkel kunnen zoeken naar elementen die aan bepaalde structuur- of inhoudsvereisten voldoen, en we die elementen dan terugkrijgen in *dezelfde* hiërarchie en volgorde zoals ze die hadden in het ondervraagde XML document, kunnen we in XQuery niet alleen veel preciezer zoeken, maar kunnen we vooral *zelf* bepalen in welk soort boomstructuur deze zoekresultaten dienen teruggegeven te worden. XQuery gaat dus meer in de richting van een echte XML data *manipulatie*taal. Toch ontbreekt er nog altijd heel wat: zo zijn er nog steeds geen operatoren voor het inserten/updaten/deleten van elementen, vermits er nog een heleboel onbeantwoorde vragen bestaan rond wat deze operaties precies inhouden binnen het XML query data model en de XML query algebra. Bvb. als we werken met een geldig XML document (d.w.z. een document dat kan gevalideerd worden t.o.v. een DTD of een XML Schema), hoe zorgen we er dan voor dat na het inserten of deleten van elementen dit document nog steeds geldig is? Er is dus onmiskenbaar nog heel wat ernstig denkwerk aan de winkel, niet alleen in het finaliseren van de XQuery syntax, maar ook in het verder uitbouwen van XQuery tot een volwaardige query taal.

Conclusie

De huidige generatie XML tools gebruiken alle XQL als XML query taal, bij gebrek aan een standaard XML query taal. Vermits het W3C nu eindelijk een eerste versie van de nieuwe XQuery standaard uitgebracht heeft, mogen we ons ongetwijfeld binnenkort verwachten aan een eerste lichting XML produkten die deze nieuwe standaard ondersteunen. Pas dan zullen we in de praktijk kunnen evalueren of XQuery wel aan de doelstellingen voor een universele XML query taal beantwoordt, en zullen we zien of het ook mogelijk is om efficiënte XML indexeer-mechanismen te ontwerpen en performante XML query processoren te bouwen gebaseerd op het XQuery data model en de XQuery query algebra. Ondertussen moeten we voor het ondervragen van XML documenten en XML data echter blijven leven met de beperkingen van XQL.

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/.

Informatie op het Internet:

http://www.w3.org/XML/Activity#query-wg (XML Query Working Group)

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

http://www.w3.org/TR/query-datamodel/ (XML Query Data Model)

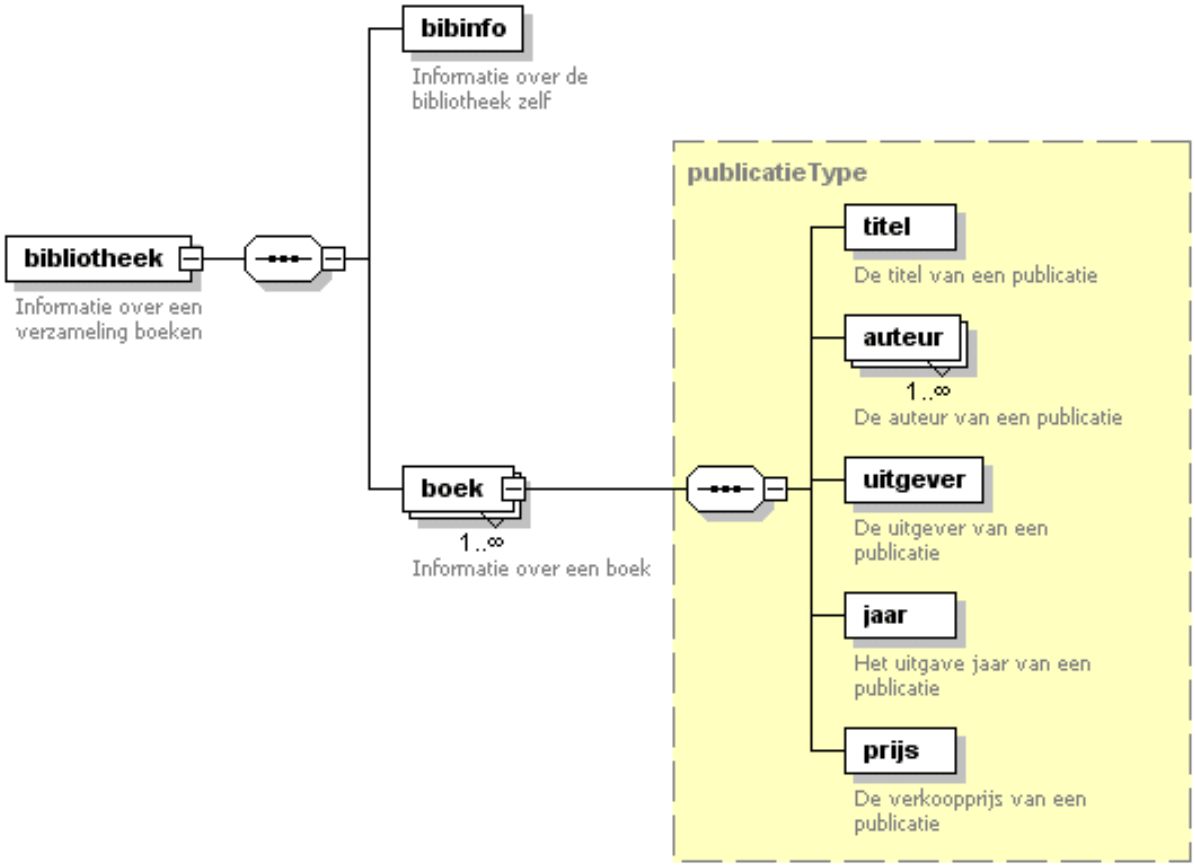
http://www.w3.org/TR/query-algebra/ (XML Query Algebra)

http://www.w3.org/TR/xquery (de XQuery taal)

http://www.almaden.ibm.com/cs/people/chamberlin/quilt.html (de Quilt taal)

http://www.w3.org/TR/xpath.html (de XPath standaard)

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



Figuur 1. Een grafische voorstelling van het document model voor het voorbeelddocument “bib.xml” (gebruikmakend van XML Schema syntax).