VLAAMSE INGENIEURS KAMER

KATHOLIEKE HOGESCHOOL KEMPEN

XML-Databases

Editie 10 – jaargang 2002-2003

Daniël Artois

Inhoudsopgave

1. Voorwoord

Eerst zou ik willen uitzoeken wat XML eigenlijk is. Hiermee kan zowel de inhoud van een document als het document zelf bedoeld worden. Het is ook belangrijk om eens te kijken waar al die andere formaten en talen, die bij XML horen, voor gebruikt worden.

XML wordt vooral gebruikt om gelezen en geschreven te worden door een computer. XML bestaat om deze zelfde computers te helpen, want een groot probleem is, omdat ze, of toch zeker hun besturingssysteem, een andere taal spreken. XML is nog een poging om een standaard te creëren en enkele problemen op te lossen. Veel standaards zoals SGML bestonden al, en ze hebben veel ervaring opgebracht. De reden voor een standaard zoals XML is ook het internet. Ook hier werd al wat ervaring opgedaan met voorlopers zoals HTML. De bedoeling is dat XML voor alle soorten documenten de enige standaard wordt.

En als XML dan toch zo veel gebruikt wordt zal een XML Database voor het opslagen en beheren van XML, ook wel zinnig zijn. Omdat XML al een structurering is van de inhoud van een document, zou men het document zelf ook al een database kunnen noemen.

De populairste en stevigste database tot op vandaag, en dat zal zo nog wel een tijd duren, is de relationele database die uitgevonden werd door Edgar F. Codd. Codd stierf op 23 April 2003. Iedereen die database studeert moet de 12 regels van Codd leren. Bachman- en ER-diagrammen en normaliseren mogen ook niet onbreken. Het lijkt me zeer onwaarschijnlijk dat er iemand durft de verantwoordelijkheid te nemen om dat af te schaffen.

En toch zijn er zo veel voordelen aan XML dat we er niet meer kunnen naast kijken. XML zou een natuurlijkere manier zijn om de data te zien. En de computers zijn nu toch krachtiger dan vroeger en kunnen de last van al die tags, extra bestanden en dubbele data best aan. Men kan met XML data doorsturen die niet alleen platform onafhankelijk is, maar zelfs niet bij een bepaald programma hoort.

Ik denk dat dan alleen de vraag overblijft, of die XML ook in een database moet, en of die database dan zo verschillend moet zijn, met wat we tot nu gebruikten. Uiteindelijk is het ook niet altijd het beste produkt, dat de markt verovert.

Met deze scriptie heb ik de bedoeling mezelf, en de geinteresseerde lezer, te introduceren in de wereld van XML Databases.

2. Wat is XML? [1]

Xml of Extensible Markup Language is een manier om data door te geven tussen computerprogrammas. Xml is een meta-taal. Dit is een taal die een andere taal omschrijft en dus een afspraak die gemaakt wordt over de manier van communiceren. Xml is een basis om afspraken te maken over de manier waarop men de data zal omschrijven en interpreteren. Met xml kan men nieuwe talen definieëren. XML zelf is al een eenvoudigere versie van SGML en ook platform onafhankelijk. Eenzelfde XML is dus bruikbaar op verschillende computers en besturingssystemen.
Een van de redenen waarom XML zo populair is , is omdat de basis heel gemakkelijk aan te leren is. Alleen is het zo dat men als gebruiker met alleen maar XML, niet veel kan doen. XML wordt normaal gelezen en geschreven door computers. Een belangrijke eigenschap van XML is dat het een scheiding maakt tussen inhoud en presentatie. Bij XML hoort een DTD of een schema om de inhoud te controleren en een stylesheet voor als we de inhoud willen tonen. Daarbij komt ook nog een programma om de data in het document weg te schrijven. XML maakt dus gretig gebruik van de nu goedkoper geworden computer-geheugens, processoren en discs. De belangrijkste reden om XML te gebruiken is dat het een universele standaard is en niet omdat het beter is dan andere reeds bestaande technologieën.

1.1. De structuur van een XML-document.

XML bestaat uit 4 basis componenten, tags, elementen, attributen en een hierarchie. Hieronder volgt een korte beschrijving van elk van deze componenten.

1.1.1. Tags

Essentiele onderdelen van xml-bestanden zijn tags. Door voor en na elke blok van data een tag te plaatsen weten we waar die blok data begint en eindigt. In de tag schrijven we dan de naam die we geven aan die block data. Een voornaam zou men bijvoorbeeld kunnen omschrijven als volgt: ‘<voornaam> Daniël </voornaam>’ Een woord tussen ‘<‘ en ‘>’ is een begintag en tussen ‘</’ en ‘>’ is een eindtag. Het is mogelijk van een lege tag te maken. die wordt dan op volgende manier gemaakt: ‘<voornaam/>’. Het beschrijven van tags gebeurd in een ander bestand. Hiervoor zijn 2 mogelijkheden een DTD (Document Type Definition). of een Schema. Om de data te tonen moet men een derde bestand aanmaken dat van het type XSL (eXtensible Stylesheet Language) of CSS (Cascading Style Sheets). Commentaar in de code wordt geschreven als ‘<!--Commentaar-->’ Bovenaan in een xml-document staat de declaratie ‘<?xml version="1.0"?>’
Het is belangrijk te weten dat XML uitbreidbaar is en een dataveld of tag enkel wordt toegevoegd als het nodig is. Niet een leeg veld zal aantonen dat de data niet bestaat, maar het weglaten van het element.

[1] http://www.w3.org/XML/

1.1.2. Elementen en Children

De tekst tussen 2 tags noemen we een element. Elementen kunnen sub-elementen hebben. Het element persoon heeft als mogelijke sub-elementen een naam en een voornaam. Sub-elementen noemt men ook nog children.

```
<?xml version="1.0"?>
<adresboek>
    <persoon>
        <naam>Modaal</naam>
        <voornaam>Jan</voornaam>
    </persoon>
</adresboek>
```

1.1.3. Attributen

Elementen kunnen attributen hebben zoals in het volgende voorbeeld thuis, stamkroeg, ouders of werk.
<adres plaats=”thuis”> ??? </adres> en <adres plaats=”werk”> ??? </adres>

1.1.4. Een well formed document.

Een document kan gecontroleerd worden of het voldoet aan de regels om Well Formed te zijn.
Deze regels zijn

- één root-element hebben
- alle andere children moeten elementen zijn van het root-element.
- alle elementen moeten correcte paren zijn met een begin- en een eind-tag.
- de ellement-namen in de begin- en eind-tag moeten dezelfde zijn
- attribuut-namen mogen maar 1 keer gebruikt worden in hetzelfde element.

```
<?xml version="1.0"?>
<adresboek>
<persoon>
<naam>Modaal</naam>
<voornaam>Jan</voornaam>
<adres plaats=”thuis”>
        <straat>Kerkweg</straat>
        <busnr/>
        <huisnr>7</huisnr>
        <postnr>0123</postnr>
        <woonplaats>
          Plattegem
        </woonplaats>
</adres>
</persoon>
</adresboek>
```

1.2. Document Type Definition of DTD [2]

Een document dat well formed is kan ook nog gecontroleerd worden of het wel valid is tegenover een DTD of een XML-schema. DTD’s zijn niet vereist maar worden enkel gebruikt als men wil weten of een XML-document valid is. Met een valid document bedoelt men dat we de data in het document kunnen beschrijven op een formele manier, de data kunnen communiceren met anderen, beperkingen opleggen op de inhoud van de elementen, de attributen en dat we eventueel standaard waarden kunnen voorzien. Een DTD beschrijft de regels waaraan een XML-document zich moet houden en hoe het moet gelezen worden. Het bevat specificaties voor elk element en zijn attributen, welke waarden ze kunnen hebben en welke elementen kunnen genest worden in andere elementen. Deze specificaties kunnen zich in een XML-document of in een appart DTD-document bevinden. Een XML-document kan maar 1 DTD of schema hebben. Het gebruik van één DTD voor verschillende documenten levert een standaard tussen deze documenten. Behalve voor parsers en browsers kan een DTD ook gebruikt worden door programmas die XML aanmaken. Voor het schrijven van een DTD wordt gewoonlijk een programma gebruikt.De taal waarin DTD’s geschreven worden is zeer complex en

[2] http://www.w3.org/TR/REC-xml

dezelfde als welke gebruikt wordt voor de DTD's van SGML. Er is dus voldoende ondersteuning en praktische ervaring.

1.2.1. De structuur van een DTD.

Bovenaan in een XML wordt een DOCTYPE tag geplaatst die aangeeft welke DTD de XML valideert. <!DOCTYPE adresboek SYSTEM "adresboek.dtd">
Het volgende is enkel maar een voorbeeld en toont niet alle mogelijkheden van DTD's.

Een element kan bestaan uit PCDATA, andere tags, ANY = beiden of EMPTY. De andere tags gescheiden door een komma maakt ze verplicht. Met een verticale streep tussen zijn ze optioneel of een keuze.

```
<!ELEMENT adresboek (persoon)*>
<!ELEMENT persoon (naam  , voornaam ,
        adres)*>
<!ELEMENT naam (#PCDATA)>
<!ELEMENT voornaam (#PCDATA)>
<!ELEMENT adres (woonplaats | postnr |
        huisnr | busnr | straat)*>
<!ATTLIST adres plaats CDATA #IMPLIED >
<!ELEMENT straat (#PCDATA)>
<!ELEMENT busnr (#PCDATA)>
<!ELEMENT huisnr (#PCDATA)>
<!ELEMENT postnr (#PCDATA)>
<!ELEMENT woonplaats (#PCDATA)>
```

Quantifiers: Een * duidt aan dat iets meerdere keren kan voorkomen. Bij + komt iets 1 of meer keer voor en is het dus verplicht. Een ? betekent dat iets 0 of 1 keer kan voorkomen. Met !ATTLIST worden attributen omschreven. CDATA staat voor character data , een lijst gescheiden door | duidt op een keuzemogelijkheid tussen deze waarden. #IMPLIED betekend dat het attribuut niet verplicht is.

1.3. XML-namespaces

Deze standaard beschrijft het opdelen van namen voor elementen of attributen in ruimtes. Namen hebben enkel hun juiste betekenis binnen hun ruimte. Deze naamruimtes worden bovenaan in het document gedefinieerd. Hierdoor kan men in documenten gebruik maken van verschillende naamruimtes en op een unieke manier naar een document refereren.
In onderstaand voorbeeld behoort elke element-naam met prefix *adr:* tot de namespace *http://www.mijnvoorbeeld.be/adressen* . Als dan dit document of een onderdeel ervan, samenkomt met een ander document dat dezelfde namen gebruikt zal men toch een onderscheid kunnen maken omdat ze tot een andere namesoace behoren.

```
<?xml version="1.0"?>

< adresboek>
< persoon>
< naam>Modaal</naam>
< voornaam>Jan</ voornaam>
< adres plaats="thuis">
   < adr:straat xmlns: adr:=http://www.mijnvoorbeeld.be/adressen>>
      Kerkweg
      </ adr:straat>
   < adr:busnr/>
   < adr:huisnr>7</ adr:huisnr>
   < adr:postnr>0123</ adr:postnr>
   < adr:woonplaats>
         Plattegem
   </ adr:woonplaats>
</adres>
</persoon>
</adresboek>
```

1.4. XML-Schemas

XML schema is een W3C specificatie uit 2001 om de inhoud van een XML document te valideren. Schemas hebben mogelijkheden die DTD's niet hebben zoals het bepalen van datatypes: integer, string, date, decimal en het gebruik van regular expressions om nog juistere definities te maken. Een schema kan extra beperkingen opleggen zoals de limieten van een element-waarde. Dit zijn eigenschappen die ook nodig zijn om data uit een XML document beter te kunnen converteren naar een database.
Schemas zijn geschreven in de XML syntax. Bovenaan krijgen schemas de namespace http://www.w3.org/2001/XMLSchema. Gewoonlijk gebruikt men de prefix xsd maar dit hoeft niet. Elk element of attribuut die deze prefix gebruikt zal geassocieerd worden met de namespace en gezien worden als code voor een XMLSchema.
Elementen kunnen qualified of unqualified zijn. Qualified betekend dat ze een prefix hebben en behoren tot een namespace. Hebben elementen geen namespace, en zijn ze dus unqualified , dan kunnen we ze toch met een schema koppellen met XSI:nonamespaceSchemaLocation = "naamvanschema.xsd > aan de namespace toe te voegen en te laten verwijzen naar het schema zelf.
Elk schema heeft een rootelement <schema>. In het rootelement kunnen 3 subelements voor komen: element, complextype en simpeltype.
Als men een Schema moet maken kan men eerst gaan kijken in een registry [3] of er niet al iets gelijkaardigs bestaat. Onderstaande is een voorbeeld van een schema voor een eerder gebruikte adresboek.xml.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified">
        <xsd:element name="adres">
          <xsd:complexType>
               <xsd:sequence>
                    <xsd:element ref="straat"/>
                    <xsd:element ref="busnr"/>
                    <xsd:element ref="huisnr"/>
                    <xsd:element ref="postnr"/>
                    <xsd:element ref="woonplaats"/>
               </xsd:sequence>
               <xsd:attribute name="plaats" type="xsd:string"
        use="required"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="adresboek">
          <xsd:complexType>
               <xsd:sequence>
                    <xsd:element ref="persoon"/>
               </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="busnr">
          <xsd:complexType/>
        </xsd:element>
        <xsd:element name="huisnr" type="xsd:byte"/>
        <xsd:element name="naam" type="xsd:string"/>
        <xsd:element name="persoon">
          <xsd:complexType>
               <xsd:sequence>
                    <xsd:element ref="naam"/>
                    <xsd:element ref="voornaam"/>
                    <xsd:element ref="adres"/>
               </xsd:sequence>
```

[3] http://www.xml.org/xml/registry_searchresults.jsp

```
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="postnr" type="xsd:byte"/>
        <xsd:element name="straat" type="xsd:string"/>
        <xsd:element name="voornaam" type="xsd:string"/>
        <xsd:element name="woonplaats" type="xsd:string"/>
</xsd:schema>
```

1.5. XPath [4]

XPath is een standaard 'path expression syntax' van de W3C. XPath is een syntax die een gedeelte van een document kan aanduiden en gebruikt het principe van een directory path zoals bij een besturingssysteem. Alleen worden hier de directories en sub-directories vervangen door elementen, child-elementen, enz... XPath is één van de onmisbare onderdelen van XSL.
Enkele korte voorbeelden. *adresboek/persoon/adres/straat*
Begint het path met een */* dan bedoelt men een absolute path.
Een *//* betekent alle elementen en sub-elementen.
Met viekante haken *[]* kan men op een bepaald element zoeken.
Met een @ wordt een attribuut aangeduid.
//adres[@plaats = "thuis"] = alles adressen die als attribuutwaarde voor plaats “thuis” hebben.

1.6. XQL [5]

Deze taal vertoont nogal wat gelijkenissen met XPath maar wordt gezien als een querie-taal om vragen te stellen aan een document zoals aan een database. Als Querie-taal zal XQuery waarschijnlijk wel de standaard worden en XPath blijft zeker bestaan omdat het essentieel is voor XSL. Soms wordt XQL een uitbreiding van de XSL-syntax genoemt. Ik denk dat er gewoon geen plaats meer is voor nog een extra taal. XQL is ook geen standaard van de W3C.

1.7. XLink

Net zoals het <A> element bij HTML heeft ook XML een mogelijkheid om een link te maken naar een ander document. Om het even welk element kan een link zijn of een verband leggen naar een ander document. Het element moet hiervoor wel het attribuut XML:LINK bevatten dat dan de waarde SIMPLE of EXTENDED meekrijgt. SIMPLE betekend een verband met één ander document. EXTENDED is een multidirectionele link met verschillende documenten. Xlink bevat een groot aantal standaard attributen waardoor het veel krachtiger is dan de links in HTML.
Door de links in het gelinkte document te volgen kan men data in verschillende documenten uitlezen.
XLINK zegt niet wat een programma of browser moet doen met een link. Een link is enkel maar een markering.

1.8. XPointer

XPointer maakt het mogelijk een bepaalde locatie in een document te bepalen en aan te wijzen. Dit kan bijvoorbeeld dienen om naar een specifieke plaats in een XML-document te verwijzen. Terwijl bij XPath de resolutie tot op het niveau van structurele elementen

[4] http://www.w3.org/TR/xpath20/
[5] http://www.ibiblio.org/xql/

gaat, kan men met XPointer willekeurige punten of gebieden in een XMLdocument aanwijzen.
XPoint is zeer nuttig om te verwijzen tussen documenten met XLink.

1.9. XFragments [6]

XFragment is een aankomende standaard om handelingen te kunnen doen met een gedeelte van een document zonder de rest van het document te moeten beheren en zonder de context van het fragment te verliezen. Hiermee zou men ook snel tot ergens midden in het document kunnen gaan zonder het ganse document te moeten overlopen. Onderdelen van een document zoals chapters, citaten enz... kunnen in andere teksten gebruikt worden zonder hun context en de binding met hun bron te verliezen.
Het probleem is dat een onderdeel van een tekst niet voldoende informatie met zich meedraagt. De bedoeling van XFragments is een middel voorzien om deze informatie mee te leveren zodat XML-elementen zoals boeken, chapters, paragrafen, tabellen enz... kunnen uitgewisseld worden.
Om een fragment te markeren maakt men een 'Fragment Context Specification'. Het stuk tekst dat uit het document wordt gehaald noemt men 'fragment body'. Het stuk tekst waarin de 'fragment body' staat is de 'fragment entity'. Soms wordt in de 'fragment entity' ook de context specificatie geplaatst.

1.10. XInclude [7]

XInclude kan gebruikt worden om een XML-document in een ander XML-document te plaatsen.

1.11. Stylesheets.

XML geeft een structuur aan de data, waardoor we de computer kunnen laten weten wat er staat. Dat geeft als voordeel dat we het maken van een geschikte weergave van de data, met een programma kunnen automatiseren. Hievoor hebben we extra gegevens nodig die we in een stylesheet terugvinden. Stylesheets zijn een hulpmiddel waarmee een document onder verschillende vormen en op verschillende hard-ware gepresenteerd kan worden. Omdat XML de inhoud scheidt van de presentatie heeft men om de tekst van een XML-document afzonderlijk te tonen altijd een stylesheet nodig. Het voordeel van het gebruiken van stylesheets is dat men de inhoud niet vervuild met stijlelementen. Dezelfde inhoud kan dan ook toegepast worden op verschillende stylesheets. Een stylesheet werkt onafhankelijk van een DTD of schema. Een eerste vorm zijn de CSS , die we al kennen van HTML. Een tweede vorm is XSL die in XML geschreven zijn en meer mogelijkheden bieden.

[6] http://www.w3.org/TR/xml-fragment
[7] http://www.w3.org/TR/xinclude/

1.11.1. Cascading Style Sheets of CSS.

```
<style
        type="text/
        css">
H1 {color: blue}
</style>
```

Bovenstaande style zal alles tussen tag <H1> en </H1> blauw kleuren.

In de oudere versies van HTML zoals in versie 2.0 waren bijna geen voorzieningen om een pagina vorm te geven. De bedoeling is altijd geweest om de vorm te bepalen met een andere codering die zich in de browser, het document zelf, of in een bijgevoegd document bevind. Er kunnen dus meerdere style-sheets invloed hebben op hetzelfde document waarbij conflicten worden vermeden door middel van een systeem van prioriteiten, overerving en hiërarchie. Vandaar komt de naam CSS die een afkorting is van Cascading Style Sheets.
CSS moet een aantal functionaliteiten missen die we wel terugvinden bij XSL zoals:

- Men kan niet bepalen waar een onderdeel van een document moet worden weergegeven.
- CSS heeft geen variabelen en kan dus ook geen berekeningen doen of tekst genereren zoals paginanummers.
- Men kan niet specifiëren hoe XML documenten moeten overgebracht worden naar verschillende toepassingen (bijvoorbeeld ten behoeve van een printer).
- CSS kent wel hiërarchische relaties, maar geen relaties tussen verwanten: het is onmogelijk om een CSS stijlblad te schrijven dat iedere andere paragraaf in het vet zou zetten.

1.11.2. Extensible Stylesheet Language of XSL.[8]

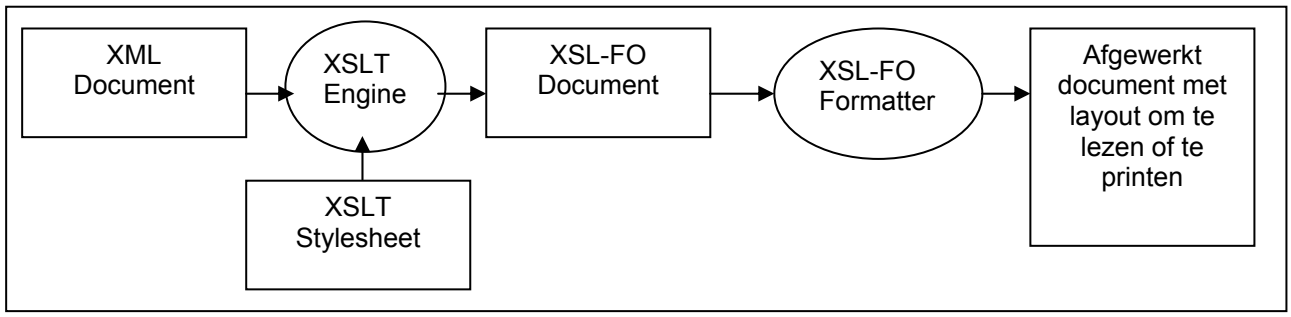
XSL is de XML versie van CSS stylesheets maar eigenlijk ook veel meer. Met XSL kan men een XML omzetten naar een andere XML-vorm. XSL is dan ook bruikbaar om XML zo te vormen dat het aangepast is om in een database op te slaan.
XSL is vooral bedoelt voor complexe formatteringen zoals:

- wanneer de inhoud van een document op verschillende plaatsen moet afgebeeld worden (tekst van een hoofding die ook voorkomt in een automatisch gegenereerde inhoudstafel)
- voor het sorteren en filtreren van XML gegevens
- voor het formatteren van XML gegevens op basis van de datawaarde (zoals het afbeelden van negatieve getallen in het rood)
- voor het invoegen van volledig nieuwe elementen in het doelbestand of het verwijderen van elementen

In feite bestaat XSL uit drie componenten. De eerste is XPath om de locatie van de elementen in de documenten te bepalen. Daarnaast bestaat er XSLT (XSL Transformaties), voor transformaties. Deze worden gebruikt om een XML hiërarchie te transformeren naar een andere hiërarchie op basis van een aantal regels en eventueel iets toe te voegen. De twee hiërarchieën zijn totaal gescheiden. Hoewel het er oorspronkelijk niet voor bedoeld was wordt XSLT ook onafhankelijk van XSL gebruikt om XML documenten te transformeren naar andere formaten zoals naar HTML. Als derde bestaat er XSL-FO (XSL Formatting Objects) dat beschrijft hoe formattering moet worden uitgevoerd. XSL-FO is het moeilijkste om te maken (en te leren). Een XSL-FO bestaat gewoonlijk al en men moet enkel met een gemakkelijker te maken XSL een juiste XML maken. Een voorbeeld van XSL-FO is waar het gebruikt word om XML om te zetten naar PDF.

[8] http://www.w3.org/Style/XSL/

Gezien dat een groot gedeelte van het werk van programmeurs het omzetten is van data van het ene formaat naar het andere, zou XSL wel eens één van de belangrijkste programmeertalen van de toekomst kunnen worden.



2. De geschiedenis van XML.

Al van vroeger, toen alles naar een drukker gedaan werd, hebben schrijvers hun documenten voorzien van extra aantekeningen over hoe de gedrukte versie zou moeten zijn.Die notas of markup hadden ook hun eigen taal. Eigenlijk hebben sommige letters ook een markup om de lezer duidelijk te maken hoe hij het moet lezen. Deze vindt men niet terug in het alfabet maar zijn toch zo belangrijk dat ze werden voorzien bij de ASCII karakters..ASCII betekent American Standard Code for Information Interchange. Computers kunnen enkel met getallen werken, ASCII code is de numerieke presentatie van een karakter. Er zijn 32 karakters die niet geprint worden. Deze controle karakters zijn voorzien om iets te besturen. De meesten hiervan zijn niet gebruikt geweest, ofwel waren ze zeker niet hetzelfde op alle systemen.Dit maakt dat het zelfs nu nog niet mogelijk is een gewone tekst die op het ene besturingssysteem gemaakt is te lezen op een ander besturingssysteem omdat ze allemaal een ander teken gebruiken om het einde van een lijn aan te duiden. In programmeertalen ging men nog veel verder. In de talen C en afgeleiden gebruikt men ‘{‘ en ‘}’ om het begin en het einde van een blok code aan te duiden. In Pascal gebruikt men 'begin' en 'end' enz... Een mogelijke oplossing die altijd en overal zou werken kwam er met de Markup Language SGML. Niet omdat deze de juiste tekens brengt maar markup gemakkelijk om te zetten is naar een andere markup. SGML betekent Standard Generalized Mark-up Language. SGML is zeer flexible maar te ingewikkeld coor de meeste toepassingen. Parsers, browsers en applicaties die met SGML moeten werken zijn te groot en te moeilijk om te maken. Daarom ontwikkelden een groep experts uit de industrie en de academische wereld, een vereenvoudigde versie die XML noemt.
HTML is ook een vorm van SGML maar te gelimiteerd om data in op te slaan en houdt te veel rekening met de opmaak van de gegevens. Het ontwikkelen van XML begon in 1996. Dit betekent niet dat SGML zal verdwijnen. Dit is niet zo voor HTML. Deze zal vervangen worden door XHTML wat wel voldoet aan de regels van XML.

1838	Morse	
1874	Baudot	
1896	Hollerith's code (ponskaart)	
1964	EBCDIC	
1968	ASCII	
1969	GML	IBM
1992	Unicode	
1978	SGML	ANSI start
1986	SGML	ISO 8879
1989	HTML	
1998	XML	W3C
2000	XHTML	opvolger van HTML die voldoet aan de regels van XML

3. Het editeren en publiceren van XML.

IDC schrijft in het rapport *Worldwide XML Development Tools Software Forecast, 2002 - 2006* dat de markt voor xml-ontwikkelgereedschappen van 2002 tot 2006 jaarlijks met 41,5 procent zou groeien tot een wereldwijde omzet van 395 miljoen dollar in 2006.

3.1. Cooktop[9]

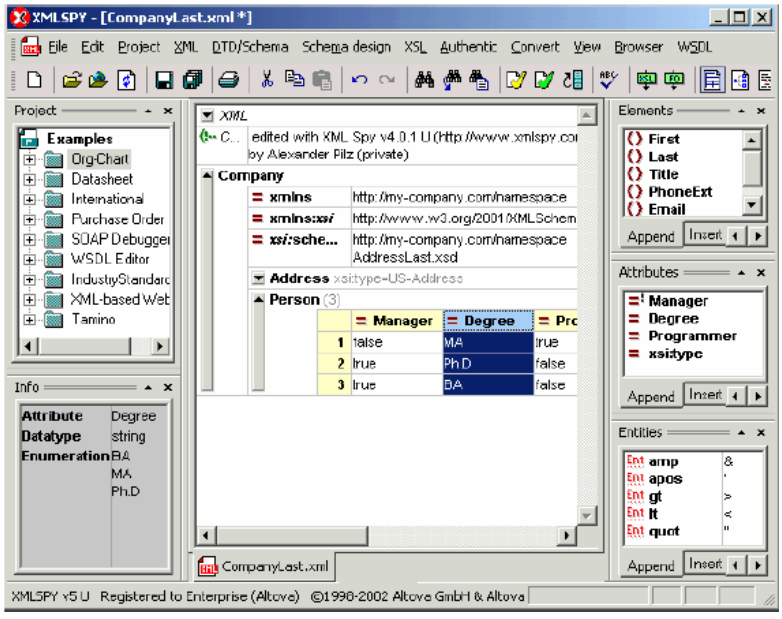
Een gratis editor onder windows voor XML, DTD en XSLT documenten. Cooktop geeft volledige toegang tot de code en is zeer geschikt om snel documenten aan te maken of te controleren.

3.2. XMetal[10] Developer van Corel *(voorheen Softquad)*

Dit is een plug-in voor Microsoft Visual Studio .NET die toelaat snel interfaces te bouwen voor de creatie van XML inhoud. De interfaces die hiermee gemaakt zijn kunnen door gebruikers ingevuld worden zonder dat zij zien dat ze eigenlijk een XML-document aan het maken zijn.
Behalve dit kan XMetal ook dat wat men terugvindt in de meeste andere XML-editors. Het pakket bestaat uit 4 onderdelen Author, ActiveX, Developer en Central. XMetal is dan ook één van de meest uitgebreide pakketten.
Minder aangenaam voor de gevorderde XML-developer is dat het produkt de code te veel verbergt en weinig of geen vrijheid laat aan de gebruiker. Het editeren van bestaande XML documenten waarbij men de juiste DTD's of schema's moet importeren in een project is te moeilijk.

3.3. XML Spy[11] van Altova

XMLSpy is momenteel de meest proffesionele en overzichtelijkste XML ontwikkelomgeving. Ze biedt steun voor alle XML technologieën zoals: SOAP, XML Schema, XSL, WSDL, DTD, enz.... XMLSpy gebruikt een overzichtelijke structuur die de XML als een boom weergeeft. Men kan ook XSLT debuggen, WSDL maken, er is integratie met de Native XML database Tamino en Webdav. Een aantal documenten zoals DTD en schemas kan men automatisch laten genereren. Schemas kunnen gemaakt worden op elke database die toegangkelijk is met ADO of ODBC. Het is mogelijk om van een XML document



[9] http://www.xmlcooktop.com
[10] http://www.corel.com/xmetal
[11] http://www.xmlspy.com

classes te genereren in Java of C++. En men kan een HTML omzetten naar XML met bijhorend XSD schema, en XSLT stylesheet. Het aanmaken van XML documenten kan gebeuren door gebruik te maken van de reeds ingebouwde templates zoals van DocBook, NewsML, P3P enz....

3.4. XML Content Management Systems

Content Management Systems zijn gemaakt voor het opslagen van text in de vorm van fragmenten zoals, procedures, chapters, samenvattingen en bevatten ook metadata, gegevens over auteur, revisie datums, document nummers enz... Content Management Systems hebben ook voorzieningen zoals editors en versie- en workflow controle. Content Management Systemen gebruiken gewoonlijk een Native XML Database die verborgen blijft voor de gebruiker.
De naam Content Management System bestaat al langer voor software die het beheer van een website kan doen. Gezien de mogelijke combinatie tussen website's en XML zullen Content Management Systems in de toekomst veel gebruikte produkten worden. Alleen is het nog niet duidelijk wat het zal worden.

3.5. Cocoon [12]

Cocoon is een XML publishing framework voor de Apache webserver. Een functonaliteit van Cocoon is dat men XML en XSL kan samenbrengen en hen een door de client aangevraagde HTML laten maken. Van XML tot het einddocument worden een aantal SAX-Streams na mekaar in een pipeline geplaatst. Deze configuratie bepaald men in het bestand sitemap.xmap. Het starten van het proces gebeurd met een HTTP request.
Cocoon laat toe gestructureerde, onderhoudbare XMLapplicaties te maken.
Andere toepassingen zijn web pages die kunnen geschreven worden in XSP (eXtensible Server Pages) een scripting taal die toelaat Java tussen XML te zetten. Voor XSP bestaat er een SQL tag-library (ESQL) om data uit een relationele database te halen met JDBC. Het SQL statement kan om het even welke SQL zijn en hoeft geen Select te zijn. ESQL is een wrapper rond de functionaliteiten van de JDBC-driver en kan niet meer of minder dan die driver.

3.6. WebDAV [13]

WebDAV is een industriestandaard en betekent Web-based Distributed Authoring and Versioning. Het is een set van HTTP uitbreidingen die gebruikers helpen om bestanden op servers te beheren. In tegenstelling tot HTTP dat maar toegang geeft tot 1 document laat WebDAV toe verschillende documenten te zien als collecties. WebDAV wordt gewoonlijk geintegreerd met andere desktop applicaties. Connecties met de server kunnen bijvoorbeeld geopend worden in Internet Explorer als een Web Folder. Vanaf dan heeft men toegang tot deze folder zoals tot elke andere folder van het systeem. Bestanden kunnen dan met drag-en-drop verplaatst worden. WebDAV geeft de mogelijkheid om documenten die door meerdere gebruikers tegelijk gebruikt kunnen worden te locken, Hierdoor kan men er voor zorgen dat er maar één gebruiker tegelijk de data kan aanpassen. Alle documenten kunnen voorzien worden van properties voor metadata. Elke property bestaat uit een name/value-paar. WebDAV is een belangrijke platform-onafhankelijke techniek die door veel XML-software zoals de meeste databases en editors gebruikt wordt.

[12] http://cocoon.apache.org/
[13] http://www.webdav.org/

4. Een XML-document om data bij te houden.

Het verschil tussen een document en een database is dat documenten niet gestructureerde data bevatten en dat de data in een database wel gestructureerd is. Een rapport dat uit een database wordt gegeneerd zal dan toch weer een document genoemd worden. Het verschil is dus niet alléén het al dan niet gestructureerd zijn maar ook of het een computer, programmeur of ander persoon is die het document verder moet gebruiken. Het verschil wordt vooral duidelijk als men data moet aanpassen in de gegevens van een database of in documenten. Een personeelsnummer die een cijfer langer wordt in een relationele database is gemakkelijk aanpasbaar. Dit zelfde in een groot aantal documenten aanpassen is onbegonnen werk of men moet van in het begin de teksten zo structureren dat het personeelsnummer gemakkelijk is terug te vinden met een computerprogramma. Gewoonlijk is het zo dat als bepaalde teksten een definitie krijgen en te onderscheiden zijn, we dan van data spreken. In XML geven we elk stuk tekst dat we willen omschrijven een naam door de tekst tussen tags te plaatsen.
Een XML-document is een collectie data en kan gezien worden als een simpele vorm van een database. Elk document bevat data en beschrijft ook de structuur en het type van de data. Nadelen van XML zijn dat door de tags de bestanden zo groot zijn en dat de toegang tot de data traag gaat omdat het ganse document moet doorlopen en gefilterd worden. Dat laatste noemen we parsing.
Vergeleken met een database is een XML-document geen efficiente manier om data op te slaan. Er is geen index, geen beveiliging, geen transacties, geen data integriteit tussen verschillende documenten, geen multi-user access, geen triggers, en er zijn geen queries over verschillende documenten mogelijk.
Voor de controle over de input van data in XML kan men DTD's en Schema's gebruiken en er bestaan ook Query-talen zoals XQuery en XPath. Programmas maken die documenten aanpassen kan via data-binding met SAX, DOM of JDOM. Dit zijn allemaal externe onderdelen die aan elk document kunnen toegevoegd worden. Het document blijft echter ook bestaan en oncontroleerbaar te behandelen met of zonder deze hulpmiddelen. Het is het document zelf dat de standaard is waar een aantal al dan niet standaard hulmiddelen kunnen aan toegevoegd worden.
Het gebruik van een los document als opslag voor data is enkel mogelijk als er maar weinig data is bij te houden en als er maar 1 gebruiker tegelijkertijd is.

1. Een voorbeeld van zo een type database is het bijhouden van eigenschappen en instellingen van een programma in een .ini bestand. XML heeft hier veel meer mogelijkheden en is met SAX, DOM of JDOM gemakkelijker te onderhouden dan data die gescheiden is door comma's, regel-einden of een '=' teken. XML heeft ook het voordeel dat men data-elementen kan nesten zodat elk element sub-elementen kan hebben.
2. - Een ander voorbeeld waar een XML-document handig is voor dataopslag, zijn eenvoudige lijsten voor persoonlijk gebruik, zoals adressen, verzamelingen van muziek, boeken enz... Voor deze toepassingen zijn bijna geen platform onafhankelijke oplossingen beschikbaar. Men moet een keuze maken tussen de desktop-databases en spreadscheets die wel zeer handig en veelzijdig zijn maar een groot gebrek hebben aan compatibiliteit. Als er enkel data werd bijgehouden zouden de compatibiliteitsproblemen nog niet zo groot zijn maar geen van de produkten beperkt zich tot de data. De meeste produkten kunnen wel een export naar XML geven maar dikwijls is die XML zo ingewikkeld dat het de data weer onbruikbaar maakt. Tot nu toe is het zo dat de verzamelingen die we met onze software willen beheren langer

meegaan dan de software. Het is dus noodzakelijk van de data op te slaan in merk- en platform onafhankelijke bestanden zoals XML.

4.1. Data-georienteerde XML-documenten [14]

Het is noodzakelijk om XML-documenten in te delen in 2 categorieën omdat ze een verschillende behandeling vragen.
Data-georienteerde XML is gestructureerde data in XML. Om gegevens gemakkelijk te kunnen lezen, schrijven en aan te passen verwachten computers gestructureerde gegevens. Dit vergemakkelijkt ook de te schrijven programmacode en versnelt de werking van het programma. De elementen hebben allemaal dezelfde structuur en geen bepaalde volgorde. Deze documenten worden gewoonlijk gemaakt en gelezen door een computerprogramma of een database. Als dit door mensen zou gebeuren, zou men geen controle meer hebben over de inhoud.
Een vorm van data-georienteerde zijn dynamische web sites zoals voor catalogussen en adreslijsten die gemaakt zijn van gekende datastructuren. Of documenten die gebruikt worden data transport zoals bestellingen, persoonsgegevens enz..
Hoe deze data wordt opgeslagen is afhankelijk van hoe goed ze gestructureerd is en hoe dikwijls ze moet aangepast worden. Gewoonlijk zal men hiervoor een relationele database kiezen. Het is dan ook dikwijls beter van dit rechtstreeks te doen in plaats van eerst XML te maken en die dan weer te moeten omzetten naar tabellen. Data die in een XML formaat moet staan kan ook vanuit een relationele database gemaakt worden.

4.2. Document-georienteerde XML-documenten [14]

Een Document-georienteerde XML heeft een structuur die kan veranderen en heeft dikwijls geen DTD of Schema. Het is dan ook moeilijk om het aanpassen van data in het document te automatiseren. Deze documenten worden gewoonlijk door mensen behandeld. Dit kan gebeuren met een programma dat een andere presentatie van het document geeft. Lezen en opzoeken van onderdelen van het document kan wel met de computer gebeuren. In tegenstelling tot data-georienteerde documenten is de volgorde van de elementen en de structuur van het document hier wel belangrijk.
Voorbeelden zijn hoofdstukken uit boeken, verschillende stappen in gebruiksaanwijzingen, procedures, websites enz... Deze documenten splitsen naar data die in een database kan opgeslagen worden zal moeilijke programmas en veel proccessortijd vragen. Complexe documenten zoals deze kunnen beter ogeslagen worden in een Native XML Database.

4.3. Metadata

Metadata is data die andere data omschrijft en gewoonlijk ook begrijpbaar is voor een computer. Het geeft informatie over karakteristieken van objecten, documenten, afbeeldingen of geluid. Zonder metadata is data al veel minder waard. Metadata is vooral belangrijk om de data langer te kunnen blijven gebruiken. Het geeft ook informatie over hoe de data verzamelt en verwerkt werd. Om metadata universeel begrijpbaar te maken bestaan er standaards zoals Dublin Core.

[14] http://www.infosup.nl/Technologie/XML.asp

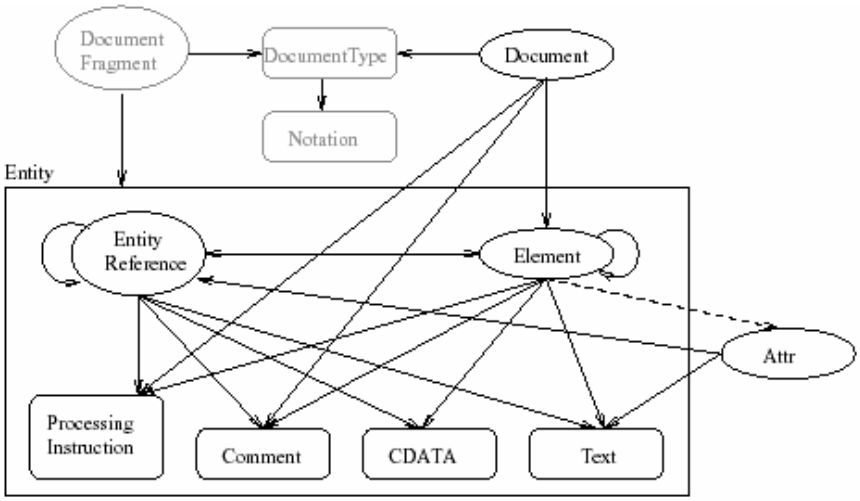
4.4. Resource Description Framework (RDF) [15]

Een manier om de metadata van een XML-document voor te stellen en bij te houden is RDF. Daarin bestaan ook standaards zoals Dublin Core. Dublin Core [16] [17] metadata is een set van 15 elementen waarmee internetbronnen kunnen omschreven worden. Deze set werd samengesteld door vertegenwoordigers uit de bibliotheek-, archief- en museumwereld, netwerk en computerwetenschappen.
Om metadata op zinvolle wijze te verspreiden op het web zodat het geïndexeerd kan worden door de grotere web-indexing robots, wordt metadata bij voorkeur in HTML-bestanden gecodeerd. Het standaard bestand voor de toekomst zou RDF/XML zijn. Omdat RDF een volledige beschrijving geeft van een document, komt het ook van pas om documenten te ordenen en op te zoeken in een database.

5. Middleware, API's en Data Binding

Als niet het ganse document, maar enkel de data uit het document, moet bijgehouden worden, dan zal het nodig zijn om ergens een omzetting te doen. Hiervoor bestaan er middelen in de vorm van reeds geschreven code, die dit vergemakkelijken, en standaardiseren. Zo een bibliotheek van code, noemt men een API, of Application Programming Interface, en men vindt ze terug bij de meeste programmeertalen.
Produkten die de omzetting doen noemt men middleware.
Data Binding is het koppelen van XML-documenten aan objecten die speciaal gemaakt zijn om de data van deze documenten te behandelen. Dit laat toe dat de data kan beheerd worden op een manier die eigen is aan de gekozen programmeertaal. De programmeur hoeft dan niet te weten hoe de data is opgeslagen en kan totaal onafhakelijk van dedatabase werken. Het programma kan dan ook snel aangepast worden aan een andere database.

5.1. Document Object Model (DOM) [18] [30]



Handelingen op een document dat aan de DOM regels voldoet kunnen geautomatiseerd worden. DOM is een data model dat een document voorstelt en een interface om met dat model te werken. De interface is dat wat door programmeurs gebruikt wordt en wat we de API noemen. Het data-model bevat alle elementen, entities, commentaren, processing instructions en niet XML secties zoals <![CDATA[.....]]> en hun relaties tot mekaar.

[15] http://www.w3.org/RDF/
[16] http://www.kb.nl/coop/donor/project-nl-index.html?/coop/donor/rapporten/DC-gebruikersgids.html
[17] http://dublincore.org/documents/dces/
[18] http://www.w3.org/DOM/

5.2. SAX

SAX is een JAVA API die gewoonlijk gebruikt wordt met data-georieënteerde XML. De SAX parser overloopt het document en stuurt events naar geregistreerden (listeners) voor elk XML onderdeel dat hij tegenkomt. SAX werkt enkel op een vaste data layout. Als de layout van de XML veranderd moet ook het programma aangepast worden. Omdat SAX enkel events stuurt, kan men de events voor data die men niet nodig heeft, negeren, wat leidt tot snellere programmas. Het document kan behandelt worden als een data-stream en moet niet volledig in het geheugen genomen worden.

5.3. CASTOR en JAXB

Castor is een data-binding produkt dat Java objecten kan opvullen vanuit, en wegschrijven naar XML-documenten, een relationele database, of LDAP een hierarchische database. Castor kan automatisch Java Beans mappen. Voor mappings naar andere objecten of databases is er een XML-based mapping language voorzien die één-op-één en één-op-veel relaties aankan.
Marschalling een Java object betekend het omzetten naar XML om het op te slaan of te verzenden. Unmarschalling is het weer omzetten van een XML-document naar een Java-object. Het voordeel hiervan is dat werken met objecten veel sneller gaat dan SAX en minder geheugen gebruikt dan DOM. Castor en JAXB, (Sun's Java Architecture for XML Binding) zijn ook frameworks om classen te maken op basis van een DTD of Schema. Met deze classes kunnen dan objecten van XMLdocumenten gemaakt worden. De Java-programmeur moet dan enkel de code schrijven die de XML gebruikt en geen code die data wegschrijft, inleest of valideert. Validatie-code komt uit de DTD of schema en zit niet in de java-code zodat het programma beter onderhoudbaar wordt.

5.4. Java API for XML Processing (JAXP)

JAXP laat toe om XMLdocumenten te parsen en aan te passen onafhankelijk van een bepaalde XML processor implementatie. JAXP heeft ook de mogelijkheid om via plug-ins te wisselen tussen verschillende XML processor implementaties. In JAXP zitten de industrie standaards DOM en SAX

6. Query-talen en API's voor databases

Er is momenteel nog geen standaard query-taal voor XML documenten maar de behoefte aan een krachtige query-taal is zo groot dat het ongetwijfeld geen jaren meer zal duren. Net zoals SQL de relationele database populair maakte zou een XML-query taal de volledige doorbraak kunnen zijn voor de XML-database. Het data-model van XML wijkt te ver af van een relationele database zodat SQL hier niet bruikbaar is. Bovendien moet bij XML steeds rekening worden gehouden met het feit dat de structuur van het document kan veranderen. Er bestaan reeds een aantal zoekmechanismen die gewoonlijk een uitbreiding zijn van bestaande talen zoals XPath wat eigenlijk een zogenaamde 'path expression syntax' is. XPath kan worden gezien als de meest gebruikte query-taal voor XML.
Een standaard API voor een XML database zoals ODBC, ADO of JDBS is er ook nog niet. Dit kan ook moeilijk zolang er nog geen standaard query taal is. Nochtans zijn er al

enkelen die op de eerste rij staan om er aan te beginnen. De belangrijkste API is XML:DB[19] en recentelijk XQJ die zich enkel tot Java richt.

6.1. XML:DB

De organisatie XML:DB is opgericht door dbXML Group L.L.C, SMB GmbH en de OpenHealth Care Group en word gesteund door een groeiende lijst van organisaties. Het doel van XML:DB is het ontwikkelen van technische specificaties voor het beheren van data in XML Databases en deze te verdelen met een Open Source License. Hun doel is ook om hiervoor samen te werken met W3C maar zich te concentreren op XML Databases.
Momenteel staan er op hun website 3 projecten.

6.1.1. XML Database API

De merk onafhankelijke XML:DB is een standaard API voor XML-database services.
De API is gebouwd rond de 4 basis concepten: Drivers, Collections, Resources en Services. Drivers: worden geleverd door de fabrikant van de database. Collections: zijn containers die andere collecties en resources bevatten. Met resources wordt bedoeld een XML of een Binary Large Object. De mogelijkheid is er om later andere soorten van resources toe te voegen. Services kunnen gevraagd worden om taken aan te vragen.

6.1.2. XUpdate XML Update Language

Xupdate is een standaard voorgesteld voor XML:DB die het aanpassen van een gedeelte van een document zou mogelijk maken. Een XUpdate gebruikt de XML syntax en is een well-formed document. Het maakt gebruik van de taal XPath en XSLT. In het document komen elementen uit volgende namespace http://www.xmldb.org/xupdate. En een element kan één van de volgende xupdate:modifications bevatten.

- xupdate:insert-before
- xupdate:insert-after
- xupdate:append
- xupdate:update
- xupdate:remove
- xupdate:rename
- xupdate:variable
- xupdate:value-of
- xupdate:if

Voorbeeld:

```
<xupdate:append select="/addresses" child="last()">
  <xupdate:element name="address">
    <town>San Francisco</town>
  </xupdate:element>
</xupdate:append>
```

6.1.3. SiXDML - Simple XML Manipulation Language

SIXDML is gemaakt om een standaard te zijn voor XML Databases zoals er een SQL en verschillende API's bestaan (ODBC, JDBC, enz...) voor relationele database. Alhoewel er reeds verschillende talen bestaan is er nog geen enkele standaard voor de functies DELETE, REPLACE en INSERT die door SiXDML wel zouden ingevuld worden. SiXDML bestaat uit 2 delen: Een data definitie en manipulatie taal op basis van SQL en een API gebaseerd op XML:DB API.

[19] http://www.xmldb.org/

6.2.XQuery [20]

De Working Draft voor Xquery werd door de W3C gepubliceerd op 3 mei 2003. XQuery is ontstaan uit de taal Quilt die op zijn beurt stukken gebruikte van talen zoals Xpath 1.0, XQL, XML-QL, SQL en OQL.
XQuery geeft ontwikkelaars de mogelijkheid om data op te halen en het resultaat eventueel te structureren als een XML document. Een van de grote voordelen van XQuery is dat men resultaten van verschillende documenten kan samenvoegen. XQuery is een combinatie van XPath 2.0 en het nesten van op SQL gelijkende uitdrukkingen die we FLWR statements noemen. FLWR is de afkorting van FOR-LET-WHERE-RETURN.

```
FlwrExpr     ::= (ForClause | letClause)+ whereClause? returnClause
ForClause    ::= 'FOR' Variable 'IN Expr (',' Variable IN Expr)*
LetClause    ::= 'LET' Variable ':=' Expr (',' Variable := Expr)*
WhereClause  ::= 'WHERE' Expr
ReturnClause ::= 'RETURN' Expr
```

The FOR doorloopt het document en zoekt naar het meegegeven pad. In LET worden variabelen opgevult. Met WHERE worden voorwaarden of een filter meegegeven en RETURN is hoe het resultaat moet weergegeven worden. Het resultaat van een Xquery kan weer gebruikt worden door een andere Xquery. Dit betekent dat men net zoals bij SQL met sub-queries kan werken.
XQueryX is de representatie van een XQuery als een XML-document.

6.3.XQuery API for Java (XQJ).

XQJ is een aanvraag van IBM en Oracle voor een standaard Java API die een programma toelaat een Xquery en XQuerX te doen naar een XML database. Deze API zou dan zijn zoals JDBC voor SQL en de relationele databases.

6.4.SQL/XML [21]

SQL/XML is een toevoeging van een datatype en een aantal functies aan SQL3. Deze query-taal laat toe het resultaat van een SQL weer te geven als XML.
SQL/XML wordt vooral gebruikt door Oracle, IBM en Microsoft. Deze drie databases zijn ook toegangkelijk met Xquery. Gezien dat XQuery meer steun krijgt in de Open Source wereld zal SQL/XML wel de laatste zijn om een standaard te worden.

[20] http://www.w3.org/TR/xquery/
[21] http://otn.oracle.com/oramag/oracle/03-may/o33xml.html

7. XML Databases

Volgens analist Ted Friedman van Gartner hebben XML databases nog maar weinig aanvaarding gevonden in de markt. Nog tot 2005 verwacht Gartner een erg beperkte groei. Marktonderzoeker IDC Research daarentegen voorspelt een grote groei en een markt van 600 miljoen dollar tegen 2006.
Eigenlijk gaat het hier over 2 onderdelen. De data en het programma waarmee we de data beheren. Welk programma we gebruiken hangt ook af van hoe we de data opslaan. De 2 gaan gewoonlijk samen omdat het programma moet zeker zijn dat de data niet verandert op een manier die fout is voor dat systeem.
Dus alle vragen en opdrachten voor de data moeten via dat ene programma gebeuren dat we DBMS of Database Management System noemen.
Omdat XML data is die gestructureerd in een bestand zit hebben we alleen maar een DBMS nodig. Gewoonlijk wordt het document toch nog opgeslagen in een database zodat de DBMS alle macht heeft over het document. Hoe minder onderdelen het systeem heeft hoe stabieler het is en hoe minder onderhoud er zal moeten gebeuren.

7.1. Embedded XML Databases

Door het grote success van relationele database systemen zoals Oracle en IBM's DB2 leeft het idee dat dit de enige vorm van database is. Deze systemen zijn groot en duur, hebben hun eigen bedieningssoftware en hebben gewoonlijk extra personeel nodig, zoals een database adminstrator voor hun opvolging en onderhoud.
Ondertussen is er een snel groeiende markt voor embedded databases ontstaan. Deze bestaan uit bibliotheken van programeercode die door softwareontwikkelaars kunnen gebruikt worden. Het is dan voor de eindgebruiker niet meer nodig om de data te beheren. Alle manipulaties gebeuren met de software. Omdat de softwareontwikkelaar het programma toch moet schrijven en de eindgebruiker niet rechtstreeks aan de data moet kunnen is een embedded database voldoende. Dit maakt installatie en onderhoud veel eenvoudiger en verhoogd de snelheid van het programma. De developer levert de inspanning om de database te integreren en bespaart hiermee vele gebruikers het kontakt met de database.Het opslaan van de gegevens in XML-formaat maakt dat de data toch nog bruikbaar is voor andere systemen. De developer krijgt met XML een formaat dat nauw aansluit tussen het object-georieënteerde applicatie-data-model en de database. Minder onderdelen en behandelingen van de data maken de software betrouwbaarder.

7.1.1. Berkeley DB XML [22]

De Berkeley DB XML is een embedded database gebouwd op de Berkeley DB en geschreven in C++. Deze db kan XML-documenten opslaan, ze indexeren en geeft de mogelijkheid queries te maken in Xpath. De database heeft geen mogelijkheid om iets in een document te veranderen. Er bestaan API's voor C/C++, Java, Perl, Python en TCL. De documenten worden er rechtstreeks opgeslagen in containers, waar al dan niet een gemeenschappelijk schema bij hoort, zonder dat ze eerst moeten bewerkt worden.De documenten blijven er altijd zoals ze zijn. Spaties, volgorde en layout van het document blijven behouden. De db vraagt wel veel geheugen en processorkracht omdat alles in het geheugen gebeurd.

[22] http://www.sleepycat.com

Ze kan wel rechtstreeks de elementen aanspreken vanaf en index in plaats van het ganse document te moeten doorlopen. Men kan zelf bepalen op welke data een index moet komen om sneller te kunnen zoeken. Een index heeft 4 karakteristieken:
1. Path Type: Hier moet men opgeven hoe diep genest de elementen of attributen liggen waarop men moet kunnen zoeken.
2. Node Type: hier bepaald men mee of er gezocht wordt op elementen of attributen.
3. Key Type: 'equality' die nodig is voor de functie '=' in XPath, of 'presence' dat nodig is voor de functie 'contains()'.
4. Syntax Type: Bepaalt het type van de te indexeren data zoals 'string' of 'number'.

7.2. Native XML-Databases. (NXD)

De term werd voor het eerst gebruikt door Tamino van Software AG. Hierdoor komt het dat er nooit een technische definitie is opgesteld.
Er is wel een definitie opgesteld door XML:DB[23] maar deze blijft zeer vaag.

1) Een NXD heeft een logisch model voor een XML-Document. Als minimum moet het model bevaten: Elementen, attributen, PCDATA en een document volgorde. Voorbeelden van zulke modellen zijn XPath, Dom en SAX
2) Een NXD moet een XML document als zijn basis unit hebben zoals een relationele database een rij in een tabel als basis heeft.
3) Een NXD moet geen bepaalde manier van dataopslag hebben en kan gebouwd worden op een bestaande database.

Native XML Databases zijn bedoeld om ganse XML-Documenten op te slaan en enkel bepaalde gedeeltes van het document uit te lezen. Voor het ogenblik zijn er nog weinig of geen mogelijkgheden om in deze documenten iets te veranderen zonder het ganse document te moeten ophalen en te herschrijven. Een standaard hiervoor is in ontwikkeling in de vorm van XUpdate van XML:DB.
De voornaamste redenen om een native XML database te gebruiken zijn:
1. Het behouden van het datamodel uit het document.
2. Het kunnen terug op halen van het document zonder dat er iets aan veranderd is.

Data die uit een XML gehaald wordt om opgeslagen te worden in een database houdt geen rekening met zijn formaat. Het gegeven CDATA, External Entity of PCDATA gaat verloren. Omdat native XML-databases het ganse document opslaan blijven deze gegevens bewaart.
3. Vragen kunnen stellen over de inhoud van 1 of meerdere documenten tegelijkertijd met een query-taal.
4. Het behandelen van de documenten sneller te laten verlopen. Documenten en inhoud worden in de database geindexeerd.
5. Ze werken onmiddelijk zonder dat er veel configuratie nodig is.
6. Men heeft geen DTD of schema nodig.
7. Ze hebben zoekmogelijkheden op teksten.

Nadelen zijn dat het zoeken in verschillende documenten traag gaat en dat het resultaat gewoonlijk in XML is.
Voor de toekomst mag nog verwacht worden:
8. verbeteringen aan de Query-talen en een stevigere standaard.
9. Concurrency Locking van XML-fragmenten in plaats van zoals nu locks op het ganse document.

[23] http://www.xmldb.org/

Native XML Databases zijn niet bedoeld om de bestaande databases te vervangen. Ze zijn enkel een gereedschap om te gebruiken met XML-documenten. Als men een groot aantal XML-documenten moet beheren die niet alleen uit data bestaan is een NXD de juiste oplossing.

7.2.1. Tamino [24]

Tamino is de oudste en bekenste Native XML-Database. Ze maakt het mixen van data uit verschillende bronnen mogelijk en geeft toegang tot opgeslagen documenten op verschillende manieren. XML-Schema's zijn niet nodig en mogen later nog toegevoegd worden. Schema's mogen ook beperkt blijven tot alleen maar een gedeelte van het document en kunnen later nog aangepast worden, ook voor bestaande data.
Tamino heeft Open Database Connectivity, Unicode Compliance, HTTP communicatie en mogelijkheden om niet XML-data te behandelen. Inbegrepen zijn ook tools zoals schema-editors, java ondersteuning met DOM2, JDOM en SAX2 , ondersteuning voor EJB's, en X-Application. De belangrijkste toegang tot de database is via HTTP waarbij men gebruik maakt van het component X_Port. Er is ondersteuning voor de instructies GET, PUT, DELETE en HEAD.
Een Tamino database bestaat uit verschillende collecties. Deze collecties zijn containers die documenten samen groeperen. Elk document hoort bij één bepaalde collectie. Bij elke collectie hoort ook een XML-schema. In de schema's krijgen de doctypes een beschrijving die eigen is aan Tamino.Het doctype wordt bepaald door het root-element. Tamino kan behalve XML-documenten ook andere objecten opslaan zoals foto's, klank en documenten in andere formaten. Deze worden geplaatst onder een speciaal doctype dat nonXML noemt.
Tamino bevat ook nog een uitbreiding op Xpath die X-Query noemt. Deze X-Query is niet hetzelfde als de Xquery van de W3C die nieuwer is.
X-Plorer is is het navigatie-gereedschap om de database en zijn documenten te bekijken en te onderhouden. Hiermee kunnen alle schema's en documenten in een tree-view gezien worden. Vanuit X-Plorer kunnen ook andere applicaties gestart worden zoals de schema-editor.

*7.2.1.1.*X-Application

X-Application is een op Java gebaseerde open source framework. X-Application laat toe om JSP-Tags (Java Server Pages) die toegang hebben tot de database in te bouwen in HTML. Deze tags kunnen als Add-On geinstalleerd worden in Macromedia Dreamweaver zodat de creatie van de JSP's van daaruit kan gebeuren. Het grootste voordeel van zo een Java-tag-bibliotheek is dat men bouwt op een gratis beschikbare technologie.

7.2.1.2.WebDAV

Via WebDAV kan Tamino zich integreren in MS Windows maar ook met een XML-editor zoals Altova's XML SPY. De WebDAV server van Tamino geeft ook extra informatie zoals last-modified, length en content-type.

7.2.1.3.X-Tension

X-Tension laat toe zelf extenties te maken in C, C++, Java of COM/DCOM die bepalen hoe een element, attribuut of boom moet opgeslagen worden en zorgt ook voor de integratie met EntireX.
EntireX, een ander produkt van Software AG, is een hulpmiddel voor het realiseren van informatie-uitwisseling met behulp van XML. Het programma analyseert de inhoud van berichten en bepaalt vervolgens automatisch voor welke persoon of afdeling de berichten

[24] http://www.softwareag.com/tamino/

bestemd zijn. Het bestandsformaat wordt eventueel aangepast aan het systeem van de ontvanger.

7.2.1.4.X-Node

X-Node zorgt voor de toegang tot andere relationele databases. Deze data kan dan geintegreerd worden in XML-Documenten die door XML-View geleverd worden. Hiermee is het mogelijk om data die veel veranderd niet op te slaan in het XML-document maar telkens te lezen uit een relationele database.

7.2.2. eXist [25]

eXist is een Open-Source Native XML-database systeem dat nauw aansluit bij bestaande XML ontwikkelingstools zoals Cocoon[26] van Apache. eXist in combinatie met Cocoon geeft een uitgebreide set van mogelijkheden zoals XSP en output naar HTML, PDF, SVG (Scalable Vector Graphics), WAP (Wireless Access Protocol) en nog andere formaten.
eXist heeft alle basis functionaliteiten van een native XML database als ook een aantal geavanceerde zoals zoeken op sleutelwoorden, op woorden waarvan de positie in een document kort bij mekaar ligt en zoeken op regular expression patterns. eXist laat toe om documenten op te slaan zonder schema of DTD. Dit geeft de mogelijkheid om documenten van een verschillend type op dezelfde manier te ondervragen. De database is ook aansluitbaar aan een relationele database. Het behandelen van de data blijft dan gewoon hetzelfde. Hiervoor moet men enkel de JDBC connectie parameters in het configuratiebestand veranderen. eXist is geschreven in Java en kan werken als een standalone db in een servlet of embeddeb in een applicatie. Stand-alone, embedded of in een servlet container, de 3 manieren zijn thread-safe en laten concurrent operations door verschillende gebruikers toe.

7.2.2.1.De XML:DB API

Embedding van de database in een applicatie zonder external server is ondersteund met de XML:DB-API driver. De XML:DB API is in Java geschreven en een poging om een standaard interface te maken die ook al gebruikt wordt door Xindice[27] een andere Native XML database. eXist ondersteund enkel XML als resource voor XML:DB.
Ook het samenwerken met Cocoon werkt volledig volgens deze API. Cocoon zal dan eXist gebruiken om de XML uit de database te lezen met XSP's. Extensible Server Pages zijn XML-documenten met Java code zoals JSP Java code in HTML is.
Andere manieren om de database te benaderen zijn via HTTP en XML-RPC of Remote Procedure Call, SOAP en WEB-DAV.

7.2.2.2.De Soap Interface

Een SOAP interface om een database aan te spreken is interessant omdat er al veel tools beschikbaar zijn die de WSDL (Web Service Description Language) zelf maken. SOAP ondersteund ook user defined types zodat de code korter en leesbaarder wordt. eXist gebruikt de AXIS SOAP toolkit[28] van Apache die in een servlet draait. Er zijn 2 webservices, de eerste voor queries en de andere voor het toevoegen, verwijderen en tonen van documenten.

[25] http://exist.sourceforge.net/
[26] http://cocoon.apache.org/
[27] http://xml.apache.org/xindice/
[28] http://ws.apache.org/axis/

7.2.2.3.Logic Sheets voor XSP

XSP is een vorm van dynamische webpagina's. De code bestaat uit XML met Java. Hier wordt herbruikbare code in 'Logic Sheets' geschreven in plaats van in 'Tag Libraries' zoals bij JSP. Dit beperkt de hoeveelheid Java Code in en XSP. EXist voor XSP levert een logic sheet die gebaseerd is op de XML:DB API en enkele tags definieerd met de belangrijkste functies. Om deze te gebruiken moet men enkel bovenaan in de XSP de correcte namespace toevoegen.

7.2.2.4.XPath uitbreidingen

Binnen in de database zitten de documenten in collections. Het zoeken gebeurd met XPath-expressies die kunnen uitgevoerd worden op een gedeelte of op de volledige collectie.
EXist is gemaakt voor document-georienteerde XML waar een node veel tekst heeft en XPath is hiervoor minder geschikt als querytaal. Daarom heeft men een aantal uitbreidingen aan de taal toegevoegd zoals text-operators en –functies, en een extra index om woorden bij te houden. Omdat de database kan gezien worden als een verzameling collecties van documenten zijn er de 2 functies 'collection()' en 'document()' die als parameter de collectie of het document krijgen waarop de Xpath moet op uitgevoerd worden. Voor het zoeken naar woorden binnen een tekst bestaan er expressies zoals 'near' met de parameters die men zoekt en hoe ver ze uit mekaar mogen liggen.
//section[near(.,'XML database',50)] geeft alle sections die de woorden 'XML' en 'database' bevatten in de juiste volgorde en waar ze maximaal 50 woorden uit mekaar gelegen zijn. Als de juiste volgorde en afstand van de woorden niet belangrijk zijn kan men operators gebruiken zoals '&=' en '|='.
LINE |= 'XML database' geeft alle lijnen waar 'XML' of 'database' in voorkomen.
LINE &= 'XML database' geeft alle lijnen waar beide woorden in voorkomen.
Beide operators kunnen ook vervangen worden door 'Match-all' en 'Match-any'. Verder wordt ook de Regular Expressions syntax ondersteund.

7.2.2.5.Indexen

Xpath werkt normaal op het doorlopen van het ganse document van voor naar achter. Dit werkt vlot zolang het systeem het ganse document in het geheugen heeft. Als het document zo groot is dat een gedeelte op een disk wordt geschreven zal een query heel veel disk I/O vragen en traag worden. Het is dus nodig van indexen te maken zodat niet alle data moet gelezen en geevalueerd worden. Om queries snel te laten verlopen maakt eXist een index-schema op alle nodes van het document zoals elementen, text en attributen. Dit maakt het voor de database ook mogelijk om relaties te vinden tussen de verschillende nodes. Dank zij deze indexen is het bij het evalueren van een query niet nodig om het ganse document van boven naar onder te overlopen. Het is mogelijk een restrictie te zetten op het volledig automatisch indexeren van de volledige tekst en enkel een gedeelte van een document te gebruiken.
eXist maakt hiervoor een boom die men een k-ary-tree noemt waarbij k gelijk is aan het maximum aantal knooppunten van elementen. Dit betekent dat elke knoop naar in de boom naar evenveel knooppunten zou verwijzen. Elk element dat naar minder elementen verwijst zal dus een knoop krijgen met een aantal lege knopen die men virtueel noemt. Het herberekenen van het aantal knooppunten gebeurd enkel per niveau van de boom. Het is dus niet zo dat als een boom op een diep niveau veel knooppunten heeft hij volledig zou moeten hermaakt worden om overal virtuele knooppunten toe te voegen.
EXist gebruikt 4 indexen:

1. dom.dbx verzamelt DOM knooppunten in een paged file en legt de link naar de knooppunten in de XML.
2. collections.dbx beheert de collections hierarchie.
3. elements.dbx is een index op elementen en attributen.
4. words.dbx houdt bij hoe dikwijls woorden voorkomen en wordt gebruikt bij het zoeken naar teksten.

7.2.2.6.XUpdate.

Op dit ogenblik ondersteund eXist nog niet het updaten van één enkel element. Hiervoor zouden er nog aanpassingen moeten gebeuren aan het bestaande indexing systeem. De gebruikte taal voor de updates zou XUpdate zijn, een standaard voorgesteld voor XML:DB.

7.2.3. Xindice [29]

Xindice is een Native XML database van de Apache Software Foundation geschreven in Java. Xindice is gemaakt om een groot aantal kleine XML-documenten op te slaan. Elementen en attribuutwaarden kunnen geindexeerd worden en documenten kunnen kleiner gemaakt worden om plaats te besparen. Documenten worden opgeslagen in collecties en kunnen ondervraagd worden met X-Path. Collecties kunnen onderdeel zijn van X-Path zodat men de vraag naar verschillende documenten tegelijk kan stellen. Xindice ondersteund XUpdate van XML:DB om documenten aan te passen. Xindice heeft ook nog een taal Xlinks waarmee men inhoud van een XML document kan vervangen of invoegen bij een ondervraging. Xindice ondersteund 3 API's, XML:DB, CORBA en XML-RPC en de programmeertalen PHP, Perl en Applescript. Xindice voorziet ook XMLObjects die toelaten de functionaliteit van de server nog uit te breiden.

[29] http://xml.apache.org/xindice/

7.3. XML Enabled en Relationele Databases. [30]

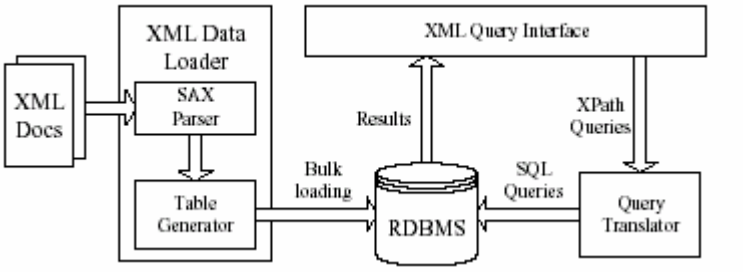
XML-Enabled databases zijn er in overvloed. Eigenlijk heeft elke bestaande database er wel voor gezorgd dat ze iets met XML te maken heeft. De meesten kunnen wel een import van XML doen. Het resultaat van een query kan gewoonlijk ook wel in XML worden weergegeven.
Relationele databases zijn de meest voorkomende. Zij maken gebruik van verschillende tabellen om vaste types van data in op te slaan. De kracht van een relationele database schuilt in de mogelijkheid van 2 tabellen te koppelen via een kolom die een gemeenschappelijk type data bevat. De meeste data bestaat uit een veel op veel relatie waarvoor een relationele database het best geschikt is. Omdat de layout van de tabellen onveranderlijk is en de SQL-queries er speciaal voor geschreven zijn krijgt men een zeer stabiel systeem. Al deze goede eigenschappen maken dat het waard is om data-georienteerde XML-documenten om te zetten naar relationele tabellen en ze daarin bij te houden. Op deze manier maken we ook gebruik van de jarenlange ervaring die is opgedaan met deze databases en de kennis die op de arbeidsmarkt beschikbaar is.
De 3 groten uit de database wereld Oracle, IBM DB2 en MS SQLServer gaan hierop verder maar leveren een bijna nieuw produkt onder dezelfde naam. Hierdoor moeten ze het bestaande niet opgeven en kunnen ze hun klanten behouden. Het zijn dan ook moeilijk te begrijpen en ingewikkelde produkten geworden.

7.3.1. XParent [31]

XParent is een methode om een XML-document management systeem te bouwen op een RDBMS. Hierbij wordt de data opgeslagen in relationele tabellen zonder rekening te houden met DTD's of XML-Schemas.
XParent gebruikt het data model van XPath om XML-documenten voor te stellen in een boom-structuur en spiegelt de data naar de volgende vier tabellen:

1 LabelPath(ID, Len, Path) = alle label-paths die hier een uniek ID krijgen.
2 DataPath(Pid, Cid) = parent-child relatie.
3 Element(PathID, Did, Ordinal)
4 Data(PathID, Did, Ordinal, Value)
In de tabellen Element en Data is PathID een ID uit LabelPath en is Did een ID uit de tabel DataPath.

Een systeem zoals XParent bestaat uit 3 onderdelen: een Data Loader die de data opsplitst naar tabellen, een Query Interface die zowel visueel als in code een query aankan en een Query Translator die de query, die bijvoorbeeld in Xpath kan zijn en deze omzet naar SQL.



7.3.2. IBM's DB2

DB2 ondersteunt XML in de XML Extender, de Text Extender en het Web Services Framework (DB2 WORF).

[30] http://www.surfnet.nl/innovatie/surfworks/xml/xml-databases.pdf
[31] http://www.cs.ust.hk/~fervvac/files/ICDE02-XParent-Final.pdf

XML Extender laat toe om queries te maken die een combinatie zijn van XPath en XSLT. De XML functies in DB2 worden ondersteunt met SQL/XML. Dit maakt het mogelijk om een relationele database te benaderen alsof het een XML-document was. XML Extender kan XML-documenten in de db opslaan in “XML Columns” of in “XML collections”. XML Columns slaan volledige documenten op als VARCHAR’s, CLOB’s of als bestanden die XCMLVARCHAR, XMLCLOB of XMLFILE gebruiken.
Met DAD (Data Access Definition) kan men bepalen welke elementen en/of attributen geindexeerd moeten worden in de site tables. Deze DAD-documenten bevatten de document-id en de indexen en zijn zelf ook geindexeerd. XML kolommen blijven gesynchroniseerd met de Site Tables. XML-collections mappen niet XML-data aan een XML-document volgens een DAD-document. Er zijn 2 soorten mappings, SQL-mapping en RDB-node mapping. SQL-mapping is een template based language waarin men gebruik maakt van een SELECT-statement. Deze wordt enkel gebruikt om data te verplaatsen van een database naar een XML-document. RDB-node mapping is een object-relationele mapping en kan gebruikt worden om data te verplaatsen van en naar een database. Er is een visueele tool voorzien om DAD-documenten te mappen naar tabellen en kolommen.
Applicatie gebruiken stored procedures om de XML Extender op te dragen data te schrijven of te lezen. De XML Extender beheert DAD-documenten en DTD’s in zijn eigen tabellen. XML Extender kan XML-documenten sturen, en ontvangen van MQSeries Message Queues, kan validatie doen met gebruik van schema’s, DTD’s, XSLT’s, XML-documenten copieeren tussen files en databases en er waarden uithalen. De DB2 Text Extender bezit zoekmethodes per zin of paragraaf. Deze kunnen gebruikt worden als een documetn opgeslagen is in één enkele kolom.
DB2 WORF laat toe een Web Service te maken met DADX-documenten. DADX is een uitbreiding op DAD en beschrijft hoe een webservice communiceerd met een database. De functionaliteiten hier zijn opslaan, en opvragen van documenten met XML Extender, SQL’s uitvoeren en het oproepen van stored procedures. DB2 WORF kan ook WSDL-documenten (Web Services Description Language) genereren van DADX-documenten.
De volgende stap van IBM wordt Xperanto die met XQuery zal werken.

7.3.3. Oracle 9i XDB [32]

XDB ondersteund XML-enabled en native opslag van XML-data. Het vervaagt de grenzen tussen relationelegegevens en XML door SQL zo uit te breiden dat men geen verschil meer ziet tussen de twee. Centraal staat het XMLType als datatype. Dit is een object-type dat een XML-document kan opslaan als CLOB of object-relationeel. Net zoals een objecttype kan een XMLType gebruikt worden als een datatype in een kolom van een tabel of een view. Dit betekend dat alle data kan gezien worden als XML. Een aantal functies werden toegevoegd aan SQL om XML te kunnen zien als relationele data en vice versa.
De operator EXTRACTNODE haalt een fragment van een document op volgens een Xpath expressie en geeft het terug als een XMLType object. Hiermee kan men XML als relationele data zien.
Bij object-relationele opslag van data zal het mappen gebeuren met een XML-schema. Men kan zelf bepalen welke mapping gebruikt wordt of een default gebruiken die door de XDB gemaakt wordt. XDB kan XML-documenten terug samenstellen op niveau van DOM. Om dit te doen wordt er gewerkt met verborgen kolommen die informatie bijhouden die

[32] http://otn.oracle.com/tech/xml/xmldb/pdf/xmldb_92twp.pdf

niet door SQL kan beheerd worden. Hiertoe horen volgorde, processing instructies, commentaren en of een kolom overeenkomt met een element of een attribuut.
Obslag als CLOB kan een document exact teruggeven daar waar object-relationele opslag het document enkel kan terug samenstellen tot op het niveau van DOM.
CLOB opslag gebruikt tekst-indexen en object-relationele opslag gebruikt B-Tree indexen. Data opgeslagen als object-relationeel is onmiddelijk beschikbaar daar waar data uit CLOB's enkel bruikbaar is door data programmas die XML verstaan.
Men kan toegang hebben tot XMLType data op verschillende manieren. Java Beans die gemaakt worden vanuit een XML-schema kunnen gebruikt worden als de data object-relationeel is obgeslagen. DOM kan altijd gebruikt worden voor elke opslag methode. Beide methodes slaan veranderingen aan de inhoud op in een tijdelijk geheugen om pas later de data aan te passen met de functie XMLType.save() .
Data kan ook benaderd worden met SQL-statements die gebruik maken van de extra operators.
Een ander belangrijk onderdeel van XDB is de XML repository. Deze voorziet in een view van alle XMLType objecten in de database. Met XML-type objecten bedoelen we zowel XML-data als XML-view's over relationele data. Alle XML objecten krijgen zo een path of URL in de repository die dan kan gebruikt worden via WebDAV, FTP, JNDI en SQL. SQL heeft speciale operators om dit te doen. De repository heeft ook extra eigenschappen per object zoals, 'owner', 'modification date', 'version' en 'access control'.

7.3.4. Microsoft SQL Server

SQL Server ondersteunt XML op 3 manieren.
1. De FORXML in SELECT statements.
2. XPath queries die 'annotated XML-Data Reduced schemas' gebruiken.
3. De OpenXML functie in stored procedures.

SELECT statements en Xpath queries kunnen aangebracht worden via HTTP of via een template file.
FORXML heeft 3 opties die bepalen hoe de SELECT zijn uitvoer in XML omzet.
4. RAW geeft de resultaten weer als een tabel met één element dat 'row' noemt per rij. Kolommen kunnen weergegeven worden als attributen of als child-elementen.
5. AUTO is hetzelfde als RAW behalve dat de rij elementen dezelfde naam draagt als de tabel en het resultaat genest is in een hierarchie van tabellen in de volgorde dat ze voorkomen in de SELECT.
6. EXPLICIT laat toe een XML document te maken van een serie samengevoegde SELECT statements. Het resultaat van elk statement wordt weergegeven als een tabel en voor elke rij wordt een element gemaakt. Dit wordt geplaatst onder het vorige parent-element. Als een relatie bestaat tussen de verschillende SELECT-statements dan zullen de resultaten zich nesten volgens hun relaties.

Annotated XML-Data Reduced schemas, ook gekend als mapping schemas, hebben extra attributen die elementen en attributen mappen naar tabellen en kolommen. Deze specifieëren een object-relationele mapping tussen een XML-document en de database en worden gebruikt voor queries met een subset van Xpath. Er bestaat een gereedschap om die schemas grafisch aan te maken.
De OpenXML functie gebruikt een table-based mapping om een deel uit een XML-document te halen als een tabel en het bruikbaar te maken op elke plaats waar een tabel-naam gebruikt kan worden zoals in een FROM of een SELECT statement. Dit kan gebruikt worden met een INSERT statement om data te verplaatsen van een XML-document naar een database. Een XPath expressie geeft het element of attribute dat de rij voorstelt. Extra XPath expressies identificieren de elementen, attributen of PCDATA die de kolommen van elke rij bevatten.

Inserts, updates en deletes worden gedaan door speciaal gevormde XML-documenten die ‘updategrams’ genoemd worden. Zij bevatten de before en after data. Normaal gebruiken updategrams table-based mappings. Zij kunnen object-relationele mappings gebruiken als men een annotatie schema voorziet.

7.4. Hierarchische Databases.

Deze organiseren de data in een boomstructuur die vergelijkbaar is met directories in een bestandsysteem. Elk knooppunt kan verschillende onderliggende knooppunten bevatten. Dit is geschikt voor één-op-één en één-op-veel relaties maar niet voor de meest voorkomende veel-op-veel relaties. Hierdoor wordt nog zelden gekozen voor een hierarchische database.Waar dit type database nog wel gebruikt word is LDAP Lightweight Directory Access Protocol. Het is mogelijk LDAP te gebruiken om XML op te slaan, omdat bij beiden de data, een boomstructuur heeft.

7.5. Objectgeorieënteerde Databases.

Met de komst van OO-talen zoals JAVA zou men denken dat er plaats is voor de Objectgeorieënteerde Databases maar de interesse blijft laag. De relationele database heeft hiervoor een te sterke positie.
Het gebruik van objectgeorieënteerde databases voor XML zou nogal overdreven zijn omdat XML een aantal van de belangrijkste eigenschappen van objecten mist.

1. XML heeft geen encapsulatie. Hiermee bedoelen we het samenbrengen van data en processen. XML bevat enkel data.
2. Objecten hebben een gedrag in de vorm van uitvoerbare code die XML niet heeft.
3. Objecten verbergen hun data voor de buitenwereld en geven enkel toegang via methods. XML toont en beschrijft zijn data rechtstreeks.

Er zijn wel mogelijkheden om vanuit DOM of PDOM documenten te mappen naar objecten en deze dan op te slaan in de database. Maar wie deze weg neemt kan ook bij de andere databases terecht.
Het huidige gebrek aan kennis en ervaring met dit soort databases zal er waarschijnlijk voor zorgen dat zij weinig zullen gebruikt worden voor de opslag van XML.

8. Besluit

XML staat los van al zijn hulpmiddelen en dit omdat ze zo talrijk zijn. Een standaard is hierin nog ver te zoeken. Er zijn DTD's en schema's om de data in een XML te beschrijven. Er zijn stylesheets in de vorm van CSS of XSL die zorgen voor de layout bij de weergave van het document. Men heeft talen zoals XPath en XQuery om de documenten te ondervragen, te filteren of aan te passen. Alles is aanwezig om XML vanuit een centrale database te beheren.

Een database zo populair als de relationele zal er waarschijnlijk niet komen. Een nieuwe database die even populair wordt zou waarschijnlijk wel kunnen. De meeste XML databases bouwen verder op de reeds bestaande databases en ook dat is een mogelijk scenario. Alleen zouden ze dan allemaal voor dezelfde oplossing moeten kiezen.

Als men XML opslaat in een relationele database zal men een bepaalde vertaling moeten maken van XML naar tabellen. Hiervoor zijn voldoende programmas, programmeertalen enz... op de markt. Dze vertaling naar tabellen vraagt tijd. Tabellen zijn dan weer veel sneller om iets in op te zoeken.

Documenten die zo veel tekst bevatten dat ze niet geschikt zijn voor de omzetting naar tabellen kunnen als CLOB opgeslagen worden maar zullen dan weer moeilijker te doorzoeken zijn.

De bekenste merkprodukten zoeken naar een gemiddelde tussen die twee. Het lijkt dat ze ook wel proberen standaards te volgen. Maar voor wie echt de toekomst wil zien zijn er de Open Source produkten. Deze laatste zijn zeker de investering van de tijd waard omdat daar de standaards gemaakt worden.

Het is duidelijk dat door de lange voorgeschiedenis en uitbreidingen, van markup-talen en databases, het onmogelijk is geworden alles te bekijken.
Darwin zei al dat alleen diegene die zich optimaal aanpast aan de veranderende omstandigheden, uiteindelijk overleeft.

9. Bijlagen

9.1. Links

XML Databases

Ronald Bourret http://www.rpbourret.com/
Dit is de meest uitgebreide bron voor XML-databases. Er is geen enkel boek of andere bron die meer informatie levert.
Onderverdelingen zijn er voor:

www.rpbourret.com/xml/XMLAndDatabases.htm
www.rpbourret.com/xml/XMLDatabaseProds.htm

Onderzoek Native XML Databases – Dennis Heij en Vincent Fleur – Juli 2002

http://www.florido.nl/xml/nativeXML.pdf

XML in de database

http://www.infosupport.nl/technologie/XML.asp

World Wide Web Consortium

http://www.w3c.org/

XML:DB

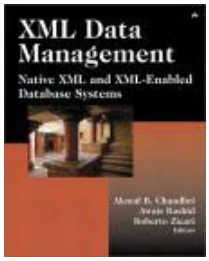
http://www.xmldb.org/

XML Tutorials

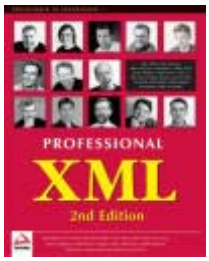
http://www.zvon.org/
http://www.w3schools.com/xml/

9.2. Boeken:

Volgende 2 boeken hebben me geholpen om XML en XML databases beter te leren kennen.



XML Data Management
Native XML and XML-Enabled Database Systems
door Akmal B. Chaudhri, Awais Rashid, and Roberto Zicari.
Publisher: Addison Wesley
ISBN: 0201844524



Professional XML
door Mark Birbeck en vele anderen
Publisher: Wrox Press Inc
ISBN: 1861005059

9.3. Alfabetische lijst afkortingen

API	Application Programming Interface
BLOB	Binary Large Object
CDATA	Character Data
CLOB	Character Large Object
CMS	Content Management System
CSS	Cascading Stylesheet
DADX	Database Access Descriptor Extension (IBM)
DOM	Document Object Model
DTD	Data Definition Language
EXD	Enabled XML Database.
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
JDBC	Java Database Connectivity
JSP	Java Server Pages
NXD	Native XML-Database
ODBC	Open Database Connectivity
PCDATA	Parsed Character Data
PDF	Portable Document Format
PDOM	Lightweight Persistency Support for DOM
RDBMS	Relational Database Management System
SGML	Standard Generalized Markup Language
SMH	System Management Hub (Tamino)
SQL	Structured Query Language
WebDAV	Webbased Distribution Authoring and Versioning
XLINK	XML Link
XML	Extensible Markup Language
XML-RPC	XML Remote Procedure Call
XSL	XML Stylesheet Language
XSLT	XSL Transformation