XDB Repository

File system in de database

Dit is het derde artikel in een reeks van artikelen over XML ondersteuning in de Oracle database. In deze reeks introduceren Erwin Groenendal en Marc Vahsen in detail, en aan de hand van veel voorbeelden, de mogelijkheden en toepassingen van XML technologie op het Oracle platform. Aangenomen wordt dat de lezer basiskennis van XML heeft.

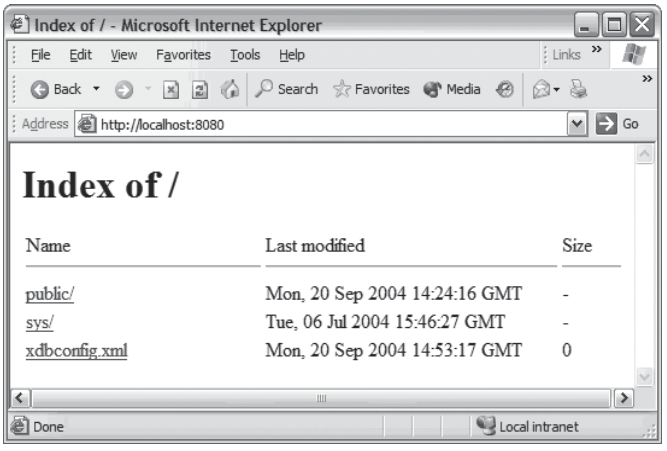
In de eerste twee artikelen is het gebruik van SQL/XML functies voor het genereren van XML data vanuit SQL en het native datatype XMLType behandeld. Dit artikel behandelt het beheren van XML files (en andere files) in de XDB Repository.

Oracle9i biedt de mogelijkheid om files op te slaan in een file system binnen de database: de Oracle XML database repository (kortweg de XDB Repository). Dit hoeven niet per se XML files te zijn, ook andere (ascii en binaire) files kunnen bewaard worden in een folder hiërarchie. De belangrijkste eigenschappen van de XDB Repository worden in dit artikel besproken.

Toegang via standaard protocollen

Een belangrijke eigenschap van XDB Repository is dat resources (dit is de XDB term voor files) benaderd kunnen worden via de standaard internet protocollen HTTP, FTP en WebDav. Poort 8080 is de standaard poort voor HTTP en WebDav, voor FTP is dat poort 2100. Zo kun je na installatie van een lokale 9.2 database de XDB Repository inhoud direct benaderen via URL http://localhost:8080.

FTP-en kan via ftp://localhost:2100 (of met username en password via ftp://scott:tiger@localhost:2100). Ook met een standaard FTP tool zoals WS_FTP kan het filesysteem worden benaderd. WebDav (dit staat voor Web Distributed Authoring and Versioning) is een standaard internetprotocol waarmee een webserver een Windows filesystem interface kan bieden in een gedistribueerde omgeving. XDB Repository sluit aan op deze WebDav standaard, waardoor de repository content benaderd kan worden door bijvoorbeeld gebruik te maken van Microsoft



Figuur 1. Na installatie van een lokale 9.2 database de XDB Repository kun je de inhoud direct benaderen.

Web Folders. Bij het voor het eerst benaderen van de webfolder wordt gevraagd om een database schema naam en password.

Indien de 9.2 database met XDB Repository op eenzelfde machine wordt geïnstalleerd als een 10g Application server, dan ontstaat er een poortconflict op poort 8080 met de OSE (Oracle Servlet Engine). Het is dan raadzaam om het poortnummer van WebDav / HTTP te veranderen. Dit kan tijdens database creatie, of achteraf door het bestand xdbconfig.xml te wijzigen in de repository root folder (kopieer het gewijzigde bestand in dit laatste geval met FTP, niet met WebDav zelf!). Het gebruik van WebDav lijkt ideaal, toch blijkt in de praktijk dat de huidige WebDav implementatie niet altijd even stabiel is. Soms lukt kopiëren van bestanden niet, en bovendien is het lastig om de schemanaam waarmee initieel is aangelogd later te veranderen, terwijl dat via FTP erg gemakkelijk is.

Toegang via SQL

Naast het benaderen van de repository via deze standaard internet protocollen, is de content ook te benaderen vanuit SQL middels de views RESOURCE_VIEW en PATH_VIEW. In de view RESOURCE_VIEW is voor iedere folder en voor

iedere resource (file) één entry opgenomen, in PATH_VIEW kunnen dat per resource (of folder) meerdere entry's zijn, namelijk als er gebruik gemaakt wordt van links (vergelijkbaar met Unix). Het absolute path van een resource kan als volgt worden opgevraagd:

```
SQL> select any_path                                        path
  2  ,      extractvalue(res, '/Resource/ModificationDate') mod_date
  3  from   resource_view
  4  where  any_path like '/public/acme_demo%'
  5  order by 1
  6  /

PATH                                MOD_DATE
----------------------------------  -------------------------------
/public/acme_demo                   22-SEP-04 10.39.33.270000 AM
/public/acme_demo/1_english.xml     02-AUG-04 09.00.53.810000 PM
/public/acme_demo/XMLType.doc       22-SEP-04 08.20.00.861000 AM
/public/acme_demo/file.xsd          22-SEP-04 10.39.33.210000 AM
/public/acme_demo/sort.xsl          02-AUG-04 09.00.47.481000 PM
/public/acme_demo/xsd               22-SEP-04 10.32.14.419000 AM
/public/acme_demo/xsd/form.xsd      22-SEP-04 10.32.14.389000 AM

7 rows selected.
```

Met de under_path operator is het mogelijk om naar een resource te zoeken binnen een specifieke folder, de depth op te vragen, de filenaam zonder pad te tonen et cetera.

```
SQL> select path(1)   path
  2  ,      depth(1)  depth
  3  from   resource_view
  4  where  under_path(res, '/public/acme_demo', 1) = 1
  5  /

PATH                                                      DEPTH
----------------------------------------                  ----------
1_english.xml                                                  1
XMLType.doc                                                    1
file.xsd                                                       1
sort.xsl                                                       1
xsd                                                            1
xsd/form.xsd                                                   2

6 rows selected.
```

Filebeheer via Standaard Protocollen

Resources (files) en folders aanmaken in XDB Repository kan eenvoudig met de standaard internet protocollen: FTP (put, mkdir,...) en WebDav (copy / paste / rechtermuisknop → new folder). Ook het kopiëren van hele subfolders ineens is mogelijk. Veel file content management tools zoals Word, Excel en XMLSpy staan bovendien toe om via WebDav en HTTP bestanden direct te openen en op te slaan. Deze directe benadering van de XDB Repository gaat goed zolang het ascii betreft. Het direct opslaan van grote binary bestanden met bijvoorbeeld Word gaat nogal eens mis. Eerst lokaal opslaan en vervolgens kopiëren via WebDav of FTP is dan het alternatief.

Filebeheer via SQL

Het manipuleren van de repository content via SQL kan (en mag) plaatsvinden door directe insert / update en delete statements op de view RESOURCE_VIEW uit te voeren:

```
SQL>   -- delete file
SQL>   delete
  2    from resource_view
  3    where any_path = '/public/acme_demo/XMLType.doc';
1 row deleted.

SQL> -- move file naar andere folder
SQL> update path_view
  2  set    path = '/public/acme_demo/xsd/file.xsd'
  3  where  path = '/public/acme_demo/file.xsd'
  4  ;
1 row updated.

SQL> -- resource toevoegen
SQL> insert into resource_view
  2  values
  3  ( sys.xmltype.createxml(
  4       '<Resource xmlns="http://xmlns.oracle.com/xdb/XDBResource.xsd"
  5            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  6            xsi:schemaLocation="http://xmlns.oracle.com/xdb/
XDBResource.xsd
  7            http://xmlns.oracle.com/xdb/XDBResource.xsd">
  8            <Author>Marc Vahsen</Author>
  9            <DisplayName>Rocklin.xml</DisplayName>
 10            <Comment>
 11              Creeer document via insert into resource_view
 12            </Comment>
 13            <Language>en</Language>
 14            <CharacterSet>ASCII</CharacterSet>
 15            <ContentType>text/plain</ContentType>
 16            <Contents>
 17               <acme_office>
 18                 <id>4</id>
 19                 <name>Rocklin</name>
 20                 <address>Sunset Boulevard</address>
 21                 <city>Rocklin</city>
 22               </acme_office>
 23            </Contents>
 24       </Resource>')
 25  , '/public/acme_demo'
 26  );
1 row created.
```

Merk op dat de bovenstaande insert laat zien dat de daadwerkelijke content van de resource binnen de <Contents> tag geplaatst wordt. Deze manier van inserten is daardoor niet erg praktisch. Verderop in dit artikel worden goede alternatieven besproken. In een grote repository vereist de performance op RESOURCE_VIEW en PATH_VIEW extra aandacht. Bijvoorbeeld het verwijderen van resources via een delete op

resource_view verloopt vele malen trager dan het aanspreken van de deleteresource() procedure uit de XDB resource API.

XDB resource API voor PL/SQL

XDB Repository biedt een standaard PL/SQL API (package DBMS_XDB) ten behoeve van resource management. Hiermee kunnen resources aangemaakt, verplaatst en verwijderd worden, zoals blijkt uit de hierna volgende voorbeelden. Allereerst het aanmaken van een subfolder:

```
SQL> declare
  2     res   boolean;
  3  begin
  4     res := dbms_xdb.createFolder('/public/acme_demo/sub');
  5  end;
  6  /

PL/SQL procedure successfully completed.
```

Het toevoegen van een resource (acme_offices.xml) verloopt via dbms_xdb.createResource. Deze functie biedt via overloading de mogelijkheid om als "source" onder meer de datatypen varchar2, clob, binary file of xmltype te gebruiken. Hier volgt een voorbeeld waarbij de xmltype content zich in de PL/SQL bevindt:

```
SQL> declare
  2     res   boolean;
  3  begin
  4     res := dbms_xdb.createResource(
  5                '/public/acme_demo/sub/acme_offices.xml'
  6             ,  xmltype('<?xml version="1.0"?>
  7                  <ACME_OFFICES>
  8                     <ROW>
  9                        <ID>1</ID>
 10                        <NAME>Costa Mesa</NAME>
 11                        <ADDRESS>600 Anton Blvd.</ADDRESS>
 12                        <SUITE>1400</SUITE>
 13                        <CITY>Costa Mesa</CITY>
 14                        <ZIP>CA 92626</ZIP>
 15                     </ROW>
 16                     <ROW>
 17                        <ID>2</ID>
 18                        <NAME>El Segundo</NAME>
 19                        <ADDRESS>222 N. Sepulveda Blvd.</ADDRESS>
 20                        <SUITE>2300</SUITE>
 21                        <CITY>El Segundo</CITY>
 22                        <ZIP>CA 90245</ZIP>
 23                     </ROW>
 24                  </ACME_OFFICES>'
 25                )
 26             );
 27  end;
 28  /

PL/SQL procedure successfully completed.
```

Het inlezen van een file vanaf een lokaal filesysteem kan middels het aanmaken van een DIRECTORY (het systeem privilege 'create any directory' is hiervoor nodig) en het specificeren van een bfilename:

```
SQL> create directory XDB_ACME_DEMO as 'c:\data\acme_demo';

Directory created.
```

Op het filesysteem bevindt zich een bestand acme_offices.xsl (de inhoud is interessant voor voorbeelden verderop in dit artikel):

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <html>
         <body>
            <h2> ACME Offices</h2>
            <table border="1">
               <th>ID</th>
               <th>NAME</th>
               <th>ADDRESS</th>
               <xsl:apply-templates/>
            </table>
         </body>
      </html>
   </xsl:template>
   <xsl:template match="ROW">
      <tr>
         <td><xsl:value-of select="ID"/></td>
         <td><xsl:value-of select="NAME"/></td>
         <td><xsl:value-of select="ADDRESS"/></td>
      </tr>
   </xsl:template>
</xsl:stylesheet>
```

Dit bestand wordt nu als volgt in de XDB Repository aangemaakt:

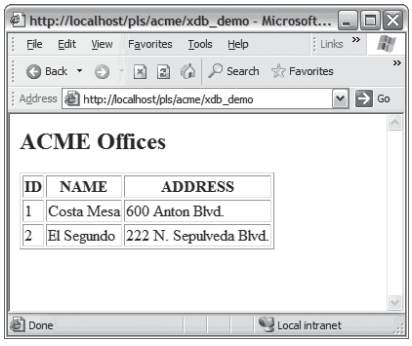
```
SQL> declare
  2     res boolean;
  3  begin
  4     res := dbms_xdb.createResource(
  5                '/public/acme_demo/sub/acme_offices.xsl'
  6             ,  bfilename('XDB_ACME_DEMO','acme_offices.xsl')
  7             );
  8  end;
  9  /

PL/SQL procedure successfully completed.
```

Nu we weten hoe repository content aangemaakt wordt, rijst de vraag hoe deze resources binnen bijvoorbeeld PL/SQL vanuit de XDB Repository ingelezen en gebruikt kunnen

worden. Dit kan met behulp van diverse methods van het objecttype XDBURITYPE. Via deze methods kan de content in PL/SQL ingelezen worden in een variabele van het type XMLType, clob of blob. Zo kun je bijvoorbeeld een XSLT transformatie volledig binnen PL/SQL uitvoeren en het resultaat direct in een browser te tonen:

```
SQL> create or replace procedure xdb_demo
  2  is
  3      l_stylesheet xmltype;
  4      l_content    xmltype;
  5      l_html       clob;
  6  begin
  7      l_stylesheet := xdburitype('/public/acme_demo/sub/acme_offices.
                         xsl').getxml();
  8      l_content    := xdburitype('/public/acme_demo/sub/acme_offices.
                         xml').getxml();
  9      l_html      := l_content.transform(l_stylesheet).getclobval();
 10      htp.p(l_html);
 11  end;
 12  /

Procedure created.
```

Indien deze procedure middels een DAD /pls/acme benaderd wordt is het resultaat van http://localhost/pls/acme/xdb_demo:



Figuur 2. Een XSLT transformatie kan volledig binnen PL/SQL uitgevoerd worden en het resultaat getoond in een browser.

Deze mogelijkheid om binnen PL/SQL bestanden aan te maken, op te halen en te bewerken biedt in combinatie met het native XMLType krachtige toepassingsmogelijkheden.

Objectrelationele opslag in XDB Repository

In het vorige artikel hebben we reeds gezien dat "schema-constrained" XML data objectrelationeel kan worden opgeslagen. Daarbij is uitgelegd hoe een xml schema geregistreerd wordt via dbms_xmlschema.registerschema(), en dat dit schema vervolgens wordt aangeduid bij het inserten van data in een xmltype kolom via xmltype('...').createschemabasedxml ('<schema URL>'). Daardoor wordt de content niet in één clob maar in een (objectrelationele) tabelstructuur opgeslagen. XDB Repository biedt ook de mogelijkheid om resources objectrelationeel op te slaan. Het onderstaande voorbeeld verduidelijkt dit. De XML Schema definitie voor het voorbeeld ziet er als volgt uit:

```
<?xml version="1.0"?>
<xs:schema elementFormDefault="unqualified"
           attributeFormDefault="unqualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:xdb="http://xmlns.oracle.com/xdb">
   <xs:element name="office"
               type="officeType"
               xdb:defaultTable="ACME_IMPORT_OFFICES"
               xdb:defaultTableSchema="ACME"/>
   <xs:complexType name="officeType">
      <xs:sequence>
         <xs:element name="city" type="xs:string" minOccurs="1"
         maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:positiveInteger"/>
   </xs:complexType>
</xs:schema>
```

Merk op dat er hier via de XDB namespace voor gekozen is de objectrelationele tabelnaam expliciet te specificeren. Dit schema wordt als file acme_offices.xsd opgeslagen in XDB Repository in een willekeurige folder, bijvoorbeeld /public/acme_demo/xsd/acme_offices.xsd. Het registreren van het schema is de tweede stap:

```
SQL> begin
  2      dbms_xmlschema.registerschema(
  3           'http://www.cumquat.nl/xsd/acme_office'
  4      ,    xdburitype('/public/acme_demo/xsd/acme_offices.xsd')
  5      );
  6  end;
  7  /

PL/SQL procedure successfully completed.
```

Als er nu een XML bestand met een verwijzing naar dit schema gekopieerd wordt naar XDB Repository, dan wordt de XML na een succesvolle validatie object relationeel opgeslagen:

```
<?xml version="1.0"?>
<office
    id="5"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://www.cumquat.nl/xsd/acme_office">
    <city>Rocklin</city>
</office>
```

Dat er sprake is van objectrelationele opslag wordt duidelijk door een select op de object relationele tabel:

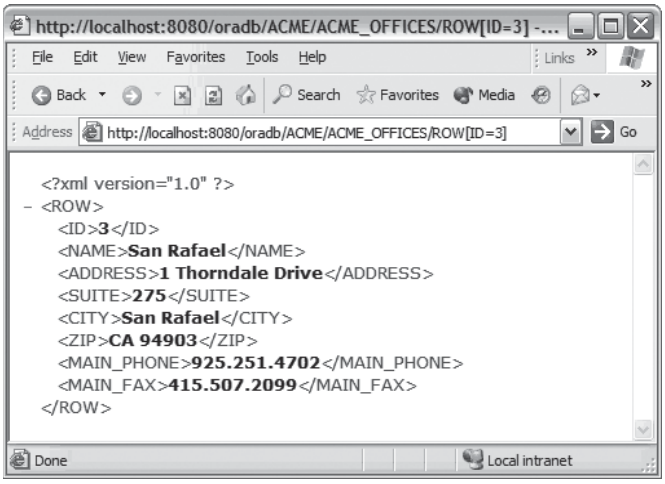
```
SQL> select table_name from user_xml_tables;

TABLE_NAME
------------------------------
ACME_IMPORT_OFFICES

SQL> select count(*) from "ACME_IMPORT_OFFICES";

  COUNT(*)
----------
         1

SQL>
```

Let op: objectrelationele tabellen staan *niet* in dictionary view user_tables maar in user_xml_tables, en tabelnamen die XDB zelf kiest zijn case-sensitive, gebruik van dubbele aanhalingstekens is daarom van belang bij het schrijven van SQL. Files die objectrelationeel zijn opgeslagen worden aangeduid met size=0. Dit is gedaan omdat het uitrekenen van de bestandsgrootte (hoeveel bytes kost de objectrelationele opslag?) de performance nutteloos verlaagt. Indien op de objectrelationele tabel vervolgens nog een of meerdere triggers worden aangemaakt dan hebben we een mechanisme in handen om een PL/SQL procedure in actie te laten komen zodra er een (schema-constrained) XML bestand ergens in de XDB Repository wordt aangemaakt!

Data access via URL's

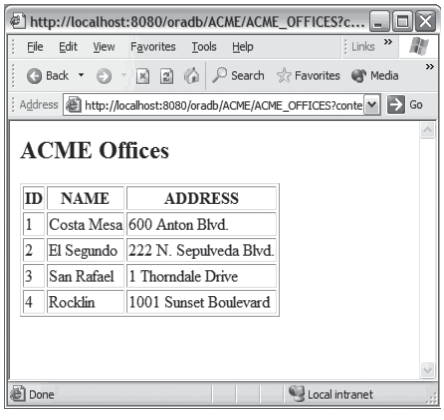
Oracle9i biedt een object type UriType waarmee het mogelijk is om Unique Resource Identifiers te benaderen en op te slaan. Een URI bestaat uit een URL, met optioneel daarachter een tweede gedeelte. Twee interessante subtypes zijn DBUriType en XDBUriType. Een toepassing van XDBUriType hebben we in een eerder genoemd voorbeeld al gezien. Daarbij wordt de getXML() method aangesproken om een XDB Resource in een PL/SQL variabele te laden van het datatype XMLType. Met DBUriType is het mogelijk om via een URI direct relationele data op te vragen in XML formaat. Bij DBUriType bestaat het tweede gedeelte van de URL uit een XPath expressie waarmee de opgehaalde XML beperkt wordt. De URI http://localhost:8080/oradb/ACME/ACME_OFFICES/ROW[ID=3] bestaat zodoende uit de machine naam, het poortnummer, de naam van de standaard DBURI Servlet (oradb), een database schema naam ACME, de tabelnaam ACME_OFFICES en daarachter de XPath expressie ROW[ID=3]. Let op: schemanaam en tabel / view naam moeten in hoofdletters staan. In Internet Explorer ziet het resultaat eruit zoals in Figuur 3.

Verder is het mogelijk om met een tweetal extra parameters direct een stylesheet transformatie uit te voeren: parameter *contenttype* specificeert het MIME type van de uitvoer, bijvoorbeeld 'text/xml' of 'text/html'. Parameter *transform* geeft de



Figuur 3. Met DBUriType is het mogelijk om via een URI direct relationele data op te vragen in XML formaat.

locatie van de stylesheet weer. Dit kan een verwijzing zijn naar een Resource uit XDB repository, maar ook een verwijzing naar een XMLType kolom in een tabel. Voorbeeld http://localhost:8080/oradb/ACME/ACME_OFFICES?contenttype=text/html&transform=/public/acme_demo/sub/acme_offices.xsl geeft het resultaat zoals in Figuur 4.



Figuur 4. Met een tweetal extra parameters is het mogelijk direct een stylesheet transformatie uit te voeren.

DBUriType is ook te gebruiken vanuit PL/SQL. De toegestane XPath expressies in de URI zijn overigens wel beperkt, zo is het gebruik van * en // operatoren niet toegestaan en moet bij het specificeren van een ROW altijd een unieke rij geïdentificeerd worden. De onderstaande expressie is niet valid omdat er meerdere rijen aangeduid worden : /oradb/ACME/ACME_OFFICES/ROW[ID>0].

Security via ACL's

Om lees- en schrijfrechten fijnmazig te regelen gebruikt XDB Repository 'Access Control Lists', afgekort ACL's. ACL's bieden een standaard beveiligingsmechanisme dat ook gebruikt wordt in Java en Windows NT. Iedere Folder en Resource in XDB Repository krijgt bij creatie altijd een ACL toegewezen, die

default gelijk is aan de ACL van de bovenliggende folder. Een ACL is intern weer onderverdeeld in Access Control Entries (ACE's) die een specifiek recht (privilege) bepaalt voor een persoon of groep (principal). Als principal kan gekozen worden voor een database schemanaam, een database-rol of de eigenaar van de resource (dav:owner). Een XDB Repository ACL bestaat uit een XML document in de folder /sys/acls. Hierin zijn verschillende standaard ACL's aanwezig zoals all_owner_acl.xml, die alleen rechten zal geven aan de resource owner.
Indien je zelf een ACL wilt definiëren op bijvoorbeeld een database rol dan kan dat in vijf stappen:

- Creëer de database rol, bijvoorbeeld ACME_DEMO_ALL
- Maak een xml bestand, bv. acme_demo_acl.xml dat de ACL beschrijft
- Plaats dit xml bestand in de /sys/acls folder
- Verklaar deze acl van toepassing op een folder of resource
- Grant de database rol aan de betreffende database schema's

Een ACL die lees- en schrijfrechten geeft aan de owner en aan rol ACME_DEMO_ALL, en alleen leesrechten aan PUBLIC ziet er als volgt uit:

```
<acl description="Acme demo privileges"
      xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
      xmlns:dav="DAV:"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
      http://xmlns.oracle.com/xdb/acl.xsd">
   <ace>
      <principal>dav:owner</principal>
      <grant>true</grant>

      <privilege>
         <all/>
      </privilege>
   </ace>
   <ace>
      <principal>PUBLIC</principal>
      <grant>true</grant>
      <privilege>
         <read-properties/>
         <read-contents/>
         <read-acl/>
         <resolve/>
      </privilege>
   </ace>
   <ace>
      <principal>ACME_DEMO_ALL</principal>
      <grant>true</grant>
      <privilege>
         <all/>
      </privilege>
   </ace>
</acl>
```

Met het volgende commando wordt deze ACL van toepassing op folder /public/acme_demo:

```
SQL> exec dbms_xdb.setacl
                  ('/public/acme_demo', '/sys/acls/acme_demo_acl.xml');

PL/SQL procedure successfully completed.
```

Let op: bestaande resources in deze folder overerven de folder ACL niet automatisch. Alleen nieuwe resources die na het toewijzen van de ACL worden aangemaakt erven de ACL. Ook files die gemoved worden in de XDB Repository behouden hun initiële ACL totdat deze expliciet wordt veranderd.

Versioning

XDB Repository ondersteunt de mogelijkheid om van één resource meerdere versies aan te maken. Daartoe biedt package DBMS_XDB_VERSION een aantal procedures zoals MakeVersioned(), Checkout(), Checkin() en GetPredecessors(). Vergeleken met andere version control tools is de functionaliteit die XDB Repository biedt beperkt te noemen. Zaken als branching/merging of folder versioning worden niet ondersteund.

Conclusie

Met XDB Repository biedt Oracle9i de mogelijkheid om naast relationele data nu veel beter grip te krijgen op data in bestandsformaat. Waar we in het verleden onze toevlucht moesten nemen tot UTL_FILE, biedt Oracle9i met XDB Repository een rijk scala aan functionaliteit voor filemanagement. We zien dat de Oracle Database steeds vaker een centrale rol gaat spelen binnen een Intranet/Internet omgeving. Daarbij biedt de Oracle XDB Repository krachtige mogelijkheden om platformonafhankelijke data (XML) te produceren, die vervolgens ontsloten kan worden middels standaard internet protocollen zoals FTP en HTTP.

Marc Vahsen
is als Senior Consultant werkzaam bij Cumquat Information Technology en kan bereikt worden via marc@cumquat.nl.