ABIS NIEUWSBRIEF VOOR DB2 PROFESSIONALS

JAARGANG 2 - NUMMER 7, APRIL 2004

OPEN CURSOR

Een tweede speciaal nummer!

Vorig jaar hebben we een speciaal nummer gewijd aan Java, dit jaar hebben we enkel oog voor web services. Naast een grondige uiteenzetting van wat web services uiteindelijk zijn, hun voordelen en karakteristieken, wordt ook bekeken hoe web services in een DB2-omgeving kunnen worden aangemaakt. WORF en stored procedures staan daarbij centraal. We gaan na hoe DB2 kan worden opgezet als een 'service provider' en als een 'service consumer'.
Tot slot kijken ook even verder in de toekomst; web services, het data grid, en DB2 komen hierbij aan bod.

Veel leesgenot!

Het ABIS DB2 team.

IN DIT NUMMER:

CLOSE CURSOR

In een volgend nummer terug met beide voeten op de grond: we hebben het over scrollable cursors! Opzet, performance en gebruik komen aan bod.

Tot dan!

Alles over web services

Tom Avermaete (ABIS)

Het lijkt niet onwaarschijnlijk dat wie het woord web services in dit nummer voor de eerste keer hoort vallen, de afgelopen jaren als een heremiet geleefd heeft. Zowat van overal worden we met het begrip rond de oren geslagen. Ook wij willen langs deze weg ons steentje bijdragen.

In wat volgt worden web services onder de loep genomen. In de grijze tekstblokken - voor de fanaten - gaan we een stapje verder.

Probleemstelling: web, services en web services

Laten we beginnen met het woordje WEB uit web services.

Het belang van het World Wide Web - het WEB - voor het opzoeken van informatie staat als een paal boven water. Die informatie zelf bestaat uit op web servers gepubliceerde html-pagina's op heel veel met elkaar verbonden computers. De teksten zelf worden doorgezonden via een gestandaardiseerd netwerkprotocol (HTTP), dat op zijn beurt gebruik maakt van TCP/IP. Om informatie op te zoeken volstaat het om een computer in te pluggen op dit WEB, en in een browser de URL van een zoeksite op te geven. Op deze site specificeert u vervolgens enige zoektermen, bijvoorbeeld 'weerbericht' en 'Benelux'; enkele seconden later verschijnen er links naar sites waarop u alle info vindt die nodig is om te beslissen of u voor uw uitje zonnecrème dan wel een paraplu behoeft. Of op een rijtje, de belangrijkste personages in dit korte verhaaltje: 1. Het WEB; 2. Een persoon op zoek naar weerinformatie; 3. Een zoekmachine; 4. Zoektermen; 5. Een site met gewenste informatie weergegeven in HTML. Het succes van het WEB staat buiten kijf, in die mate zelfs dat het reeds onmisbaar is geworden. Het geeft aan welke mogelijkheden standaarden ('HTTP', 'HTML') kunnen hebben op wereldschaal.

Het web - en nu komen we bij het woordje SERVICE - heeft echter nog meer mogelijkheden. Immers, ook programma's kunnen deze infrastructuur gebruiken om naar andere programma's te 'surfen' en er informatie op te halen. De spreektaal? XML! Er wordt een vraag gesteld in XML-formaat; en er komt antwoord terug in XML-formaat. Misschien even verduidelijken met een voorbeeld. Een huis-, tuin-, en keuken programma heeft toegang tot het web en 'surft' naar een zoekmachine om er adressen te zoeken waar weercomputers informatie aanbieden. Deze weerkundige informatie wordt gebruikt om te beslissen of de zonneluifel en de tuinbesproeiinstallatie al dan niet moeten worden geactiveerd. De spelers in dit verhaaltje: 1. Het web; 2. Een programma op zoek naar informatie; 3. Een zoekmachine; 4. Zoektermen; 5. Een web service die een vraag in XML binnenkrijgt en een antwoord in XML terugstuurt.

De wakkere lezer ziet waarschijnlijk reeds waar het schoentje knelt.

- De communicatie met de zogenaamde zoekmachine. Gesteld dat er een gestandaardiseerde manier bestaat om met een zoekmachine-voor-programma's gegevens uit te wisselen, dan nog blijft de vraag of mijn programma uit al de teruggestuurde links een degelijke URL zal kunnen kiezen.

- Een tweede probleem duikt op wanneer de weer-web-service moet worden aangesproken. Immers, hoe moet de XML er uitzien die moet worden opgestuurd? En nog belangrijker: hoe ziet de XML eruit die de weer-web-service gaat terugsturen naar ons huis-, tuin- en keukenprogramma.

Stel dat deze problemen worden overwonnen, wat zijn dan de voordelen? Integratie tussen verschillende programma's en programmeren zou hoe dan ook gemakkelijker verlopen: via HTTP en XML kunnen programma's van op gelijk welk platform in gelijk welke programeertaal andere programma's op gelijk welk platform in gelijk welke andere programeertaal waar ook ter wereld bereiken zonder last te hebben van firewalls, iets wat in het verleden wel eens een belemmering was voor het gedistribueerd programmeren. Door het gebruik van XML is er bovendien duidelijkheid omtrent de gebruikte characterset van de uitgewisselde boodschappen.

Toch nog even een definitie van een web service. Een web service is een via een netwerk, bij voorkeur via HTTP, bereikbaar programma (de service) dat, met behulp van XML, gegevens uitwisselt met een aanroepend programma (de service 'consument').

SOAP

De boodschap die tussen zender en ontvanger moet uitgewisseld worden, wordt standaard in een XML-schil ingepakt die men SOAP noemt - een afkorting voor 'Simple Object Access Protocol' omdat men op die manier web services (objecten in OO) kan aanspreken. Dat het letterwoord 'zeep' betekent, is geen toeval. Loodgieters gebruiken zeep om plastic buizen in elkaar te laten glijden, zoals programmeurs met SOAP programma's kunnen koppelen. De uit te wisselen boodschap wordt op een specifieke wijze ingepakt in 1 XML element met de toepasselijke naam ENVELOPE (zie tekstkader). Deze bestaat uit een optioneel element HEADER met metadata over de boodschap, een verplicht element BODY, met de eigenlijke inhoud, en een optioneel element FAULT met foutboodschappen.

Voorbeeld 1: Een eenvoudige SOAP boodschap

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <request-persons xmlns="uri:abis.be">
         <pno>12</pno>
      </request-persons>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

LITERAL vs ENCODED

De structuur van een boodschap met bijbehorende datatypes kan op twee manieren bepaald worden: literal of encoded. In het geval van LITERAL treffen we in de BODY 1 XML element aan, in een welbepaalde namespace. De namespace wordt bepaald door een XML-schema dat zowel door de afzender als door de bestemmeling gekend is. SOAP voorbeeld 1 is LITERAL gecodeerd. Bij ENCODED wordt gebruik gemaakt van een standaard schema om het dataformaat van de te versturen boodschap te bepalen. Een namespace is dan optioneel. Het element BODY kan uit meerdere elementen bestaan.

RPC vs DOCUMENT style

SOAP laat toe om Remote Procedure Calls (RPC) uit te voeren. De service 'consument' zet inputargumenten klaar, specifieert de procedure die hij wil uitvoeren, roept de service aan en wacht op antwoord. De service voert de functie uit en stuurt het resultaat naar de client die wacht op antwoord. De client krijgt het resultaat binnen en gaat verder met uitvoeren. RPC is dus in wezen synchroon. Naast de inputargumenten, (gebruik literal of encoded encoding), is het belangrijk de uit te voeren procedure aan te geven; SOAP voorziet hiervoor een standaard. DOCUMENT-style betekent dat zonder meer een XML-structuur naar de server wordt gestuurd en dat die op basis van de inhoud, of op basis van vorige vragen gaat bepalen welke acties moeten worden uitgevoerd. Dit vraagt heel wat processing. Het is dan ook logisch dat deze methode asynchroon zal gebruikt worden.

Voorbeeld 2: SOAP - literal encoding, document style

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <soapenv:Body>
     <Person>
         <Naam>Mickey</Naam>
         <Age>70</Age>
     </Person>
   </soapenv:Body>
</soapenv:Envelope>
```

Voorbeeld 3: SOAP - encoded encoding, RPC style

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:sayDays soapenv:encodingStyle="http://
schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="urn:abis.be">
         <String_1 xsi:type="xsd:string">Mickey</String_1>
         <int_2 xsi:type="xsd:int">70</int_2>
      </ns1:sayDays>
   </soapenv:Body>
</soapenv:Envelope>
```

Voorbeeld 1 verduidelijkt wat we met een XML-schil bedoelen. De boodschap zelf bestaat uit het element <request-persons>. Alle bovenliggende elementen zijn soap elementen. Merk op dat SOAP zich dus niet uitspreekt over de structuur van de boodschap zelf. Vergelijk dit met een brief die we op de post doen. Het formaat van een enveloppe, de plaats waar we de postzegel plakken, de manier waarop de postbode aangeeft dat de bestemmeling niet meer op het vermelde adres woont is allemaal informatie die niets met de boodschap zelf te maken heeft.

SOAP bepaalt niet wie de bestemmeling is, SOAP bepaalt ook niet hoe de XML-formaten worden omgezet naar een specifieke taal, dus ook niet hoe XML omgezet moet worden naar objecten. Dat is de verantwoordelijkheid van de omgeving waarin we de web service beschikbaar maken (deployen). En dit schept onmiddellijk problemen en uitdagingen. Denk hierbij aan verschillen in charactercodering, het voorstellen van strings, coderingsproblemen.

WSDL

Eenmaal gedeployed blijft nog de vraag hoe de gebruiker - de 'consument' - van een web service kan weten hoe de XML-boodschap er zelf moet uitzien, welk protocol gebruikt dient te worden (standaard HTTP, SMTP, ...) en op welke poort de web service aanspreekbaar is. We spreken hier over de functionele beschrijving van de web service.

Ook voor het maken van deze beschrijving bestaat een XML-standaard: WSDL ofwel Web Service Description Language. WSDL is een open XML-standaard. Dit betekent dat men, net zoals bij SOAP, gebruik kan maken van namespaces binnen de elementen. WSDL bestaat typisch uit 5 zich eventueel herhalende elementtypes (zie sidebar). WSDL zorgt:

- voor een abstracte beschrijving van de web service - bijvoorbeeld de functies door de service aangeboden;
- voor een concrete verwijzing naar een protocol of een specifieke URL waar de service kan worden aangesproken.

Met een WSDL heeft een programmeur - of een programma - alle technische informatie die nodig is om een service aan te spreken.

Er bestaan heel wat programma's die op basis van zo'n WSDL de programmacode voor de client - stubs - genereren in een bepaalde programeertaal. Maar daarmee is de kous nog niet af! Deze gegenereerde code is namelijk nog niet geïntegreerd in het clientprogramma zelf. De integratie van een web service in bestaande client-programma's via taalspecifieke interfaces - zoals bijvoorbeeld een Java-interface - vergemakkelijkt de hele zaak. natuurlijk wel. Maar dan verplaatst het probleem zich naar de serverkant, in die zin dat er een specifieke web service - of porttype - gevonden dient te worden die deze interface ondersteunt.

Merk op dat een op basis van een WSDL gegenereerde stub vaak op run-time andere WSDL's van een bepaalde service kan gaan doorzoeken enkel om de URL te weten te komen!

TYPE

Het type element wordt gedefinieerd met behulp van een schema. Zeker voor LITERAL encoding is dit erg belangrijk. Het schema zelf kan ook geïmporteerd worden.

MESSAGE

Het message element bepaalt welke types samen een boodschap vormen. Het is immers mogelijk dat een boodschap zowel van server naar client gestuurd wordt als van client naar server.

PORTTYPE.

Een porttype bestaat uit een reeks OPERATIONS. Deze OPERATIONS zijn de 'functies' die een bepaalde service kan uitvoeren. De OPERATIONS worden aangesproken met een message en zenden in de meeste gevallen ook een message terug. Eenzelfde message kan in meerdere operaties gebruikt worden. Indien er fouten optreden bij het aanroepen van de OPERATION kunnen er ook foutboodschappen teruggezonden worden.

BINDING

De binding beschrijft het protocol dat gebruikt wordt (HTTP, SMTP, ...) en hoe de encoding van de boodschappen dient te gebeuren. Indien SOAP gebruikt wordt, kan men deze encoding in WSDL beschrijven met SOAP-elementen om bijvoorbeeld aan te geven welke style (RPC of Document) of welke encodingstyle (LITERAL of ENCODED) gebruikt wordt. Een PORTTYPE kan op verschillende manieren (bindings) aangeproken worden.

SERVICE

De service bestaat uit de URL - of endpoint - waarop een binding kan aangesproken worden. Een bepaalde binding kan op meerdere poorten gedeployed worden.

UDDI

UDDI (Universal Description, Discovery and Integration) is een XML-specificatie die gebruikt wordt om services kenbaar te maken en op te zoeken. UDDI-registries ondersteunen deze specificatie zodat programmeurs of programma's zelf een concrete web service kunnen 'publiceren' of 'opzoeken'. Want daar gaat het om: publiceren en kenbaar maken enerzijds, en opzoeken anderzijds. Inderdaad, de essentie van UDDI.

En wat is dan UUID?

UUID staat voor Universal Unique Identifier. Het is een 128-bit nummer dat door een combinatie van URL, Timestamp en random info uniek is, waar ook ter wereld. Een UUID wordt gegenereerd door een UDDI registry op het ogenblik dat men een service registreert. Het is dus bijgevolg geen algemene unieke sleutel voor een bepaald bedrijf of die bepaalde service: een bedrijf kan dus in meerdere registries onder andere UUID's geregistreerd zijn.

Een aantal UDDI-specifieke termen worden in het tekstkader aangehaald.

BusinessEntity

Een BusinessEntity stelt een organisatie, bedrijf of een bedrijfsonderdeel voor. Een bedrijf dient zich kenbaar te maken door zijn naam, contactinformatie, URL en een beschrijving van zijn doelstellingen te registreren. De BusinessEntity is gekoppeld aan een UUID die businesskey wordt genoemd. Een bedrijf kan verder ook uniek gedefineerd worden door een of meerdere zogenaamde identifiers. Een identifier is een unieke key binnen een bepaalde namespace - het ABIS telefoonnummer binnen de lijst van Belgische telefoonnummers, het ABIS BTW-nummer, etc. Een BusinessEntity kan meerdere BusinessServices definiëren.

BusinessService

De BusinessService zelf is een abstracte definitie van wat een service met woorden beschrijft. Nog geen technische beschrijving. Een BusinessService kan verwijzen naar de UUID van een BusinessEntity. Dit is het bedrijf dat de service gespecificeerd heeft of aanbiedt. De BusinessService zelf krijgt bij publicatie in een registry zelf ook een UUID. BusinessServices worden net als BusinessEntitiies ook geclassificeerd in taxonomieën. Een BusinessService kan meerdere of geen BindingTemplates bevatten.

BindingTemplate

Een BindingTemplate is een concrete beschrijving van waar een service aangeroepen moet worden - het adres. Een BindingTemplate kan verwijzen naar de UUID van een BusinessService, indien er een abstracte beschrijving van de binding bestaat. De BindingTemplate krijgt zelf ook een UUID. Een BindingTemplate heeft minstens één verwijzing naar een tModel.

tModel

Een tModel is een technische fiche die beschrijft hoe de web service wordt opgeroepen. Elk tModel heeft een UUID. Een tModel kan bijvoorbeeld naar een URL verwijzen waar WSDL te vinden is. Een WSDL is natuurlijk maar een manier om web service te beschrijven. Een tModel zelf kan ook weer onderverdeeld worden in een taxonomie.

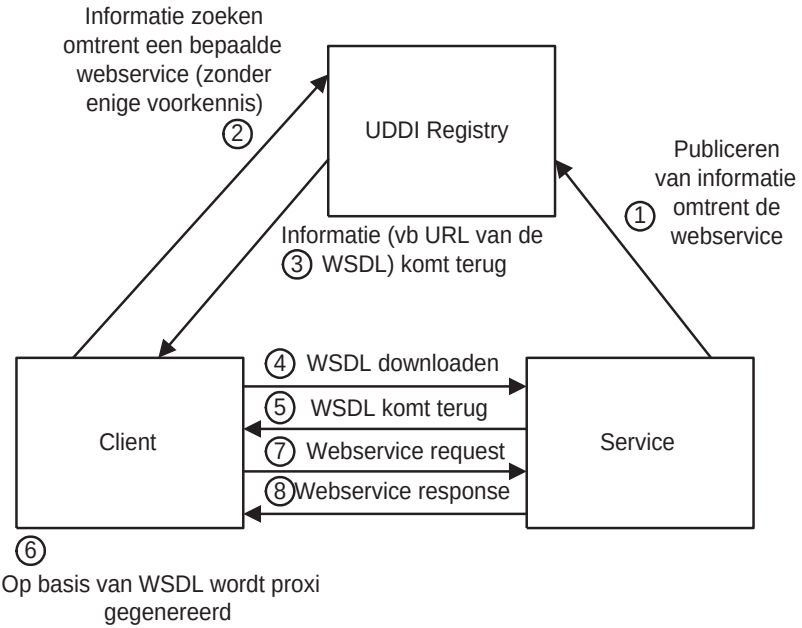
Een tModel kan ook nog een andere functie hebben. Een taxonomie (voor classificatie) en een namespace (voor identificatie) worden ook gedefinieerd door een tModel.

PublisherAssertion

Een PublisherAssertion geeft relaties tussen verschillende bedrijven aan. De vestiging van ABIS Nederland is bijvoorbeeld een dochterbedrijf van ABIS België. Binnen een tModel waarvan we u de UUID gaan onthouden, heet deze relatie parent-child.

Figuur 1 geeft u een samenvattend overzicht van de verschillende web services-specifieke elementen, die boven werden aangehaald: publiceren van WSDL informatie, het opzoeken van informatie in een UDDI registry, het bouwen van een client stub, en het aanroepen en aanwenden van een service.

Figuur 1: Een web service in actie



SLOT

We laten u nu misschien wat verweesd achter na dit technisch bombardement. Als troost misschien vermelden dat deze specificaties allemaal nog relatief jong zijn en dat er dus nog heel wat beweging te verwachten is. Om af te sluiten misschien nog even terug naar ons verhaaltje over het huis-, tuin- en keukenprogramma. Een huis-, tuin- en keukenprogramma neemt contact op met een UDDI-registry en zoekt er naar de URL van een Binding die een vooraf gekend tModel implementeert. Dit vooraf gekend tModel is gedefinieerd met een WSDL zodat ons huis-, tuin- en keukenprogramma dus op voorhand weet wat het kan verwachten van de web service die het gaat aanspreken. De URL volstaat om op runtime contact te leggen. En laat de zonneluifel nu maar uitrollen terwijl u in uw luie tuinstoel toekijkt hoe de plantjes in de tuin water krijgen allemaal dankzij web services... Wordt dus zeker nog vervolgd.

DOSSIER 8

XML

XML deed zijn intrede in DB2 V7 voor OS/390 en z/OS met de XML Extender.

Gegeven het groeiende belang van XML als dataformaat voor data transport en internet, is de aandacht voor XML in V8 nog toegenomen. V8 bevat 6 nieuwe XML publishing functies; en XML extender werd uitgebreid zodat XML kan gevalideerd worden met behulp van XML schema en kan getransformeerd worden met XSLT stylesheets.

V8 introduceert ook het nieuwe XML-datatype. Dit datatype moet tijdens de query uitvoering gebruikt worden en kan niet dienen als datatype in kolomdefinities. Het kan niet 'embedded' worden gebruikt. Het kan als waarden een XML-element, meerdere elementen, een leeg element of de content van een element bevatten.

De nieuwe built-in SQL-functies hebben als doel gemakkelijker op basis van relationele gegevens XML te genereren. Zowel XML-elementen met attributen als geneste structuren kunnen gegenereerd worden. Het resultaat van dergelijke 'generatie' bevindt zich steeds in het XML-datatype. Een eerste nieuwe functie, XML2CLOB, is dan ook een cast functie die deze XML-data omzet in 'toonbare' CLOB-data.

Een aantal scalaire XML-functies worden in wat volgt aangehaald.

a) Functie XMLELEMENT zet een relationele waarde om in een XML-element. Deze functie kan als argument van een andere XMLELEMENT functie-call worden gebruikt.

```
SELECT XML2CLOB ( XMLELEMENT( NAME "VoorNaam", firstname))
FROM persons;
<VoorNaam>Zeger</VoorNaam> <VoorNaam>Quinten</VoorNaam>
<VoorNaam>Albijn</VoorNaam> <VoorNaam>Korneel</VoorNaam>
```

b) Functie XMLATTRIBUTES genereert attributen bij XML-elementen.

```
SELECT XML2CLOB ( XMLELEMENT( NAME "Naam",
XMLATTRIBUTES (lastname AS "FamilieNaam"))) FROM persons;
<Naam FamilieNaam = "Peeters"/> <Naam FamilieNaam = "Pieters"/>
<Naam FamilieNaam = "Wouters"/> <Naam FamilieNaam = "Janssens"/>
```

c) Functie XMLAGG laat ons toe complexere structuren neer te schrijven.

```
SELECT XML2CLOB (XMLELEMENT (NAME "Lijst" ,
XMLAGG (XMLELEMENT(NAME "VNaam", fname),
         XMLELEMENT(NAME "FamNaam", lname)))) FROM persons;
<Lijst><VNaam>Zeger</VNaam><FamNaam>Peeters</FamNaam>
       <VNaam>Quinten</VNaam><FamNaam>Pieters</FamNaam>
       <VNaam>Albijn</VNaam><FamNaam>Wouters</FamNaam>
       <VNaam>Korneel</VNaam><FamNaam>Janssens</FamNaam></Lijst>
```

d) Functie XMLFOREST maakt een lijst XML-elementen aan - vergelijkbaar met het nesten van de XMLELEMENT-functie. En functie XMLCONCAT laat u toe XML-elementen te concateneren.

Katrien Platteborze (ABIS)

Over web services, WORF, en DB2

Kris Van Thillo (ABIS)

Web services worden typisch gedefinieerd als programma's - de services - die via een netwerk, bij voorkeur via HTTP, bereikbaar en aanroepbaar zijn. Deze services communiceren met het aanroepend programma - de service consumer - aan de hand van XML.

In dit artikel wordt de rol die DB2 in deze context kan spelen, in detail uiteengezet. We hebben aandacht voor DB2 als web service provider en als web service consumer!

Over WORF

WORF - een afkorting van Web Object Runtime Framework - heeft tot doel, DB2-applicaties in te pakken in een web service. Op die manier is het dus mogelijk om vanuit gelijk welke omgeving via het web een DB2-applicatie aan te spreken. Omdat web services SOAP gebruiken als communicatiemiddel en DB2 werkt met SQL en parameters, dient de ontwikkelaar in een resource file een mapping te definiëren tussen beide. Met behulp van deze mapping genereert WORF een web service-schil. Wijzigingen aan de resource file worden door WORF gedetecteerd - de web service wordt als gevolg hiervan automatisch opnieuw geladen volgens de nieuwe specificaties.

Daarnaast staat WORF ook in voor:

- HTTP Get/Post, en SOAP bindings;
- automatische generatie van WSDL voor UDDI-support - WORF fungeert echter niet als een UDDI registry, doorzoekt geen registries, etc;
- automatische generatie van test pagina's en documentatie.

De WORF-omgeving is beschikbaar voor de meeste platformen - Windows, Linux, AIX, Solaris, OS/390 en z/OS. Afhankelijk van het platform moeten specifieke afhankelijkheden worden nageleefd.

DADX resource file

De Document Access Definition eXtension (DADX) file beschrijft de door WORF te genereren web service. Strikt bekeken is een DADX-document niets anders dan een XML-document, dat aan bepaalde voorwaarden moet voldoen. Elke web service vereist aldus één DADX-document. De aangeboden web service diensten worden in de DADX file aangegeven als operaties.

Twee types van DADX web services worden ondersteund:

- SQL web services: de DADX resource file bevat SQL-statements (select, insert, update, delete en CALL), die naar de DB2-database worden gezonden. Eens uitgevoerd, wordt het resultaat (afhankelijk

van het SQL-statement, indien relevant) teruggezonden naar de oproepende applicatie - consumer - waarbij de SQL-kolomnamen als default XML-tags worden gebruikt.

- XML web services - de DADX resource file bevat DB2 XML Extender 'compose'- en 'decompose'-instructies. De compose-instructies laten het ons toe op basis van bestaande relationele data, XML-documenten samen te stellen; de decompose-operatie ontrafelt de inhoud van een aangeboden XML-document, en bewerkt hiermee de inhoud van DB2-tabellen. Een Document Access Descriptor (DAD) file is vereist om de XML - DB2 mapping te beschrijven. Voor een concrete bespreking met voorbeelden van de DB2 XML Extender, verwijzen we graag naar exploring DB2, jaargang 1, nr 4.

Voorbeeld 1 - Een DADX resource file - SQL

```
<DADX xmlns="http://schemas…">
     <operation name="listPersons">
          <query>
               <SQL_query>SELECT pfname, plname
                          FROM db2.persons</SQL_query>
          </query>
     </operation>
     <operation name="Cnt">
          <query>
               <SQL_update>UPDATE counter
                           set cnt = cnt + 1</SQL_update>
          </query>
     </operation>
     <operation name="InsertPerson">
          <call>
               <SQL_call>call db2.InPers(:lname, :com) </SQL_call>
               <parameter name="lname" type="xsd:string"/>
               <parameter name="com" type="xsd:string" kind="out"/>
          </call>
     </operation>
</DADX>
```

Een DADX resource file is opgenomen in voorbeeld 1. De SQL-based DADX-file bevat 3 operaties - een SELECT-statement die op de persons tabel zal worden uitgevoerd, een UPDATE-statement, en een CALL naar een stored procedure InPers. Deze verwacht één input- en één outputparameter. Parameters kunnen zowel voor stored procedure calls als SQL-statements worden gebruikt. Ze moeten echter wel individueel worden gedefinieerd en we moeten waken over de mapping tussen de parametertypes en de DB2-kolomdefinities!

Voorbeeld 2 - Een DADX resource file - XML

```
<DADX xmlns="http://schemas…">
     <operation name="InsertPerson">
          <retrieveXML>
              <DAD_ref>get_sessions.dad</DAD_ref>
              <no_override/>
          </retrieveXML>
     </operation>
</DADX>
```

Voorbeeld 2 toont een DAD-gebaseerde web service - de traditionele DB2 XML Extender stored procedures worden aangeroepen om XML-documenten te genereren en/of te bewerken. De tabellen die moeten worden benaderd, en de mapping tussen de kolommen en de XML-elementen (labels) wordt in de DAD-file opgenomen. Een uittreksel uit de in dit voorbeeld opgenomen DAD-file is opgenomen in voorbeeld 3.

Voorbeeld 3 - De DAD file

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "d:\dxx\dtd\dad.dtd">
<DAD>
<validation>NO</validation>
<Xcollection>
<SQL_stmt>select plname, coname, cltitle, sdate from enrol, sess,
cours, pers, comp where e_sno = sno and s_cid = cid and e_pno = pno and
e_cono = cono and pno = 104 and sno = 1 order by plname, coname;</
SQL_stmt>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Confirm SYSTEM "...\composingXML\confirm.dtd"</
doctype>
<root_node>
<element_node name="Confirm">
  <element_node name="Enrollee">
    <element_node name="LastName">
      <text_node>
       <column name="plname"/>
      </text_node>
    </element_node>
...
</root_node>
</Xcollection>
</DAD>
```

DB2 als web service provider

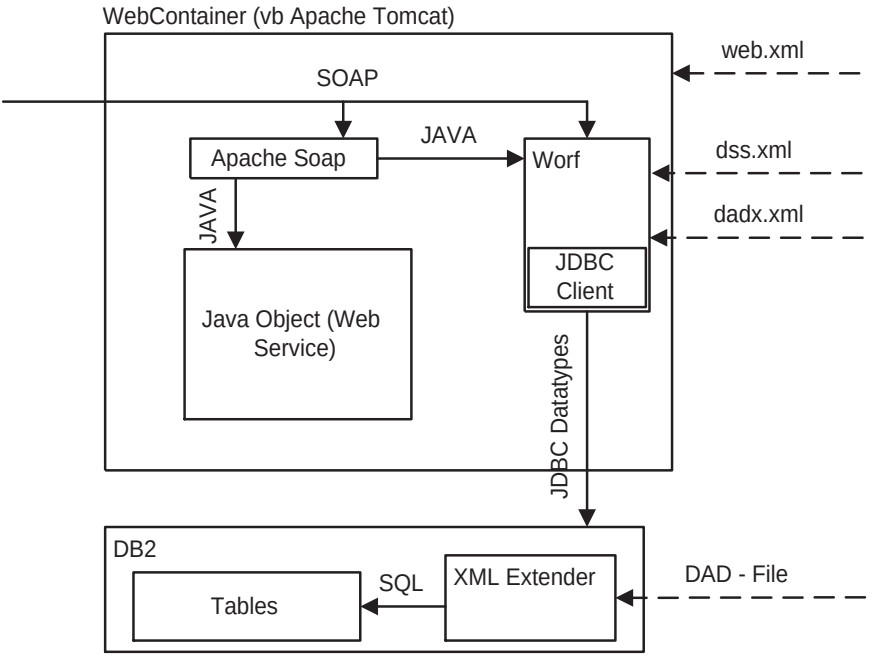
WORF dient geïnstalleerd te worden in een web container. Een web container is een omgeving waarbinnen een Java-programma zoals bijvoorbeeld een servlet kan geïnstalleerd worden. Figuur 1 geeft een globaal overzicht van de essentiële componenten die het ons moeten toelaten - aan de hand van WORF - DB2 te positioneren als een web service provider.

We hebben concreet behoefte aan volgende componenten:

- een web container voor het beschikbaar stellen van web-applicaties - en dus ook de WORF-omgeving (WORF servlet). Deze web container moet minstens SOAP 2.2 ondersteunen - WebSphere Application Server 4.0 en hoger, en Apache Jakarta Tomcat, voldoen aan deze voorwaarden. De webapplicatie heeft behoefte aan een belangrijke configuratiefile, met name de *web.xml*-file, die de binnenkomende URL-servlet mapping definieert. Of concreet: een HTTP-request associeert met een uit te voeren WORF-actie;

- een DB2-database (versie 7 of versie 8), met, afhankelijk van de te genereren web service types, een geconfigureerde DB2 XML Extender omgeving.

Figuur 1: DB2 als een web service provider - architectuur



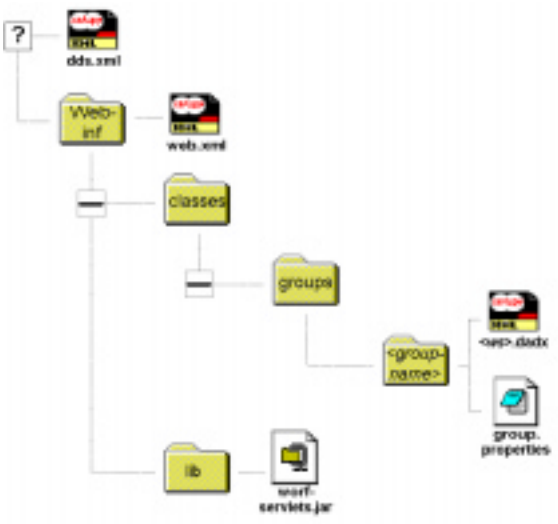
- WORF: eigenlijk een installeerbare Java servlet in een web container. Strikt bekeken heeft WORF behoefte aan volgende configuratiefiles:

  - een deployement descriptor (dds.xml). Deze file kan worden beschouwd als een mapping file, die de via een HTTP-request binnenkomende SOAP-datatypes (en requests), omvormt naar Java-datatypes (en requests). Voor elke web service die via de webapplicatie beschikbaar zal worden gesteld (lees dus: DADX-file), moet een entry worden toegevoegd aan de dds.xml-file. Hoe dit gebeurt is web server specifiek;
  - een DADX en/of DAD file. Deze laatste is enkel vereist als de DB2 XML Extender wordt gebruikt. Zoals een dds.xml SOAP omvormt naar Java, zo vertaalt de DADX Java code naar DB2/ JDBC uitvoerbare statement;
  - een group.properties-file. Deze bevindt zich op de locatie als aangegeven in de web.xml-file. Deze properties-file bevat o.a. database-connectie-informatie, en andere specificaties die voor WORF eventueel van belang kunnen zijn.

Beide laatste files vertegenwoordigen de web service die door WORF zal worden gegenereerd, en die DB2-access toelaten.

Figuur 2 geeft een overzicht van een typische directory structuur, waarin een WORF/web service omgeving werd beschikbaar gemaakt. De hier getoonde structuur is typisch voor een Apache Jakarta Tomcat omgeving.

Figuur 2 - Filestructuur




```
dds.xml  beschrijft beschikbare applicaties (web services)
web.xml  de 'group' structuur, met andere configuratie files
```

SOAP request sequentie

Bij het aanroepen van een web service door een service consumer, voert WORF de volgende stappen uit:

- laden van de DADX-file;
- laden van de DAD-file, voor XML-based DB2 web services;
- vervangen van de parameters in de DADX-file
- maken van een connectie naar DB2, uitvoeren van de gevraagde actie, en het aanvragen van een commit;
- formateren van het resultaat in XML; merk op dat het resultaat ook kan worden opgeleverd als een Java-object;
- terugsturen van de SOAP-enveloppe.

En de consumer applicatie?

WORF genereert een web service test client (de consumer) - eigenlijk een webapplicatie op basis van Java servlets en JSPs. De test client kan HTTP dan wel SOAP-bindings gebruiken, om de web service te testen - en dus is een gewone web browser voldoende!

Eens getest kan de SOAP-binding door gelijk welke consumer-applicatie worden aangewend om DB2-data ter beschikking te krijgen.

SQL vs. stored procedures

Bij het beschikbaar maken van DB2-data via web services aan de hand van WORF, hebt u dus een aantal opties. Belangrijk hierbij is te weten dat op dit moment, SQL-based web services slechts uit één enkel SQL-statement kunnen bestaan, en dat elk statement wordt beschouwd als een Logical Unit of Work (LUW). Concreet heeft dit als gevolg dat een commit wordt gevraagd op het eind van elk statement.

Vandaar ook het belang dat moet worden gehecht aan het gebruik van stored procedures, ook in deze context. Naast de traditionele voordelen (het gebruik van statische SQL, extra veiligheid, beperking van de vereiste netwerk bandwith, etc) bieden stored procedures immers de mogelijkheid om verschillende, logisch gegroepeerde acties als één geheel via een web service aan een consumer aan te bieden.

Het spreekt voor zich - eigenlijk in lijn met de idee van web services - dat deze stored procedure in gelijk welke door DB2 ondersteunde taal kan worden aangemaakt: C, C++, Java, COBOL, SQL/PL, om er maar een paar te noemen.

DADX, WSDL en UDDI

De DADX resource file bevat alle informatie die nodig is om een WSDL-document aan te maken. WORF kan dan ook op verzoek een WSDL-document aanmaken voor publicatie/registratie in en UDDI.

DB2 als web service consumer

Uiteraard kan DB2 ook als web service consumer worden ingezet. Inderdaad, aan de hand van User-Defined Functies (UDF) kunnen, rechtstreeks vanuit SQL-statements, web services worden opgeroepen. De UDF wordt aangemaakt door de DBA en/of applicatieontwikkelaar. Tijdens het uitvoeren van het SQL-statement wordt een connectie gelegd met de web service provider; de XML-response kan in de UDF worden verwerkt.

Om DB2 UDB als web service consumer te gebruiken, is minimaal volgende informatie vereist:

- het service endpoint van de op te roepen web service;
- de naam van de uit te voeren operatie/web service methode, encoding style, en eventueel input parameters.

Deze gegevens dienen als input voor de db2xml.soaphttp()-functie, die concreet volgende acties zal ondernemen:

- samenstellen van een SOAP-request;
- ‘post’ van een SOAP-request bij de web service endpoint;
- ontvangen van de SOAP-response;

- opleveren aan de oproepende omgeving (het SQL-statement) van de SOAP-body als resultaat.

Merk op dat van deze functie een aantal varianten beschikbaar zijn, afhankelijk van het gewenste datatype van de SOAP-response (CLOB, VARCHAR). Typisch zal de db2xml.soaphttp()-functie geïntegreerd worden in een afzonderlijke, applicatiespecifieke UDF. Voorbeeld 4 geeft een overzicht van de syntax van de basis db2xml.soaphttp() UDF.

Voorbeeld 4: db2xml.soaphttp() syntax

```
db2xml.soaphttpv returns VARCHAR():
    db2xml.soaphttpv (endpoint_url VARCHAR(256),
                      soap_action VARCHAR(256),
                      soap_body VARCHAR(3072))
                  RETURNS VARCHAR(3072)
db2xml.soaphttpc returns CLOB():
       db2xml.soaphttpc (endpoint_url VARCHAR(256),
                         soapaction VARCHAR(256),
                         soap_body CLOB(1M))
                      RETURNS CLOB(1M)
```

Merk op dat deze functionaliteit GEEN deel uitmaakt van de WORF-omgeving, maar juist een onderdeel vormt van de DB2 Information Integrator functionaliteit.

Tot slot - tools

Een aantal tools zijn beschikbaar om het ontsluiten van DB2 - zowel als web service consumer als provider - eenvoudiger te maken.

Binnen de DB2-omgeving kan vooral de DB2 Stored Procedure Builder (V7) dan wel het DB2 Development Center (V8) worden gebruikt. Dit voornamelijk voor het aanmaken van stored procedures, alsook de web service consumer UDF's. Voorts is een WebSphere Studio Plug-in beschikbaar om bestaande WSDL-definities te converteren naar DB2 UDF-functies.

IBM WebSphere Studio Site Developer Advanced, en WebSphere Studio Developer ondersteunen het ontwikkelen van DB2 web services aan de hand van WORF.

Grid, web services en DB2 data

Kris Van Thillo (ABIS), Eric Venmans (ABIS)

Wat is de link tussen grid, web services, en DB2 - dus data? Een korte uiteenzetting.

Grid computing

Met de term grid computing, verwijzen we eigenlijk naar een ver gevorderde vorm van 'virtualisatie'. Virtualisatie van applicaties en diensten, virtualisatie van resources (als bijvoorbeeld netwerk, hardware, storage, operating systems), virtualisatie van data. Cruciaal in een dergelijke opzet zijn bijvoorbeeld volgende uitgangspunten.

- De vaststelling dat vaak suboptimaal gebruikte resources - netwerken, CPU, I/O subsystemen - gecombineerd en gecoördineerd ingezet, tot aanzienlijke besparingen kunnen aanleiding geven, alsook tot een verhoging van de productiviteit.

- De vaststelling dat de combinatie en integratie van niet op mekaar afgestemde data en applicaties, tot nieuwe inzichten kan leiden - een uitbreiding van het datawarehouse concept.

- De vaststelling dat door het loutere effect van virtualisatie, sneller kan worden ingespeeld op zich wijzigende behoeften.

- De vaststelling dat resource virtualisatie ook leidt tot een toegenomen beschikbaarheid en performantie.

Data en het grid

Wat is dan de link tussen data en de grid? Bijvoorbeeld:

- Een individuele database, maakt samen met andere databases, onderdeel uit van een groter, virtueel geheel.
- Gebruikers van een virtuele database genereren toegevoegde waarde door data, aanwezig in onderling disparate databases, te combineren en analyseren.
- Data virtualisatie wordt aangewend door database beheerders, om bijvoorbeeld beschikbaarheid, performantie te garanderen.

Binnen de context van 'data virtualisatie' hebben we het natuurlijk niet alleen over (relationele) databases - ook data files, block level data (tape archives) etc. moeten worden in aanmerking genomen.

Wat zijn dan de belangrijkste elementen die 'data virtualisatie' mogelijk maken? Een kleine inventarisatie.

- Dataopslag-neutraliteit: data is typisch opgeslagen in een veelheid van opslagbronnen; deze data is slechts toegankelijk mits gebruik van een veelheid aan accessprotocollen - FTP, NFS, SQL, etc.

Een servicegericht data grid moet op een transparante wijze o.a. al deze protocollen ondersteunen. Efficiënte toegang vergt ook het gebruik van een datacache - de cache wordt opgeladen met data aan de hand van deze 'native' accessprotocollen.

- Lokatieneutraliteit: specifieke behoeften kunnen vereisen dat bepaalde data lokaal beschikbaar moet worden gemaakt. Dit vergt zeer specifieke eisen wat betreft datareplicatie. Concreet: replicatie wordt eigenlijk een 'end-user'-karakteristiek, waarbij de gebruiker bepaalt welke data op welk platform, wanneer beschikbaar moet worden gesteld. Reken hierbij ook migratie-tools, die geoptimizeerd zijn voor datatransfert overheen LAN/WAN-netwerken.

- Access-neutraliteit: data in het data grid moet op een gestandaardiseerde en uniforme wijze beschikbaar worden gemaakt voor de gebruiker van het grid. En dus onafhankelijk van de originele opslag wijze - oude bekende, en nieuwe standaarden kunnen hiervoor worden gebruikt: SQL, XPath, XQuery - gezamenlijk wordt hier naar verwezen onder de vlag DAIS - Data Access Interface Services.

- Metadata: het is voor de data grid gebruiker belangrijk te weten waar bepaalde data zich bevindt, waar bepaalde replica's kunnen gevonden worden, wat de kwaliteit van deze replica is. Ook systeem- en opslagelementen komen hierbij aan bod. Deze gegevens worden ook aangewend om optimaal data-access te garanderen - denk hierbij bijvoorbeeld aan 'advisories' - die het de gebruiker moeten toelaten om optimaal data-accesspaden en accessprotocollen te kiezen.

- Andere, minstens even belangrijke features: geavanceerde authenticatie- en authorisatiemechanismen, PKI, netwerksnelheid, etc.

Wat dan met web services?

De web service architectuur moet het mogelijk maken disparate diensten, applicaties, services onderling met mekaar te laten communiceren. Dit kan enkel worden gerealiseerd als de verschillende onderliggende componenten aan de hand van open standaarden communiceren - SOAP, WSDL, en JAX-RPC (Java client/server-binding voor WSDL) om er een paar te noemen. En dus ook bij het opzetten, beheren en uitwisselen van data in een data grid!

DB2 v8 - data grid

DB2 versie 8 (en omringende tools) bevat reeds een aantal technologieën die belangrijk zijn om de uitgangspunten aan de basis van het concept data grid te realiseren. In wat volgt worden een aantal van deze technologieën geïntroduceerd, met speciale aandacht voor de DB2 Information Integrator.

De DB2 Information Integrator heeft tot doel, informatie die binnen een bepaalde organisatie beschikbaar is, te integreren, en dit zowel naar de applicatieontwikkelaar/gebruiker toe, door deze data via open en transparante interfaces aan te bieden (web services, SQL-

database clients, XML, messages en workflows), als naar de data bronnen toe. Zo wordt een veelheid aan relationele en niet-relationele bronnen geïntegreerd.

Een aantal belangrijke onderdelen van de Integrator op een rijtje.

- data virtualisatie - data federation: 'read/write'-access wordt mogelijk naar een veelheid aan data bronnen. Relationele bronnen (SQLServer, Oracle, DB2, Sybase, etc), maar ook naar Messaging Systemen, ODBC en Xcell-datastructuren, content system, en natuurlijk packaged applicaties. De integrator wordt als dusdanig een uniek - lees enkelvoudig - 'aanspreekpunt' voor data-oriented applicaties.

- heterogene replicatie: Integrator ondersteunt de mogelijkheid om data te repliceren overheen verschillende bronnen, bijvoorbeeld, van DB2 naar Oracle of van Oracle naar SQL Server. Transformaties kunnen 'on-the-fly' worden uitgevoerd. Deze replicatie kan zowel tabel-based als transactie-based. Updates, i.e. synchronisatie, kan realtime dan wel uitgesteld. MQSeries speelt hierbij een belangrijke rol.

- heterogene data cache: toegenomen data access performance neemt toe door caching van data afkomstig van hetergene bronnen. Hiertoe wordt het concept van 'MQT' (Materialized Query Tables) verder uitgewerkt en uitgebreid. Inderdaad, deze kunnen nu ook verwijzen naar gelijk welke originele bron gedefinieerd in de 'federated databron'. Informatie opgeslagen onder de vorm van meta-data bepaalt of specifieke vragen op de MQTs dan wel op de originele databron worden uitgevoerd. Schrijven gebeurt in principe steeds op de originele data (of in de memory cache waar de originele data van afkomstig is). De data caches kunnen aan de hand van een aantal technieken worden refreshed - de boven besproken replicatie is er één van.

Enkele slotbedenkingen ...

Is DB2 versie 8 dus een data grid enabler? Op dit moment nog niet. Maar het bevat zeer zeker een aantal features en karakteristieken, die fundamenteel zijn voor de implementatie van een data grid. Merk trouwens op dat, alhoewel conceptueel zeer duidelijk te definiëren, we kunnen verwachten dat concrete en uitgewerkte data grid implementaties nog lange tijd op zich zal laten wachten.

CURSUSPLANNING APR - MEI - JUN 2004

DB2 concepten	375 EUR	07/06 (W)
DB2 for OS/390, een totaaloverzicht	1625 EUR	24-28/05 (W), 07-11/06 (L)
DB2 UDB, een totaaloverzicht	1625 EUR	24-25/05&01-03/06 (W)
RDBMS concepten	325 EUR	24/05 (W), 07/06 (L)
Basiskennis SQL	325 EUR	25/05(W), 08/06 (L)
DB2 for OS/390 basiscursus	975 EUR	09-11/06 (L)
DB2 UDB basiscursus	975 EUR	01-03/06 (W)
SQL workshop	700 EUR	21-22/06 (L)
DB2 for OS/390 programmering voor gevorderden	700 EUR	17-18/05 (W)
Gebruik van DB2 procedural extensions	350 EUR	19/05(W)
DB2 for OS/390: SQL performance	1200 EUR	23-25/06 (W)
DB2 UDB applicatieperformance	400 EUR	08/06 (W)
Database applicatieprogrammering met Java	800 EUR	01-02/06 (W)
Fysiek ontwerp van relationele databases.	700 EUR	03-04/05 (L)
DB2 for OS/390 database administratie	1600 EUR	24-27/05(L)
DB2 for OS/390 operations and recovery	1500 EUR	21-23/04(L)
DB2 UDB systeembeheer en performance	400 EUR	30/04 (L), 22/06 (W)
DB2 UDB en zijn extenders: XML en text search	200 EUR	04/06 (W)
DB2 UDB integratie met MQSeries	200 EUR	04/06 (W)

Plaats: L = Leuven; W = Woerden; details en extra cursussen: www.abis.be

Postbus 220
Diestsevest 32
BE-3000 Leuven
Tel. 016/245610
Fax 016/245691
training@abis.be

abis
TRAINING & CONSULTING

Postbus 122
Pelmolenlaan 1-K
NL-3440 AC Woerden
Tel. 0348-435570
Fax 0348-432493
training@abis.be