XML-opslag vaak beter dan 'relationeel'

XML-native databases: bent u er klaar voor?

Steven Noels

Steven Noels onderzoekt de bestaansredenen voor XML-native databases en geeft aan waar die XND's zinvol inzetbaar kunnen zijn. Technisch zijn XND's nog niet volmaakt, maar ook overigens blijft het de vraag of deze producten in vergelijking met een bijdetijds rdbms veel bestaansrecht hebben.

```
<?xml version="1.0" encoding="UTF-8"?>
<document xmlns="http://outerthought.org/ns/document"
          xmlns:xhtml="http://www.w3.org/1999/xhtml" >
  <title>Een document met Namespaces</title>
  <body>
    <paragraaf>
      Voor een beschrijving van het gebruik van
      Namespaces verwijzen wij graag naar de <xhtml:a
      href="http://www.w3.org/TR/REC-xml-names">Namespaces
      in XML</xhtml:a> aanbeveling.</paragraaf>
  </body>
</document>
```

Figuur 1: Een XML document met 'mixed content' en Namespaces.

Dat de Extensible Markup Language de oplossing zou worden voor zowat alles behalve de Noord/Zuid-problematiek, stond al vast toen de heren van het World Wide Web Consortium (W3C) kozen voor de X als eerste letter van de afkorting XML. Scully, Mulder en Copland waren er wellicht niet vreemd aan: X was synoniem voor *"the next generation"*, postmodernisme of de zoveelste zucht van de mensheid naar vernieuwing. Men zou het beter gaan doen dan in het verleden, en toch maakte dat verleden handig gebruik van de factor X om nog maar eens te stellen dat alles wel degelijk terug komt. Zo is XML gebaseerd op SGML, en zo hopen de leveranciers van objectdatabases van weleer dat XML hun eindelijk *het grote gelijk* zal geven. Sterker nog, de paleontologen van de hiërarchische databases merken dat hun kennis en ervaring rond het beheer van niet-relationele datasets weer zeer op prijs wordt gesteld.

We zouden gaan denken dat er eindelijk een einde gekomen is aan de hegemonie van de relationele database, ware het niet dat de intussen roemruchte Fabian Pascal niet in het minst geneigd is XML als een volwaardig alternatief te beschouwen.[1]

Hoe dit ook zij, de rdbms-*vendors* geven zich niet gewonnen: zowel Oracle als IBM DB2 als Microsoft SQLServer gaat steeds verder in de adoptie van XML, wat op de keper beschouwd merkwaardig is - zoals we hieronder zullen zien. Ieder van deze grote drie ondersteunt nu een min of meer doorontwikkeld *native XML column type*, en vaak kan daar zelfs XPath-gewijs verder in geselecteerd worden. De vraag rijst of de markt überhaupt ruimte biedt[2] voor die XND's, een vaandel waaronder onder andere Excelon, Tamino en X-Hive zich scharen.[3]

DICHOTOMIE

Geregelde bezoekers van IT-congressen hebben het ongetwijfeld al opgemerkt: het XML-wereldje is *anders*. Op de een of andere gekke manier trekt XML namelijk ook niet-computerwetenschappers aan, met name taalwetenschappers en een ruim aandeel aan alfa-richtingen. Binnen de XML-gemeenschap is de verhouding oestrogeen-testosteron ook iets beter opgelijnd met de algemene maatschappelijke verhoudingen. Terwijl de bèta's onder ons zich veelal prima weten te bewegen in het strakke -want mathematisch onderbouwde- E/R-model, durf ik te stellen dat de heren en dames alfa zich beter thuisvoelen in een minder strikt hiërarchisch model.

De prima donna's van beide kampen leggen zich erop toe duidelijk aan te geven hoe geweldig hun zicht op de wereld wel is, en waar de anderen de bal fundamenteel verkeerd slaan. Aangezien dit soort discussies veelal op niets eindigen en de hoffelijkheid gebiedt ieder in zijn eigenwaarde te respecteren, leest men vaak dat tegenwoordig drie datamodelleringstechnieken worden onderkend: het E/R-model, objectoriëntatie (OO) en het hiërarchische XML-model. OO modellering is hier de vreemde eend in de bijt; immers, zij tracht als enige tegelijkertijd model én gedrag neer te slaan.

En zo komen het model van entity-relationship en XML lijnrecht tegenover elkaar te staan. Vreemd genoeg haasten de gevestigde rdbms-leveranciers zich om almaar meer XML-support onder hun

Gemeenschappelijke API's

Zoals dat wel vaker gaat bij *emerging technologies*, werd de kreet XML-native database door tal van spelers in de markt gebruikt om hun product te laten uitstijgen boven die van de concurrentie. Software AG en vooral Excelon (voorheen Object Design) hadden er zelfs een heuse *corporate re-branding* voor over om de toekomst van hun bedrijf te laten afhangen van het toekomstige succes van de XND-markt. Redenen genoeg, zo vonden enkele lagere goden, om meteen een Native XML Database-consortium op te richten (www.xmldb.org/), dat zou gaan werken aan gemeenschappelijke API's, een soort ODBC/JDBC-API, zoals die bestaan voor relationele databases. Eerste wapenfeit, intussen gevolgd door een *Working Draft* en een referentie-implementatie van die API, was het opstellen van een eenduidige definitie van een XND.

motorkap in te bouwen. Een fenomeen dat lijkt op het *universal server*-initiatief van zes jaar geleden, waarbij allerlei objectrelationele extensies aan het E/R-model werden toegevoegd. De XML-jongens op hun beurt kijken dan weer haast watertandend toe wanneer de rdbms-wereld over transactiesupport, beschikbaarheid en schaalbaarheid durft te spreken.

Alle toenaderingspogingen ten spijt kan men ervan uitgaan dat XML en E/R zich orthogonaal ten opzichte van elkaar verhouden. Relationele databases slaan alles

XML is een pragmatische, maar uitermate zinvolle oplossing

neer in tabellen en rijen, wat zo zijn voordelen heeft voor indexering en versnelde toegang. De querytaal zelf kan terugvallen op het onderliggende wiskundige model, zodat voorspelbare én meetbare optimalisaties zijn te implementeren op het niveau van de search engine. En juist omwille van die performance hoeft men niet altijd op een tabelletje meer of minder te kijken. Met als resultaat dat er reeds relatief succesvolle pogingen werden ondernomen om ook XML in het keurslijf van het E/R-model te persen[4].

Figuur 2: De hiërarchie binnen het XML-document.

Toch is van een onverdeeld succes geen sprake. Dat heeft alles te maken met de aard van het beestje. XML put zich immers uit in het vastleggen van een zogenaamd semi-gestructureerd overdrachtsformaat, dat slechts een lexicale neerslag is van het feitelijke businessmodel van de overgedragen of neergeslagen informatie. Een achterliggend model is niet of nauwelijks aanwezig. En wat er daarover gedocumenteerd bestaat, is allerminst wetenschappelijk te noemen. De W3C-aanbeveling Information Set[5] omvat slechts een lijst van mogelijke *information items* die men in een *well-formed* XML-document kan tegenkomen. Ook de Post-Schema-Validation Infoset (PSVI), die handelt over XML-documenten ná validatie ten opzichte van een XML-schema, is koren op de molen voor hen die beweren dat XML beslist geen gestructureerd formaat is. Niet gek, als je bedenkt dat de voornaamste auteurs van dit soort specificaties opereren vanuit de [Natural, ed.] Language Technology Group van de Universiteit van Edinburgh.

Ook is een aantal lexicale constructies binnen XML op z'n minst *anders* te noemen: denken we hierbij aan het *mixed content model*, waarbij tekstkarakters en elementen op hetzelfde niveau kunnen voorkomen. Of *Namespaces*, de veelgewraakte maar intussen genoegzaam aanvaarde uitbreiding op de XML 1.0-aanbeveling.

Figuur 1 toont een eenvoudig XML-document dat niet eenduidig valt neer te slaan in een rdbms-schema zonder afbreuk te doen aan de inhoudelijke connotaties die erin verwerkt zijn. Figuur 2 laat duidelijk zien hoe het <**a**>-element binnen de **xhtml**-namespace op hetzelfde niveau voorkomt als de tekstinhoud van de <**paragraaf**> binnen een andere default namespace. Alleen al het probleem van *mixed content* oplossen binnen een relationele omgeving -die per definitie beter gediend is met nette encapsulatie en, als het enigszins mogelijk is, ook nog vaste veldlengten- is een heuse krachttoer.

En wat blijkt nu? Zowel alfa's als bèta's willen momenteel niet meer aan XML voorbijgaan. Het is een weliswaar pragmatische, maar uitermate zinvolle neerslag van het soort informatie dat men hetzij over de modemlijn tussen applicaties wil uitwisselen of typisch als ongestructureerde CLOB in een rdbms opslaat. Hoewel mensen in essentie geen probleem hebben met relaties -wel in de strikt theoretische zin van het woord- en het OO denken met open armen is ontvangen als middel om de wereld eindelijk in een model te gieten, blijkt dat velen maar al te graag teruggrijpen naar een sequentieel-hiërarchisch formaat voor het neerschrijven of uitwisselen van een informatieset; niet in het minst omdat een UML-diagram zo moeilijk bekt aan de telefoon.

Het gevolg is dan ook dat XML niet meer alleen voor uitwisseling, maar ook voor opslag van informatie wordt gebruikt. En waar data worden opgeslagen, duurt het niet lang of er ontstaat de noodzaak tot het bevragen van die opgeslagen gegevens. Omdat het zo moeilijk is het sequentieel-hiërarchische model van XML op te slaan in een rdbms-telraam, werd snel uitgeweken naar andere opslagvormen.

WAT IS NU PRECIES EEN XND?

Het Native XML Database-consortium (zie kader *Gemeenschappelijke API's*) noemt een aantal criteria waaraan een XML-native database moet voldoen. Samengevat gaat het om de volgende eigenschappen.

- Een XND definieert een logisch model voor het XML-*document*, niet zozeer voor zijn *inhoud*, en maakt het mogelijk

documenten volgens dit model op te slaan en op te roepen. De programmatuur heeft kennis van XML-begrippen, zoals elementen, attributen en karakterdata (PCDATA), en van de volgorde van deze entiteiten in een document. Als voorbeelden worden het XPath-datamodel, de XML Infoset, de DOM en SAX-events aangehaald.

- Een XND heeft als fundamentele logische opslageenheid een XML-document: bij rdbms'en is dit een rij of record.

Een specifiek fysiek opslagmodel is niet vereist. XND's kunnen bovenop een relationele, hiërarchische of OO database geïmplementeerd worden. XIndice, de Apache Group-reïncarnatie van de open source XND dbXML (www.dbxml.org/), heeft zijn eigen opslagformaat gedefinieerd.

Een XND is onder meer te prefereren boven een rdbms als men verwacht dat het datamodel aan wijziging zal onderhevig zijn...

Alle grote(re) XND-leveranciers passen perfect binnen deze definitie. Wat betreft het onderliggende opslagmodel is er een zekere diversiteit: X-Hive maakt gebruikt van Objectivity, B-Bop XFinity van MS SQL Server, Excelon van zijn eigen ObjectStore-gebaseerde oodbms-technologie en Software AG Tamino van iets waarvan men kan vermoeden dat het zijdelings te maken heeft met hun kennis rond hiërarchische databases. Ook XYZFind gebruikt een eigen indexformaat.

Typisch kenmerk van een XND is ook de manier waarop deze toegang verschaft tot de databaseserver. Terwijl X-Hive en Excelon een vorm van *remote DOM* ondersteunen, werkt Tamino enkel over http. Ook de Java-API is een wrapper om een http-request heen, dat als antwoord een XML-document teruggestuurd krijgt. In het huidige tijdperk van snelle XML-parsers, vaak geoptimaliseerd voor gebruik binnen een omgeving van webservices, hoeft dit geen performanceproblemen te veroorzaken. Http-operaties zijn vanzelfsprekend minder atomair dan manipulaties van remote object-referenties.

B-Bop XFinity	www.b-bop.com/
Excelon	www.exceloncorp.com/
Software AG Tamino	www.softwareag.com/tamino/
X-Hive	www.x-hive.com/
XIndice / dbXML (open source)	www.dbxml.com/
XYZFind	www.xyzfind.com/

Tabel 1: Een overzicht van representatieve XND-leveranciers.

Het identiteitspasje van de XND vermeldt ook bij voorkeur de bevragingstaal. Het is duidelijk dat XPath de honneurs waarneemt, zolang een gestandaardiseerde W3C XML Query-aanbeveling op zich laat wachten (www.w3.org/XML/Query). Een aantal XND's voegt aan XPath een set van eigen query constructs toe of verkiest een eigen querytaal te implementeren. Er wordt verwacht dat iedereen die overeind wil blijven in deze markt uiteindelijk toch de XML Query-taal (XQL) zal moeten ondersteunen. Hopelijk wordt deze dan ook pas vrijgegeven als er ook een Update-mechanisme in verwerkt is. Want anders dreigt het XND-concept een roemloze dood te sterven.

Als laatste karakteristiek nemen we de datadefinitietaal (DDL) onder de loep. XND's zullen hiervoor steeds meer gebruik maken van de relevante W3C- en Oasis-aanbevelingen: XML Schema (www.w3.org/TR/xmlschema-0/) en Relax NG (www.oasis-open.org/committees/relax-ng/). Dankzij de volledigheid van deze schematalen zal hierbij minder snel de wildgroei aan DDL-varianten ontstaan, die we kennen van rdbms-implementaties.

WANNEER EEN XND TE GEBRUIKEN?

Het zal de lezer intussen duidelijk geworden zijn dat effectieve XML-opslag slechts kans van slagen heeft binnen de domeinen waarin het zich kan onderscheiden van het rdbms-monopolie. En hoewel de E/R-*die-hards* in drommen zullen aanschuiven om mij van het tegendeel te overtuigen, zijn er wel degelijk enkele punten waar het relationele model moeilijk kan volgen.

Een van de typische eigenschappen van de meeste XND's is dat een schema (of DTD) optioneel is, en dat vaak zelfs documenten die aan verschillende schemata voldoen binnen één collectie beheerd kunnen worden. Schemavalidatie of type checking gebeurt dan ook meestal uitsluitend bij het ingeven of updaten van documenten. De gedachte dat een relationele database zonder een vooraf gedefinieerd schema is te initialiseren, valt onder het soort waanbeelden die de gemiddelde dba lichte rillingen langs de ruggegraat laten lopen. Het is dan ook een van zijn hoofdtaken de gebruiker van het serverpark ervan te overtuigen dat het bestaande schema voldoet aan zijn zich wijzigende - want vooruitschrijdende - inzichten. Fundamentele veranderingen van het rdbms-schema hebben vaak omslachtige migratieprocedures tot gevolg.

...en is ook uitermate geschikt als frontend-technologie voor aggregatie en eenduidige representatie

Lees verder op pagina 22.

IBM DB2	www-3.ibm.com/software/data/db2/extenders/xmlext/support.html
Microsoft SQL Server	www.microsoft.com/sql/techinfo/xml/
Oracle 9i	http://otn.oracle.com/tech/xml/content.html

Tabel 2: XML-ondersteuning door de drie grote rdbms-vendors.