XML-dataservers uit de kinderschoenen

Een nieuwe manier om data bronnen en applicaties te integreren binnen een Web omgeving

Hans C. Arents

XML werd oorspronkelijk ontworpen als een krachtig gegevensformaat voor de opslag en het beheer van elektronische documenten. Het wordt echter duidelijk dat XML ook een belangrijke rol zal vervullen als gegevensformaat voor de opslag en de uitwisseling van data. Vooral in een Web omgeving bestaat er een grote nood aan een oplossing die toelaat van gegevens uit verschillende data bronnen met elkaar te integreren en uit te wisselen met externe Web applicaties. Een nieuwe categorie databanksystemen, zogenaamde XML data servers, maakt het mogelijk om een geïntegreerd XML zicht te krijgen op onderliggende heterogene data bronnen en applicaties. In dit artikel verduidelijken we de precieze rol van deze XML data servers, bespreken we de verschillen tussen de twee soorten XML data servers, en bekijken we een aantal commerciële produkten. We gaan ook na hoe relationele databanken nog steeds een belangrijke rol zullen blijven spelen in een dergelijke oplossing.

Waarom XML voor data?

Bedrijven worden tegenwoordig op twee plaatsen geconfronteerd met een toenemende vraag om bestaande applicaties via het Web te integreren. Een eerste plaats is binnenin het bedrijf zelf, waar men er naar streeft nog slechts één enkele Web gebruikersinterface te hebben boven verschillende applicaties. Men kan daartoe zelf een op maat gemaakte Web integratie laag bovenop deze applicaties bouwen (bvb. EJBs binnenin een Web applicatie server), of een kant-en-klare Web integratie oplossing kopen. Beide benaderingen zijn echter duur en meestal niet flexibel genoeg om de steeds wijzigende gebruikersbehoeften vlot te kunnen volgen. Een eenvoudiger oplossing bestaat er in de applicaties zelf ongemoeid te laten en enkel hun gegevens te integreren. Dit houdt in dat men selectief de benodigde gegevens uit de achterliggende data bronnen en applicaties haalt en deze integreert tot één enkel stel gegevens waar een Web applicatie dan rechtstreeks gebruik kan van maken. XML is een uitstekend data formaat voor een dergelijke *gegevens integratie*, vermits het een standaardiseerde syntax biedt om die gegevens te beschrijven, en een rijk gamma aan bijkomende standaarden om die gegevens te benutten.

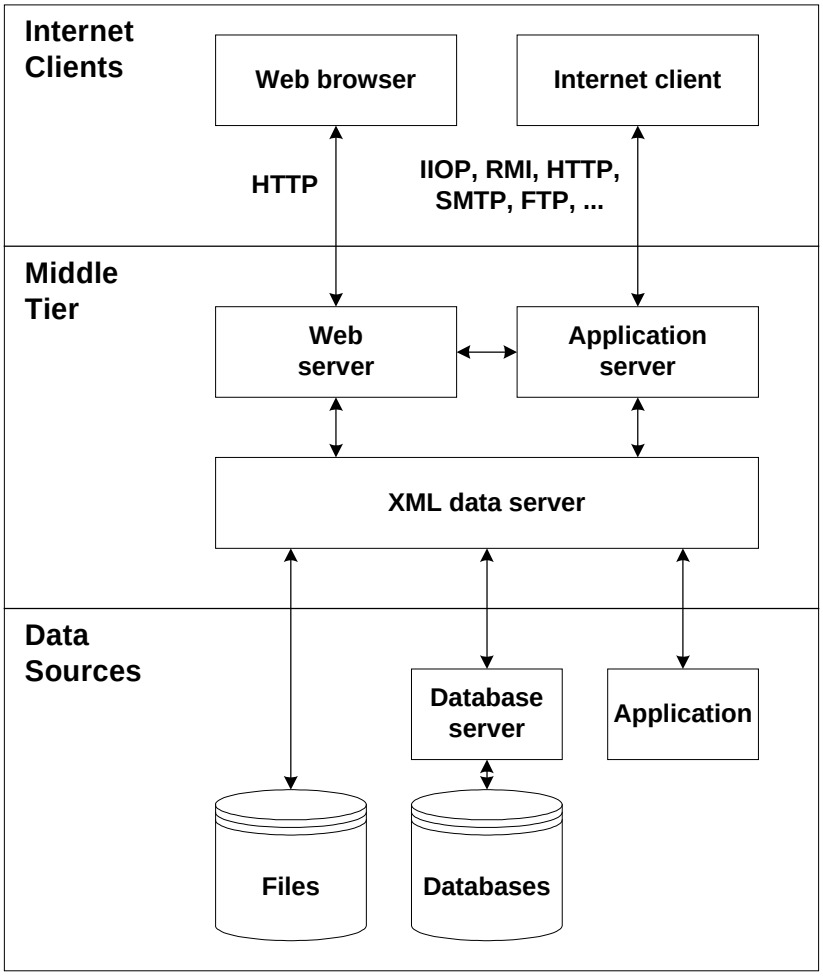
Een tweede plaats waar de vraag naar applicatie integratie optreedt is tussen bedrijven onderling, waar men er naar streeft om de eigen applicaties over het Web heen te koppelen aan de applicaties van de partners of toeleveranciers. Men kan daartoe kiezen voor een ‘tight coupling’ tussen de applicaties, waarbij ze elkaar rechtstreeks over het Web aanroepen (via een technologie zoals CORBA). Of men kan kiezen voor de eenvoudiger oplossing van een ‘loose coupling’ waarbij de applicaties enkel de benodigde gegevens over het Web uitwisselen, en die gegevens vervolgens elk op hun eigen manier gebruiken. Opnieuw vormt XML een uitstekende keuze als data formaat voor een dergelijke *genetwerkte gegevensuitwisseling*, vermits het toelaat de uitgewisselde gegevens op een platform en programmeertaal-onafhankelijke manier te beschrijven. De vraag is dan door welke tool die XML-gebaseerde gegevens integratie of genetwerkte gegevensuitwisseling precies zal beheerd worden. Het is hier dat een nieuwe categorie databanksystemen, zgn. XML data servers, een onmisbare rol begint te spelen.

De rol van de XML data server

Een typische Web applicatie gebruikt de ondertussen welbekende *three tier* architectuur, waarbij de data tier (met daarin de data bronnen) via een middle tier (met daarin de Web server en een eventuele applicatie server) in verbinding staat met de client tier (met daarin de Web browser of een andere Internet client). De plaats van de XML data server in deze architectuur is eveneens in de middle tier, waar het de rol speelt van data toegang/data transfer middleware bovenop de data bronnen en applicaties. De beste manier om de werking van de XML data server te begrijpen is door die te vergelijken met de werking van de vertrouwde Web server. De belangrijkste taak van de Web server bestaat er in te antwoorden op HTTP requests van de Web browser door HTML documenten terug te sturen. De Web server staat ook in voor het beheer van de interactie van de gebruiker met een back-end databank of applicatie, eventueel door een beroep te doen op de diensten van een Web applicatie server.

Een XML data server gedraagt zich gelijkaardig aan een Web server: hij kan ook antwoorden op HTTP requests van de Web browser, maar ditmaal door XML documenten terug te sturen. Deze XML documenten kunnen dan door de Web server of de Web browser omgezet worden naar HTML en op de vertrouwde manier bekeken worden. Op twee vlakken gedraagt een XML data server zich echter totaal verschillend. De XML data server is

vooreerst in staat om aan de Web server of de applicatie server de indruk te geven dat de onderliggende data bronnen uit één enkele verzameling XML bestanden bestaat. In plaats van met al die verschillende data bronnen te moeten praten, kan de Web server of de applicatie server onrechtstreeks via die XML bestanden werken. De XML data server staat in voor het beheer van de XML bestanden, en voor het synchroon houden van de XML gegevens met de werkelijke data. Daarnaast is de XML data server niet alleen in staat om HTTP te praten, maar kan hij de XML gegevens ook via andere Internet communicatie protocollen uitwisselen met een andere Internet client. De XML data server staat dan in voor het beheer van die communicatie, voor het verwerken van de uitgewisselde XML berichten en voor het doorspelen van de XML data aan de achterliggende data bronnen.



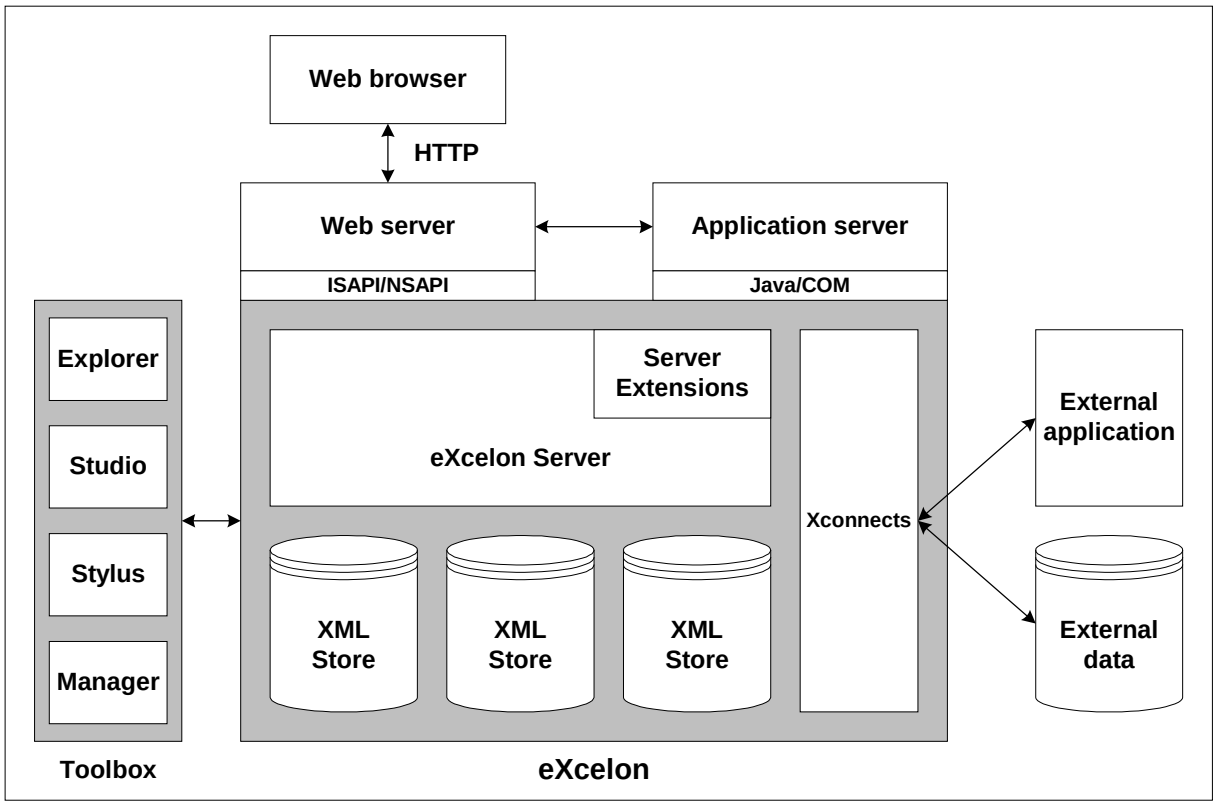
Figuur 1: De rol van de XML data server in een Web applicatie architectuur.

De twee soorten XML data servers

Een XML data server kan zijn rol vervullen op twee manieren: hij kan de geïntegreerde XML gegevens permanent opslaan, of hij kan de XML gegevens enkel op aanvraag genereren. Een *repository* XML data server is gericht op de efficiënte opslag, het beheer en de opvraging van XML gegevens. Hij doet dienst als een soort XML data cache, een buffer waarin de geïntegreerde XML gegevens beschikbaar zijn voor de Web applicatie. Hierdoor vermijdt hij o.a. dat de achterliggende data bronnen te zwaar belast worden wanneer deze massaal via het Web geraadpleegd worden. Een repository XML data server kan ook dienst doen als een volledig losstaand XML data beheersysteem, waarbij er geen achterliggende data bronnen zijn, maar alle gegevens meteen in XML beheerd worden. Een *dynamische* XML data server daarentegen is gericht op het genereren, verwerken en uitwisselen van XML gegevens. Hij doet dienst als een soort XML vertaalbrug, waarbij gegevens op aanvraag vanuit een achterliggende data bron naar XML omgezet worden en omgekeerd. De repository XML data servers die men momenteel op de markt vindt onderscheiden zich van elkaar door de wijze waarop ze XML opslaan, en door de graad van integratie die ze bereiken met de achterliggende data bronnen. We bekijken in detail Object Design's eXcelon en Software AG's Tamino, die op duidelijk verschillende soorten Web toepassingen toegespitst zijn. Als voorbeeld van een dynamische XML data server bekijken we in detail Bluestone's XML-Server.

eXcelon

eXcelon is een repository XML data server die gebaseerd is op Object Design's ObjectStore object-georiënteerde databank. Gelanceerd in maart 1999 en momenteel reeds in versie 2.0, is eXcelon uitgegroeid tot een volwaardig produkt waarin de nadruk vooral ligt op het gebruik van de inherente extensibiliteit van de erin opgeslagen XML gegevens. Daartoe worden in eXcelon de XML gegevens, afkomstig uit externe data bronnen of applicaties, enkel opgeslagen als goedgevormde XML bestanden (dwz. bestanden die enkel voldoen aan de "wellformedness" regels van XML, nodig om een DOM parse tree te kunnen opbouwen, en die dus niet gevalideerd worden met betrekking tot een DTD). Deze XML bestanden worden opgeslagen in het virtueel file systeem van een *XML Store*, waar naast XML bestanden ook andere bestanden (beelden, documenten, e.d.) kunnen opgeslagen worden. Bij het opslaan van een XML bestand wordt dit bestand geparsed en geïndexeerd, en wordt ieder XML element erin als een apart object toegankelijk. Deze XML bestanden kunnen dan via XQL ondervraagd worden, waarbij de resultaten opnieuw als XML teruggegeven worden. Via XSL kunnen deze XML resultaten omgevormd worden naar HTML om vervolgens naar een Web server gestuurd te worden. M.b.v. een aantal eXcelon-specifieke extensies aan XQL (XQL is immers nog niet officieel goedgekeurd als een query taal standaard) kan men XML elementen toevoegen, wijzigen, of verwijderen. Vermits de XML bestanden enkel goedgevormd dienen te zijn, en geen DTD dienen te respecteren, heeft de applicatie ontwikkelaar hierbij een grote vrijheid om de opgeslagen XML gegevens op elk moment op een flexibele manier te wijzigen en uit te breiden.



Figuur 2: De architectuur van Object Design's eXcelon.

Als de mogelijkheden van XQL niet volstaan om de XML gegevens te manipuleren is het steeds mogelijk om Java of COM-gebaseerde *Server Extensions* te schrijven die de XML bestanden rechtstreeks via de DOM aanspreken. Zowel de XQL queries als deze server extensies kunnen via een URL vanuit de Web server, of via een Java of COM API call vanuit een applicatie server of een andere applicatie opgeroepen worden. Om gegevens in XML vorm in eXcelon binnen te halen biedt eXcelon *xConnects* aan, een 70-tal voorgedefinieerde data-naar-XML mappings voor Data Junction die o.a. kunnen gebruikt worden om Excel of Dbase III gegevens naar XML om te zetten. Daarnaast kan men ook via ODBC of OLE DB relationele gegevens in XML formaat in eXcelon binnen-halen. Wenst men niet enkel gegevens in XML formaat binnen te halen, maar vanuit gewijzigde XML bestanden gegevens terug te schrijven naar de achterliggende data bronnen, dan dient men zelf server extensies te schrijven die hiervoor instaan. Om de vereiste data integriteit tijdens deze schrijfoperaties te bewaren dienen deze server extensies dan als onderdeel van een MTS (Microsoft Transaction Server) transactie uitgevoerd te worden.

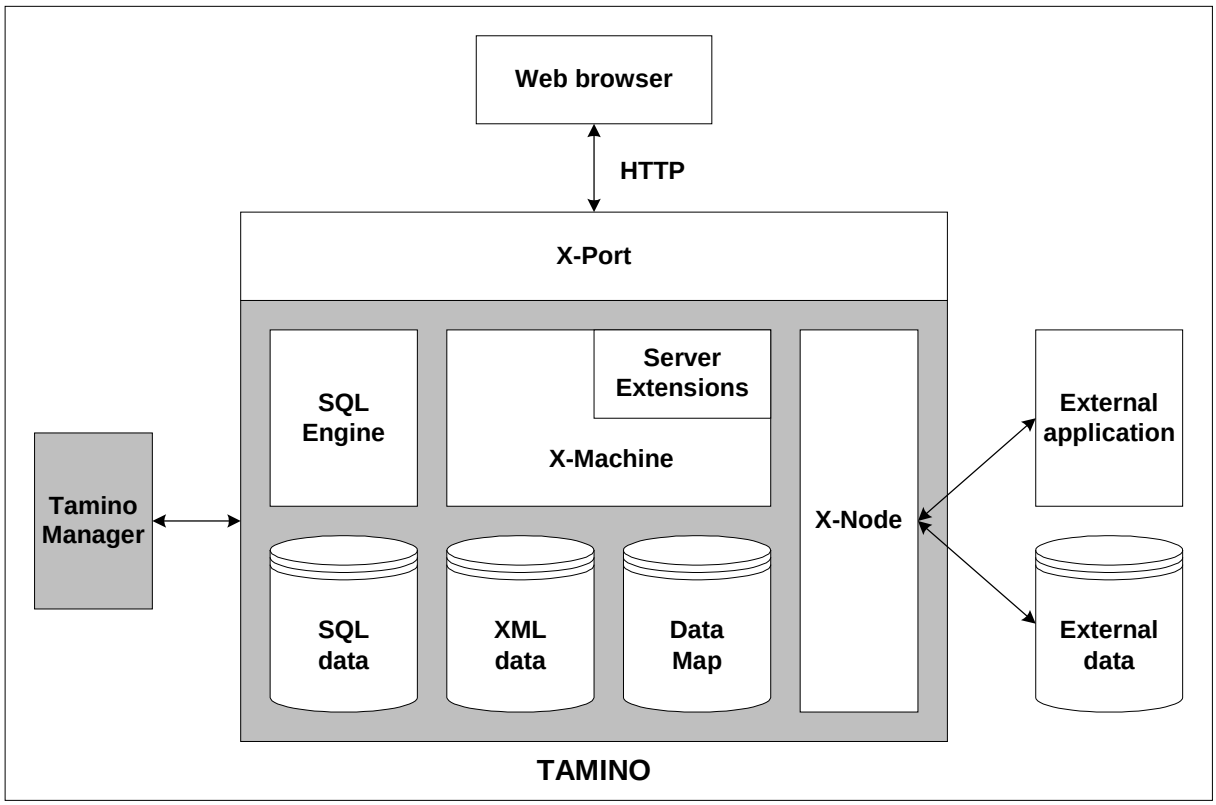
De grote vrijheid die eXcelon biedt om op een ad-hoc manier de structuur en de inhoud van de erin opgeslagen XML gegevens te wijzigen komt vooral van pas in toepassingen waar men via een Web interface vaak veranderende of slecht gedefinieerde gegevens ter raadpleging wenst aan te bieden. Zo wordt eXcelon o.a. succesvol gebruikt in de back-end van de Dell EMEA e-commerce site, waar het de beste manier bleek om de moeilijk definitief vast te leggen PC configuratiegegevens op een uniforme manier te beheren en aan de eindgebruiker te presenteren. Vermits eXcelon niet in staat is om de erin opgeslagen XML gegevens automatisch synchroon te houden met de achterliggende data bronnen dient eXcelon voornamelijk als read-only XML data cache of als losstaand XML data beheersysteem. Een belangrijk punt in eXcelon's voordeel is wel dat het beschikt over een volledige set visuele ontwikkeltools (*Explorer*, om XML gegevens interactief in/uit te voeren, aan te passen en via XQL te ondervragen, *Studio* om de structuur van de XML gegevens te definiëren en van daaruit klassen en code templates in Java/C++/VB te genereren, *Stylus* om XSL stylesheets aan te maken, en *Manager* voor het beheer van de data server) die het de ontwikkelaar gemakkelijk maken om snel Web toepassingen te bouwen.

Tamino

Software AG, de maker van o.m. de Adabas databank, heeft vrij snel het belang van XML ingezien, en is reeds twee jaar geleden begonnen aan de ontwikkeling van een volledig nieuwe databank specifiek gericht op de opslag en het beheer van native XML. Het resultaat is Tamino, een repository XML data server waarin de nadruk vooral ligt op het beheer van goedgestructureerde XML gegevens en op het ondersteunen van de toegang vanuit die gegevens naar de externe data bronnen. In tegenstelling tot eXcelon kunnen de XML gegevens, afkomstig uit externe data bronnen of applicaties, hier worden opgeslagen als gevalideerde XML bestanden, wat inhoudt dat ze een DTD dienen te respecteren. Tamino kan deze DTD dan gebruiken om in een *Data Map* schema te definiëren hoe de XML gegevens precies dienen geïndexeerd te worden en waar die gegevens eigenlijk vandaan komen. Een XML bestand in Tamino mag immers data bevatten die zowel intern in native XML door de *X-Machine* XML databank kernel beheerd wordt, als data die afkomstig is uit een externe Adabas databank of een andere externe relationele databank die via ODBC, JDBC of OLE DB toegankelijk is. De *X-Node* is verantwoordelijk voor het lezen en schrijven van de data uit die externe data bronnen, met behulp van de schema's bijgehouden in de Data Map, en staat ook in voor de data integriteit van deze transacties. Een applicatie die gebruik maakt van de XML gegevens in Tamino leest en schrijft gewoon in de XML bestanden, en Tamino zorgt er op een transparante wijze voor dat de data in de interne XML databank en de externe databanken consistent gehouden wordt.

De XML bestanden kunnen via XQL ondervraagd worden, waarbij de XQL query via een URL of via HTTP requests naar de *X-Port* gestuurd wordt, de ingebouwde Web server van Tamino. De resultaten van de query worden als XML teruggegeven, die via XSL omgevormd kan worden naar HTML. Het wijzigen van de XML gegevens gebeurt via een URL of via speciale HTTP requests die ook naar de X-Port gestuurd worden, en desgewenst kan men Java of COM-gebaseerde *Server Extensions* schrijven die ingrijpen in het parsings- en indexeringsproces. Men kan Tamino ook rechtstreeks aanspreken via een Java of COM API call, en de requests die in deze calls verstuurd worden zijn dezelfde als de HTTP requests die via de X-Port verwerkt worden. Naast de interne XML databank biedt Tamino ook nog een kleine *SQL Engine* aan, die toelaat van desgewenst nog een (niet te groot) aantal gegevens in relationele vorm op te slaan. Dit kan nuttig zijn wanneer een klassieke client/server applicatie, die enkel over ODBC of OLE-DB mogelijkheden beschikt, nog toegang wenst te kunnen krijgen tot een aantal gegevens in Tamino. Via een schema in de Data Map kan men er dan bvb. voor zorgen dat een gedeelte van de data in een XML bestand gemapt wordt naar een tabel in deze interne SQL databank.

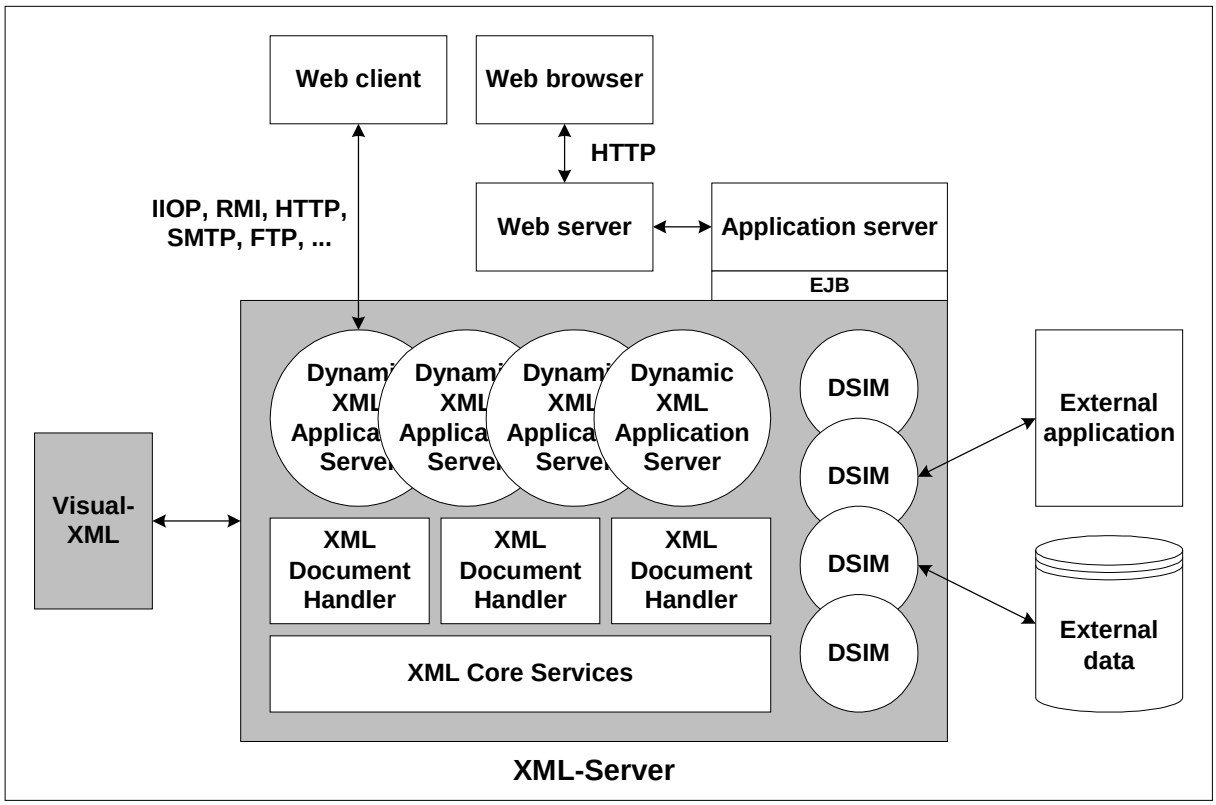
De nadruk binnen Tamino ligt duidelijk op de gedisciplineerde opslag van goedgedefinieerde XML gegevens, zodat het mogelijk is om deze gegevens op een betrouwbare manier te gebruiken, en op het veilig synchroon houden van deze XML gegevens met de achterliggende data bronnen. Momenteel is er enkel ondersteuning voorzien voor Adabas en andere relationele databanken, maar via Software AG's EntireX produkt is het mogelijk om ook ERP systemen zoals SAP, Peoplesoft en Baan aan te spreken. De Tamino benadering is zwaarder maar ook robuuster dan de eXcelon benadering, en leent zich het best tot toepassingen waar men nood heeft aan een Web-toegankelijke read-and-write XML cache bovenop een aantal bedrijfskritische data bronnen en applicaties. Een punt in Tamino's nadeel is wel dat het momenteel nog niet over echte visuele ontwikkeltools beschikt, op een *Manager* na die instaat voor het beheer van de data server en de definitie van Data Map schema's. Er worden wel gratis plug-ins aangeboden voor o.a. Microsoft's Visual C++ en Software AG's Bolero ontwikkelomgevingen.



Figuur 3: De architectuur van Software AG's Tamino.

Bluestone XML-Server

Bluestone, bekend om zijn Sapphire/Web Java-gebaseerde Web applicatie server, was eigenlijk de allereerste die de behoefte inzag aan een dynamische XML data server. Bij het uitbrengen van de XML-Server begin 1999 werd dan ook, naast de ondersteuning van XML als data type, meteen gekozen voor de ondersteuning van een aantal typische applicatie server functionaliteiten: sessie beheer, connectie pooling, ondersteuning van transacties, enz. Het belangrijkste onderdeel van de Bluestone XML-Server is de *Dynamic XML Application Server*, die kan beschouwd worden als een kleine applicatie server die zich enkel bezig houdt met het verlenen van XML diensten. Eén of meerdere Dynamic XML Application Servers staan in voor het beheer van de sessies met de client en voor de verwerking van de binnenkomende en buitengaande XML gegevens. Daartoe doen ze beroep op *Document Handlers*, Java klassen die de applicatie logica verbonden aan een bepaalde set XML gegevens encapsuleren. Om met deze XML gegevens te kunnen werken maken de Document Handlers gebruik van de *XML Core Services*, die o.a. DOM/SAX parsing en XQL query functionaliteiten aanbieden. Een Document Handler staat ook in voor de connectie met de achterliggende data bronnen via *Data Source Integration Modules* (DSIM). DSIMs bieden een uniforme interface aan om gegevens te lezen en weg te schrijven naar externe data bronnen. Standaard komt de Bluestone XML-Server met DSIMs voor ODBC, JDBC, LDAP, SMTP en FTP toegang, en zijn er DSIMs te koop voor toegang tot SAP R/3, PeopleSoft, Tuxedo, CICS, and MQSeries. Men kan ook zelf een DSIM ontwikkelen om toegang te krijgen tot een externe applicatie. De *Visual-XML* visuele ontwikkeltool helpt de ontwikkelaar bij het definiëren van de mapping tussen de XML gegevens en de externe data bron, en genereert vervolgens het skelet van de Java code voor de bijbehorende Document Handler.



Figuur 4: De architectuur van de Bluestone XML-Server.

De Dynamic XML Application Servers communiceren ofwel rechtstreeks met de buitenwereld, door via IIOP, RMI, HTTP, SMTP, FTP, ... met een externe Web client te praten, ofwel onrechtstreeks als een EJB binnenin een Web applicatie server. De omweg via een aparte applicatie server (waarbij Bluestone's eigen applicatie server Sapphire/Web natuurlijk de aanbevolen keuze is) is aangewezen als er nog bijkomende gegevensverwerking naast de verwerking van de XML gegevens dient te gebeuren, of als er scalability, load-balancing e.d. eisen gesteld worden die door de XML-Server alleen niet kunnen gedragen worden.

De architectuur van de XML-Server is gelijkaardig aan die van eXcelon en Tamino, maar er worden natuurlijk nergens permanent XML gegevens opgeslagen. De nadruk bij de XML-Server ligt volledig op het vlot verwerken van binnenkomende en buitengaande XML gegevens, en op het doorspelen van die gegevens naar de achterliggende data bronnen. Op die manier is mogelijk een zeer robuust en goed schaalbaar doorgeefluik te bouwen voor de uitwisseling van gegevens in XML formaat. Het enige zwakke punt van de XML-Server is wel de aanzienlijke leercurve, een gevolg van de inherente complexiteit van de server.

Is er nog een plaats voor relationele databanken?

Bij de rechtvaardiging van het gebruik van een XML data server als een data middleware laag zijn we eigenlijk stilzwijgend uitgegaan van de veronderstelling dat de onderliggende data bronnen niet in staat waren om zelf XML te genereren of te beheren. De bestaande relationele databank vendors zijn echter druk bezig met het uitbouwen van de XML mogelijkheden van hun produkten, zodat het in de nabije toekomst heel eenvoudig zal worden om ze te laten praten met een XML data server. Als de enige data bron waarmee men rekening dient te houden een bestaande relationele databank is dan kan men het eventueel zelfs zonder bijkomende XML data server stellen, en gewoon rechtstreeks de XML mogelijkheden van de databank gebruiken.

Bij de opslag van XML gegevens in een relationele databank kan men 3 strategiëen volgen: de XML elementen worden als aparte stukjes data opgeslagen in kolommen en tabellen, de XML gegevens worden als één groot document opgeslagen in een speciale kolom (BLOB of CLOB), of er wordt gekozen voor een hybriede opslag waarbij de XML gegevens worden opgeslagen als een document maar een gedeelte ook als elementen. Opslag van XML als elementen heeft als voordeel dat alle XML gegevens als normale databank gegevens beschikbaar zijn, en dus via normale SQL operaties kunnen beheerd worden. Het nadeel van deze werkwijze is de aanzienlijke overhead die gepaard gaat met het desassembleren en assembleren van de XML gegevens telkens die

gegevens in en uit de databank gehaald worden. Opslag van XML gegevens als documenten heeft als voordeel dat deze (des)assemblage stap niet telkens hoeft plaats te vinden, maar vermits de XML gegevens nu als één groot bestand zijn opgeslagen dient men bijkomende programma's te schrijven om dit XML bestand te manipuleren zodra het via normale SQL operaties uit de databank gehaald wordt. Dit is trager en minder eenvoudig dan het rechtstreeks beheren van de XML elementen via normale SQL operaties. Bij de hybriede vorm van opslag zoekt men een gulden middenweg, maar heeft men wel af te rekenen met de redundante opslag van een gedeelte van de XML gegevens. We bekijken even kort in hoeverre de belangrijkste relationele databank vendors nu reeds deze 3 mogelijkheden aanbieden.

Bij Oracle 8i beperkt de XML ondersteuning zich momenteel tot de Oracle XML Developer's Kit (XDK), een verzameling Java, C, C++ en PL/SQL componenten, tools en utilities die toelaten van XML binnen Oracle 8i te gebruiken. Via de XDK is het mogelijk om XML als elementen, als documenten en in hybriede vorm op te slaan. De XDK is enkel bedoeld voor ontwikkelaars die reeds vertrouwd zijn met XML, maar is nu al een commercieel produkt en wordt ook als dusdanig door Oracle ondersteund. Bedoeling is dat de XDK snel uitgroeit tot een volwaardig onderdeel van de Oracle8i databank en de Oracle Application Server.

Microsoft staat momenteel het minst ver van alle relationele databank vendors wat betreft XML integratie. Het enige wat al aangeboden wordt is een Technology Preview voor Microsoft SQL Server 7.0, die in combinatie met Internet Information Server 4.0 toelaat van SQL queries via de Web server naar de databank te sturen, waarna de resultaten in XML teruggestuurd worden. Er kan nog geen XML in de databank zelf opgeslagen worden, laat staan hierbinnen ook gebruikt worden. Microsoft kondigt wel een volledige XML ondersteuning aan in SQL Server 8.0 (release voorzien voor medio 2000), maar wat dit precies zal inhouden is nog verre van duidelijk.

IBM's DB2 Universal Database versie 6.1 bood reeds in het begin van 1999 XML ondersteuning aan en heeft momenteel de DB2 XML Extender in beta test. Met deze XML Extender is het mogelijk om XML als elementen, als documenten en in hybriede vorm op te slaan. Bij het opslaan van XML gegevens zorgen mapping definities voor het automatisch opbreken van het XML bestand in kolommen en tabellen, en bij het terugopvragen van de gegevens zorgen stored procedures via dezelfde mapping definities voor het opnieuw opbouwen van het XML bestand. Het is echter nog niet mogelijk om de XML gegevens zelf vanuit stored procedures te manipuleren.

Sybase is de jongste jaren qua functionaliteiten wat achteropgebleven ten opzichte van zijn voornaamste concurrenten, maar is momenteel aan een flinke inhaalbeweging bezig. Via de Java in SQL mogelijkheden van zowel de Adaptive Server Anywhere 6.0 als de pas uitgebrachte Adaptive Server Enterprise 12.0 is het nu ook mogelijk om XML als elementen, als documenten en in hybriede vorm op te slaan. De daartoe benodigde Java klassen worden al meegeleverd als onderdeel van het produkt, maar vormen nog geen geïntegreerde component van de databank kernel zelf. Informix tenslotte biedt enkel XML ondersteuning aan in de vorm van een Web DataBlade voor de Informix Internet Foundation.2000 databank, waardoor het mogelijk is om XML gegevens te genereren als resultaat van een SQL query. Er is wel al een beta versie aangekondigd van een XML DataPort, die het moet mogelijk maken van XML als elementen, als documenten en in hybriede vorm op te slaan.

Conclusie

Wanneer men bestaande applicaties via het Web wenst te integreren door aan gegevens integratie of genetwerkte gegevensuitwisseling te doen is XML een voor de hand liggende keuze als data formaat. Dankzij een XML data server kan men een geïntegreerd XML zicht krijgen op onderliggende heterogene data bronnen en applicaties, kan men deze XML gegevens opslaan en beheren of dynamisch laten genereren ter uitwisseling, en kan men er voor zorgen dat deze XML gegevens synchroon blijven met de onderliggende data bronnen. Dit alles maakt het nu veel eenvoudiger om bestaande applicaties via het Web te integreren, zowel binnen het bedrijf als daarbuiten. De eerste generatie XML data servers is duidelijk al zijn kinderschoenen ontgroeid, en we mogen dan ook verwachten dat ze binnenkort een onmisbaar onderdeel van uw Web data engineering toolkit zullen worden.

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 XML verzameling van standaarden:

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

XSL (E**x**tensible **S**tylesheet **L**anguage): een style sheet taal die zowel toelaat om XML gegevens om te vormen naar een andere structuur als er een bepaald uitzicht (bvb. HTML) aan te geven.

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

DTD (**D**ocument **T**ype **D**efinition): een formele specificatie van de verplichte structuur van XML gegevens, m.a.w. de elementen, de attributen van die elementen en de manier waarop die elementen mogen genest worden.

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:

www.w3c.org/xml

www.odi.com/excelon

www.softwareag.com/tamino

www.bluestone/xml

www.oracle.com/xml

msdn.microsoft.com/sqlserver/

www-4.ibm.com/software/data/db2/extenders/xmlext

www.sybase.com/produkts/databaseservers/ase

www.informix.com/xml/