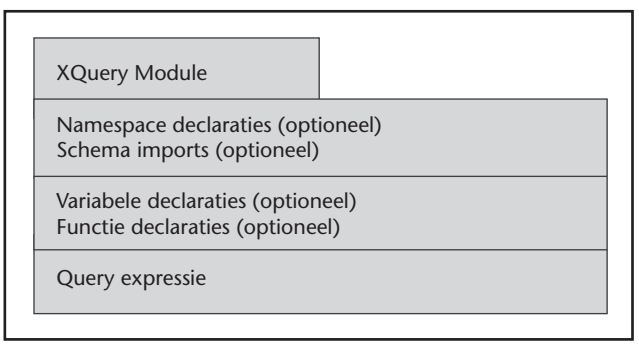
thema

Steeds meer informatie is beschikbaar in XML-formaat, maar het ontbreekt tot nu toe aan een krachtige en elegante vraagtaal voor XML-data. Met de komst van de veelbelovende XQuery vraagtaal gaat dit veranderen. XQuery bevindt zich in de laatste fase van het standaardisatieproces van het World Wide Web Consortium (W3C), maar kent al vele implementaties. In dit artikel belicht Willem Koppenol de syntax en werking van XQuery en gaat in op de relatie met andere XML vocabulaires.

XQuery: vraagtaal voor XML-data

Hoeveelheid code wordt sterk gereduceerd

Het XML-formaat is flexibel en biedt veel mogelijkheden voor data-manipulatie, -transformatie, -integratie en -conversie. Relationele databases kunnen tegenwoordig hun data in XML-formaat aanleveren, EDI- en CVS-bestanden kunnen naar XML vertaald worden en webservices wisselen XML-berichten uit. Met de komst van XQuery hebben we binnenkort de beschikking over een krachtige en elegante vraagtaal voor XML-data. Een query in XQuery wordt samengesteld uit één of meer modules. Iedere query heeft minimaal één hoofdmodule die uit drie onderdelen bestaat. Eerst een prologsectie met namespace declaraties, imports van schema's of bibliotheek-modules en globale settings, dan een proloogsectie waarin functies en variabelen worden gedefinieerd en tenslotte de sectie met de query-expressie waarin het resultaat van de query wordt gedefinieerd. Namespaces, functies en variabelen kunnen ook worden ondergebracht in bibliotheekmodulen die vervolgens door een hoofdmodule kunnen worden geïmporteerd.



FIGUUR 1. Structuur van een XQuery-module.

De zogeheten query-expressies, waarover later meer, zijn het centrale element van XQuery. Een voorbeeldquery die een zelf gedefinieerde functie aanroept is:

```
// namespace prolog
declare namespace xs = "http://www.
w3.org/2001/XMLSchema";
declare namespace nv = "http://www.novasoft.
eu/XQuery";

// declaration prolog
declare function nv:square($n as xs:integer )
as xs:integer {
   $n * $n
};

// query expressie
<Results>
  <Description>Square of 12345</Description>
  <Value>{nv:square(12345)}</Value>
</Results>
```

De query-expressie die hier wordt gebruikt is een element-constructor, die zoals de naam al aangeeft, een element maakt met de naam `Results`. De query-expressie roept de zelf gedefinieerde `square` functie aan met als parameter 12345. Het resultaat van deze functie aanroep wordt de inhoud van het `Value` element.

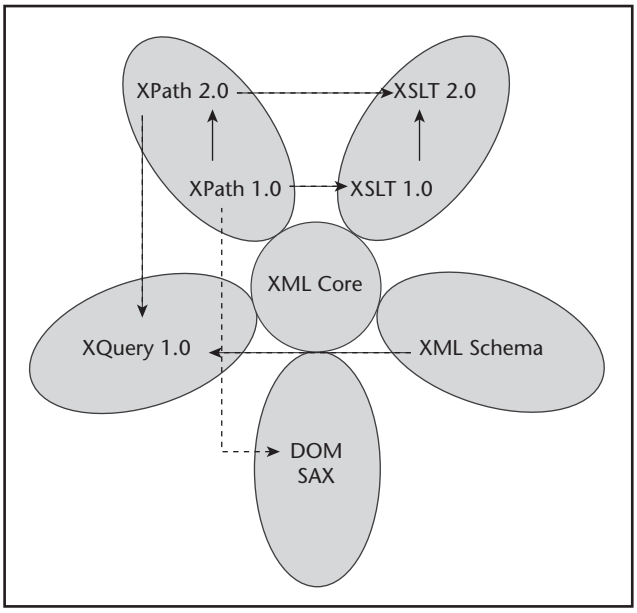
XQUERY EN XSLT XQuery kent de nodige overlap met XSLT (XML Stylesheet and Transformation Language). XSLT 1.0 is een reeds geruime tijd bestaande

standaard die veel wordt gebruikt voor het opmaken en transformeren van XML-data naar verschillende outputformaten, zoals naar HTML voor presentatie in de browser. XSLT kent een flinke schare van gebruikers en toepassingen maar wordt ook nogal eens bekritiseerd vanwege het afwijkende programmeerparadigma. Het is namelijk gebaseerd op 'template rules' en XSLT-stylesheets kennen geen mainfunctie.

XQuery 1.0 wordt in parallel met de nieuwe XSLT 2.0-versie ontwikkeld. Zowel met XQuery als met de nieuwe 2.0 versie van XSLT kunnen problemen zoals XML-transformatie en geavanceerde XML-query´s worden opgelost. XQuery gebruikt daarbij een op SQL lijkende syntax die veel ontwikkelaars die met SQL bekend zijn aanspreekt. Beide talen zijn straks een standaard met een schare aanhangers en de debatten over welke taal het best is voor welk probleem zullen voorlopig wel niet de wereld uit zijn.

XPATH XPath 1.0 wordt momenteel veel gebruikt voor het selecteren van XML-nodes binnen XSLT en DOM (Document Object Model). XPath 2.0 zal zowel gebruikt worden in XQuery als XSLT 2.0. De XML Query werkgroep van het W3C leidt dan ook samen met de XSL-werkgroep de formulering van de nieuwe XPath 2.0-specificaties. De XQuery-taal is zodanig ontworpen dat iedere geldige XPath 2.0-expressie ook een geldige XQuery is. Om precies te zijn is XQuery gedefinieerd in termen van XQuery 1.0 en het XPath 2.0-datamodel. En ditzelfde XPath-datamodel is ook het datamodel van XSLT 2.0.

XQUERY DATA MODEL XQuery 1.0 heeft zoals gezegd een gezamenlijk datamodel met XPath 2.0 en XSLT 2.0. Dit datamodel beschrijft de waarden die tijdens de evaluatie van een query kunnen voorkomen. Het XQuery-datamodel kent de volgende categorieën van waarden:



FIGUUR 2. XML-standaarden en hun relaties.

Simpele waarden: Waarden van een simpel type zoals gedefinieerd door de XML Schema-specificatie. Hieronder bevinden zich integers, strings, datums en dergelijke.

Nodes: Het XQuery-datamodel kent een zevental node-types: document, element, attribute, text, namespace, processing instructie en comment. Een XML-document is voor het datamodel een boomstructuur van nodes. De documentnode van een document is de wortel van de boom. Documentnodes en namespace nodes hebben geen ouders. Andere nodes kunnen nul of één ouder hebben. En alleen de document-node of element-nodes kunnen kinderen hebben.

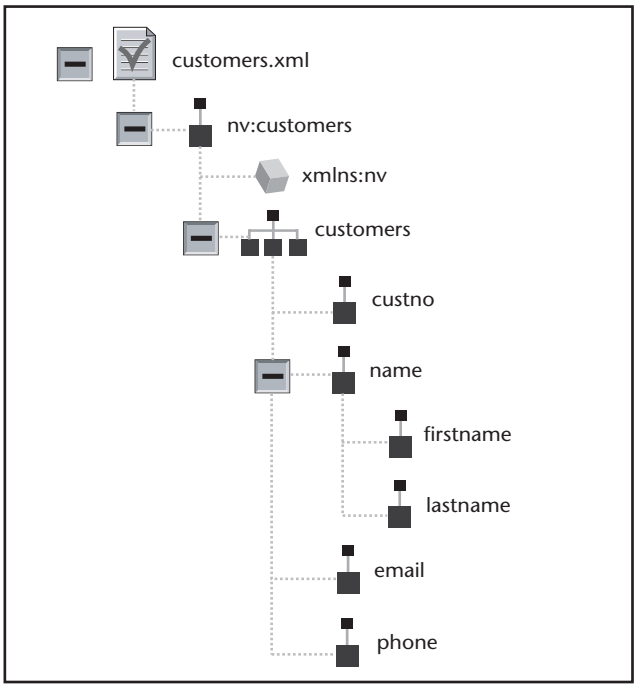
Sequenties: Een sequentie is een geordende lijst van simpele waarden of nodes. Sequenties zijn vlak hetgeen betekent dat er geen andere sequentie in de lijst mag voorkomen. Een sequentie die slechts één lid heeft, wordt een singleton-sequentie genoemd. Een belangrijke karakteristiek van het datamodel is dat er geen onderscheid is tussen een item (een node of een simpele waarde) en de singleton-instantie die dat item bevat. Een item is equivalent aan de singleton instantie die dat item bevat en vice versa.

De XQuery-taal is verder gesloten ten opzichte van het datamodel omdat de waarde van iedere expressie gegarandeerd iets uit het datamodel oplevert.

XQUERY-EXPRESSIES XQuery-expressies zijn de basisbouwstenen van XQuery en zijn de kern van de XQuery-taal. Ze kunnen van een verschillend type zijn. Veel gebruikte query-expressies zijn path-expressies, FLWR-expressies (spreek uit Flower), element/constructors en conditionele expressies. Maar ook constanten en variabelen, functie-aanroepen en operatoren vormen query-expressies.

XPATH-EXPRESSIES De XQuery-syntax heeft in haar typerende vorm een sterke gelijkenis met SQL. Voor simpele queries gebruikt XQuery echter de XPath 2.0-syntax. In XPath formuleer je expressies die resulteren in de selectie van een aantal nodes of in een waarde. Een voorbeeld van een simpele query die met een XPath-expressie alle `email` elementen uit een `customers.xml` bestand selecteert is: `//email`

In een XPath-expressie selecteer je nodes uit een XML-document door een pad in het document op te geven. Navigatie met XPath door de nodes van een XML-docu-



FIGUUR 3. De structuur van een customers.xml document.

ment kun je wel vergelijken met de navigatie door een directory-structuur van een bestandssysteem. De "/" in een pad betekent 'ga één niveau naar beneden', terwijl "//" betekent 'ga een willekeurig aantal niveaus naar beneden'. Ook constructies als "/.." om één niveau hoger te gaan en "/@id" om een attribuut te selecteren zijn mogelijk. Het resultaat van bovengenoemde query is:

```
<email>winston@hotmail.com</email>
<email>rodger@xs4all.nl</email>
<email>douglas@hotmail.com</email>
<email>stanley@xs4all.nl</email>
```

Een iets complexere query met XPath-expressie is:

```
//customer/email[ends-with(., 'nl')]
```

en deze geeft het volgende resultaat:

```
<email>rodger@xs4all.nl</email>
<email>stanley@xs4all.nl</email>
```

De XPath-expressie in dit voorbeeld bestaat uit twee delen: een pad `//customer/email` dat aangeeft in welke elementen we geïnteresseerd zijn en een predikaat `[ends-with(., 'nl')]` dat een testconditie formuleert waaraan nodes moeten voldoen om geselecteerd te worden. Het predikaat wordt voor ieder element één keer geëvalueerd. De expressie "." (dot) in het predikaat refereert aan de node die het predikaat aan het testen is, dat wil zeggen de geselecteerde `email`. De `ends-with()` functie is één van de vele nieuwe functies in XPath 2.0.

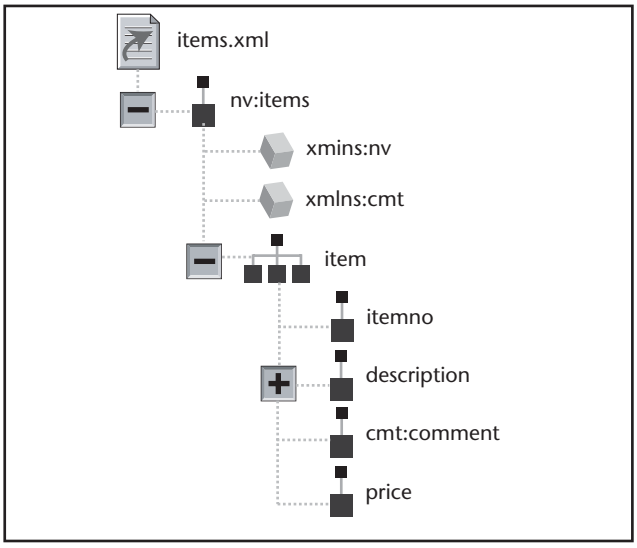
Met XPath-expressies kunnen ook complexere query's worden gemaakt. Veel ontwikkelaars ervaren dan echter dat de XPath-syntax moeilijk leesbaar wordt en gemakkelijk aanleiding geeft tot onbegrip en fouten. XQuery voorziet daarom voor complexere query's in andere syntactische mogelijkheden.

CONSTANTEN EN VARIABELEN Constanten en variabelen zijn de meest basale XQuery expressies. Constanten zijn letterlijke representaties van waarden zoals:

```
12345
"My String"
```

Variabelen beginnen altijd met een dollarteken ($), gevolgd door de naam van de variabele. Variabelen worden door expressies aan waarden gebonden en je kunt variabelen ook aan functies meegeven. In de onderstaande voorbeeld query is 0025 een constante en `$itemno` een variabele:

```
let $itemno := "0025"
return doc("items.xml")
    //item[itemno=$itemno]
```



FIGUUR 4. De structuur van een items.xml document.

FLWR EXPRESSIES FLWR-expressies, uitgesproken als 'flower' zijn de meest typerende XQuery-expressies. Syntactisch lijken ze op SQL select statements en ze hebben ook vergelijkbare mogelijkheden.

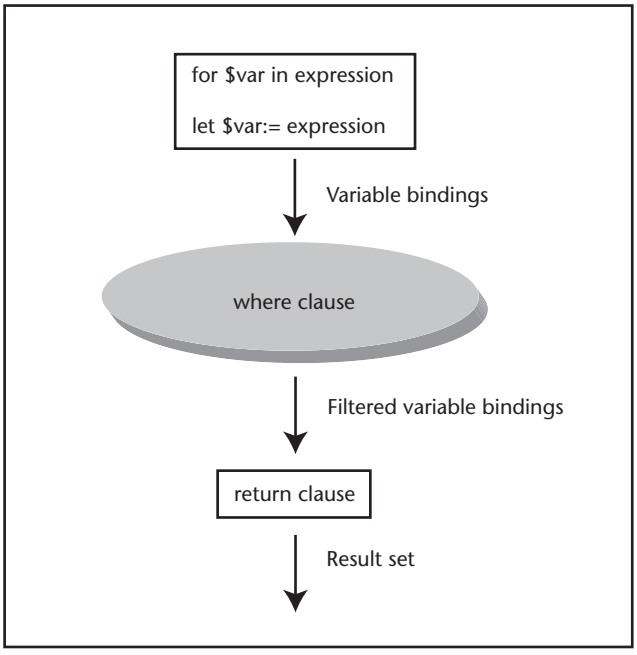
FLWR staat voor de vier clausules waaruit een expressie van dit type bestaat namelijk: For-Let-Where-Return. FLWR expressies worden gebruikt om te itereren over groepen nodes en variabelen te binden aan resultaten. Onderstaande query retourneert bijvoorbeeld uit het document `customers.xml` een lijst met de `lastname` elementen van alle customers:

```
<customers>
{
  for $c in doc("customers.xml")//customer/
name
  return
    $c/lastname
}
</customers>
```

Het resultaat van deze query ziet er als volgt uit:

```
<customers>
    <lastname>Fastchess</lastname>
    <lastname>Niceguy</lastname>
    <lastname>Bigeyes</lastname>
    <lastname>Smarthead</lastname>
</customers>
```

FLWR-SYNTAX EN SEMANTIEK Een FLWR-expressie moet op zijn minst één `for` of `let` clausule bevatten. Deze clausules evalueren expressies en kennen het resultaat van deze expressies toe aan variabelen. Bij een `for` clausule worden er een aantal verschillende bindingen met een variabele gerealiseerd als de expressie een lijst met resultaten oplevert. Bij een `let`-clausule krijgt de variabele in dat geval één enkele binding met de hele lijst met resultaten. Deze binding tussen variabelen en resultaat worden vervolgens doorgegeven naar een `where` clausule die er op grond van bepaalde condities een filter op loslaat. Deze `where` clausule is vergelijkbaar met de `where` clausule van in een SQL select statement. De `return` clausule construeert de resultaten van de FLWR expressie en wordt voor iedere variabele binding die het filter (`where clausule`) overleefd één keer aangeroepen. Eén en ander wordt geïllustreerd door figuur 5.

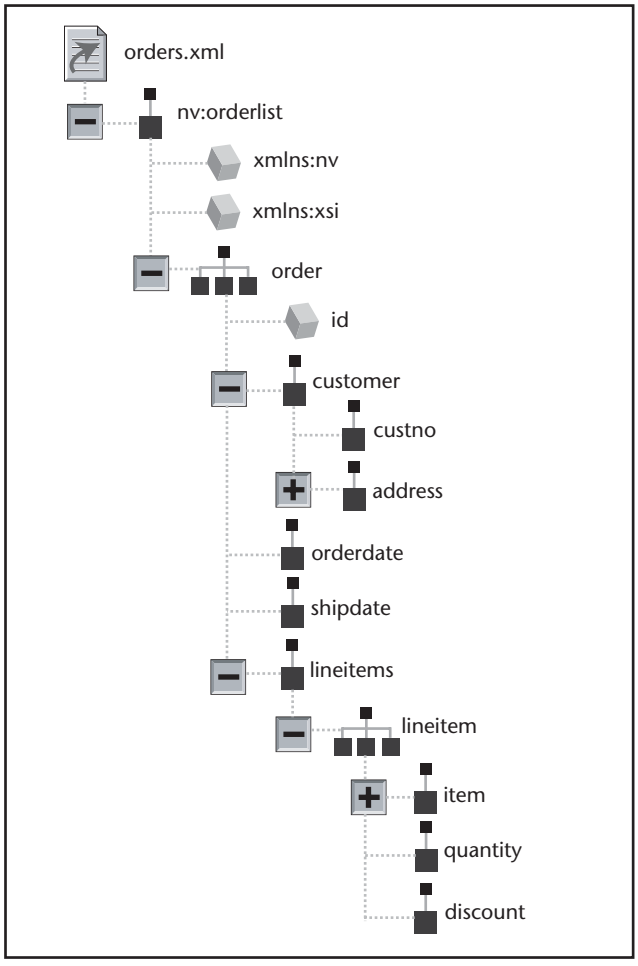


FIGUUR 5. Onderdelen en volgorde van uitvoering van een XQuery FLWR expressie.

Query's met FLWR-expressies zijn nuttig voor het verwerken en herstructureren van data afkomstig uit één of meer documenten. Zo ziet een query die een lijst maakt van alle items in het document `items.xml` die zijn besteld door customers uit het document `orders.xml`, er als volgt uit:

```
for $i in doc("items.xml")//item
let $p := doc("orders.xml")//order
where $i/itemno = $p//itemno
return
  <ordered_item>
  {$i/description/text()}
  </ordered_item>
```

De `for` loop itereert over ieder `item` element in `items.xml`, terwijl de `where` clausule alleen die items selecteert wiens `itemno` element voorkomt in het `order` element van `orders.xml`, zoals gespecificeerd in de `let` clausule. De `return` clausule maakt tenslotte



FIGUUR 6. De structuur van een orders.xml document.

binnen een `<ordered_item>` element een lijst van alle item beschrijvingen die voorkomen in het uiteindelijke resultaat. Het resultaat van de query is:

```
<ordered_item>XBox</ordered_item>
<ordered_item>Gameboy</ordered_item>
<ordered_item>Playstation</ordered_item>
<ordered_item>Laptop</ordered_item>
<< einde kader met computercode >>
```

SCOPE VAN VARIABELEN Een variabele moet eerst gebonden worden in een `for` of `let` clausule alvorens hij gebruikt mag worden. De variabele blijft vervolgens in scope tot het eind van de FLWR-expressie waarin hij werd gebonden. Een locale binding overschrijft daarbij een globalere binding. Als een variabele dus bij het binden al gebonden was, verdwijnt de oorspronkelijke binding uit het zicht, tot de variabele uit scope gaat. Vanaf dat moment refereert de variabele weer aan zijn oorspronkelijk binding. Een voorbeeld is:

```
for $p in doc("orders.xml")//order
   return
      <new_orders>
      {
      for $j in $p//item
      for $p in doc("items.xml")//item
      where $p/itemno=$j/itemno
      return
         <item>{$p/description/text()}</item>
      }
      </new_orders>
```

De voorafgaande query maakt een lijst met alle item beschrijvingen in order.xml. De buitenste for clausule bindt de variabele $p aan ieder order-element in het document order.xml. In de binnenste for-clausule wordt $p hergebruikt en gebonden aan ieder item element in items.xml. Als de binnenste iteratie is afgelopen gaat verlaat $p zijn scope en krijgt weer zijn oorspronkelijke waarde uit de buitenste iteratielus (wijzend naar order in order.xml). Het resultaat van de query is:

```
<new_orders>
     <item>Laptop</item>
     <item>Server PC</item>
</new_orders>
<new_orders>
     <item>XBox</item>
     <item>GameBoy</item>
</new_orders>
```

JOINS IN XQUERY Eén van de krachtigste technieken in SQL is de join. Met een join-operatie kunnen data uit verschillende tabellen van een relationale database worden verenigd. Met FLWR expressies kun in XQuery vergelijkbare functies uitvoeren op verschillende documenten. In de zojuist gegeven voorbeeld query werden twee `for` clausules genest in de `return` clausule van de buitenste FLWR-expressie. De query retourneerde die items uit `items.xml` wiens itemno overeenkwam met een itemno in `orders.xml`. Een dergelijk type relatie wordt een *inner join* genoemd.

Ook het andere join type, de *outer join*, is in XQuery mogelijk. Een outer join is een join die de informatie uit één of meer van de deelnemende documenten vasthoudt ook al zijn er elementen die geen match hebben in de andere documenten. Een voorbeeld is de volgende query die een lijst oplevert van alle customers inclusief de `orderid` voor die customers die een order hebben geplaatst:

```
for $u in doc("customers.xml")//customer
return
   <customer id={$u/custno}>
   <name>{$u//firstname/text()}
         {$u//lastname/text()}</name>
         {
           for $p in doc("order.xml")//order
           where $u/custno = $p//custno
           return
           <order>{$p/@id}</order>
         }
   </customer>
```

Uit deze voorbeelden wordt duidelijk dat de FLWR-expressie in XQuery vergaande mogelijkheden bieden voor het processen en transformeren van data in XML-documenten.

XQUERY OPERATOREN XQuery kent verder de meeste operatoren die je ook in andere programmeertalen tegenkomt. Zo kent XQuery rekenkundige operatoren zoals + en - , vergelijkingsoperatoren zoals < en > en booleaanse operatoren. Deze operatoren werken zoals je verwacht dat ze werken. Een voorbeeld query met het gebruik van vergelijkingsoperatoren in een element/ constructor is:

```
for $item in doc("orders.xml")//item
return
<result>
  {$item/itemno}
  <ApprovalLevel>
  {if ($item/price >= 200) then 2 else
   if ($item/price <= 50) then 0 else 1}
  </ApprovalLevel>
</result>
```

Daarnaast zijn er ook een aantal nieuwe operatoren die specifiek zijn voor XQuery en die toegepast worden

op sequenties en nodes. Operatoren die werken op sequenties zijn bijvoorbeeld `except`, `union` en `intersect`. Node specifieke operatoren zijn node-after (>>), node-before (<<) en node-equal(==). De volgende query met `except` geeft alle `itemno` terug die niet aan de gegeven voorwaarde voldoen:

```
let $allitem :=
  doc("items.xml")//item/itemno
let $orderitem:=
  doc("items.xml")//item[price=1000]/itemno
return
<result>
  {$allitem except $orderitem}
</result>
```

SLOTWOORD XQuery is momenteel nog een relatief nieuw en onbekend XML vocabulaire. Het is primair een vraagtaal voor XML-data. XQuery gaat echter verder en omvat ook de nieuwe generatie voor XML-selectie (XPath 2.0), XML serialisatie, full-text search en functionele XML data modellering. XQuery is kortom een veelbelovende standaard die bovendien wordt omarmd door grote spelers in de markt zoals IBM, Oracle en anderen. Het is dan ook de verwachting dat XQuery de komende jaren sterk in momentum gaat toenemen. Op den duur kan XQuery zelfs het inmiddels hoogbejaarde SQL als primaire vraagtaal gaan verdringen. Ontwikkelaars en architecten die zich bezig houden met XML-webservices en Service Oriented Architectures (SOA) zullen profijt hebben van XQuery. De hoeveelheid code die nodig is voor XML-webservices wordt door XQuery in belangrijke mate gereduceerd.

drs. Willem Koppenol is Senior Trainer en Product Specialist Software Development bij Twice IT Training (e-mail: wkoppenol@twice.nl)

PATCHES PATCHES PATCHES PATCHES

Swing is King Kong

Hans Muller op de gelijknamige Blog op Java.net heeft moeite met het feit dat Swing nu de meest gebruikte GUI toolkit in Noord-Amerika is. Nu ja moeite, hij heeft moeite met het verbergen van zijn enthousiasme daarover.

Muller: 'Ik heb nagedacht over een manier om op een bescheiden manier aan te kondigen dat geen geringere autoriteit dan Evans Data Corporation (EDC) bekend heeft gemaakt dat Swing de leidende GUI Toolkit voor Noord Amerikaanse ontwikkelaars is.'

Met een gebruikspercentage van 47%, en een groei van 27% sinds herfst 2004, heeft Swing WinForms ingehaald.

Muller: 'Microsoft werd vaak als een "eight hundred pound gorilla" aangeduid. Dankzij de vasthoudendheid en het enthousiasme van Swing ontwikkelaars overal ter wereld, hebben we de gorilla en de kooi van het eiland gegooid. Wij zijn het nieuwe alpha mannetje, de King Kong van GUI toolkits.'

Siebel's Component Assembly eerste op SOA gebaseerde CRM-oplossing

Siebel Systems heeft de Siebel Component Assembly aangekondigd. Het betreft hier een nieuwe lijn open, op de gangbare standaarden gebaseerde en uiterst flexibele producten. Hiermee worden nu voor het eerst de essentiële bouwstenen aangeboden om sneller, eenvoudiger en efficiënter CRM-toepassingen op maat samen te stellen. De nieuwe oplossing is het resultaat van meer dan drie jaar research en ontwikkeling in nauwe samenwerking met BEA Systems, IBM en Microsoft (onder de noemer 'Project Nexus') en is de enige op een Service Oriented Architecture (SOA) gebaseerde CRM-oplossing die native kan draaien op zowel een .NET als een J2EE-toepassingsserver.

Siebel Component Assembly is speciaal ontworpen om CRM-toepassingen goedkoper, sneller en met minder risico's te kunnen bouwen en implementeren, speciaal voor organisaties die een verregaande maatoplossing nodig hebben, maar wel op basis van de gangbare standaarden. Zonder de nieuwe oplossing zouden deze organisaties gedwongen zijn om hun eigen toepassingen van nul af aan op te bouwen. Siebel Component Assembly steunt volledig op de principes van de Siebel Customer Adaptive Architecture, met schaalbaarheid, openheid en op de standaarden gebaseerde ontwikkel- en runtime-ondersteuning. Daarmee kan dit tot dusver achtergebleven segment van de markt voor CRM-toepassingen nu ook profiteren van alle voordelen van de Siebel Customer Adaptive Solutions (zie apart persbericht voor meer informatie over Siebel Customer Adaptive Solutions). Siebel Component Assembly sluit aan op de rol die Siebel Systems speelt op de markt voor on-premise en gehoste CRM. De Siebel Customer Adaptive Solutions staan nu ter beschikking van alle mogelijke klanten, hoe complex hun vereisten of implementatievoorkeuren ook mogen zijn.

Volgens AMR Research bestaat de CRM-markt voor meer dan tachtig procent uit organisaties die hun eigen toepassingen willen maken of die bestaande toepassingen willen uitbreiden met functionaliteiten op maat. Dit willen ze doen met behulp van native ontwikkeltools op een toepassingsserver in een servicegeoriënteerde architectuur. Het maken van dergelijke maatwerkoplossingen is echter altijd een complexe, moeizame en dure aangelegenheid geweest. Siebel Component Assembly is de eerste oplossing die specifiek is gericht op de markt voor CRM-maattoepassingen, een markt die wordt geraamd op 24 miljard dollar.