XML und die Oracle Datenbank
Storage, Query und mehr ...

Carsten Czarski
ORACLE Deutschland B.V. & Co KG
München

Einleitung

Bereits seit der Version 9.2 ist die Oracle-Datenbank mit umfassender XML Funktionalität ausgestattet. So können XML-Dokumente objektrelational, in Textform (CLOB) oder, seit Oracle11*g*, im Binärformat (*Binary XML*) gespeichert werden. Das besondere Merkmal der XML DB ist jedoch die *XML and SQL Duality*, die es erlaubt, XML Dokumente mit Hilfe von Views als relationale Tabellen und umgekehrt relationale Tabellen als XML Dokumente darzustellen.

Darüber hinaus bringt die XML DB eine Reihe weiterer Funktionen mit: Die *Protokollserver* erlauben direkte HTTP-, Webservice- oder FTP-Zugänge zur Datenbank – die Verbindung erfolgt allein über den Oracle-Listener. Das XML DB Repository bildet ein "virtuelles Dateisystem" ab, so dass ein FTP- oder WebDAV Client ohne weiteres direkt mit der Datenbank arbeiten kann. Zahlreiche Werkzeuge und Utilities runden den Funktionsumfang ab.

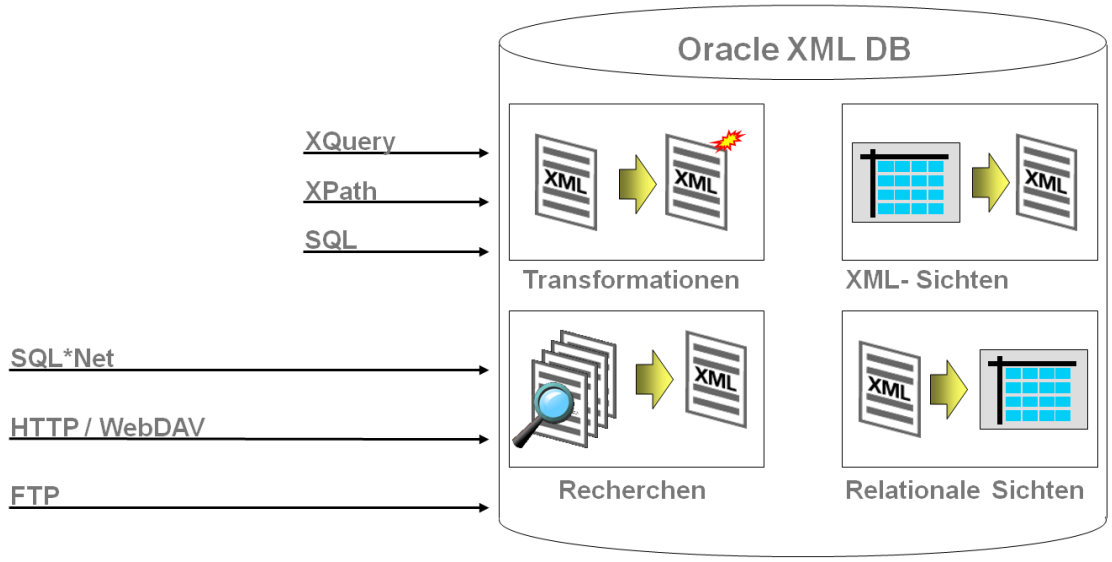


Abbildung 1: Funktionsumfang der Oracle XML DB

Umgang mit XML: XMLTYPE

Der Datentyp *XMLTYPE* ist die Kernfunktion der XML DB. XML wird in der Oracle-Datenbank als XMLTYPE repräsentiert; dieser Datentyp kann in Tabellen und PL/SQL-Logik verwendet werden. Für Tabellenspalten vom Typ XMLTYPE kann zusätzlich mit einer Storage-Klausel bestimmt werden,

welche *Speicherungsform* für die XML-Dokumente verwendet werden soll. Die Oracle-Datenbank kennt drei: Textbasiert, objektrelational und Binary XML.

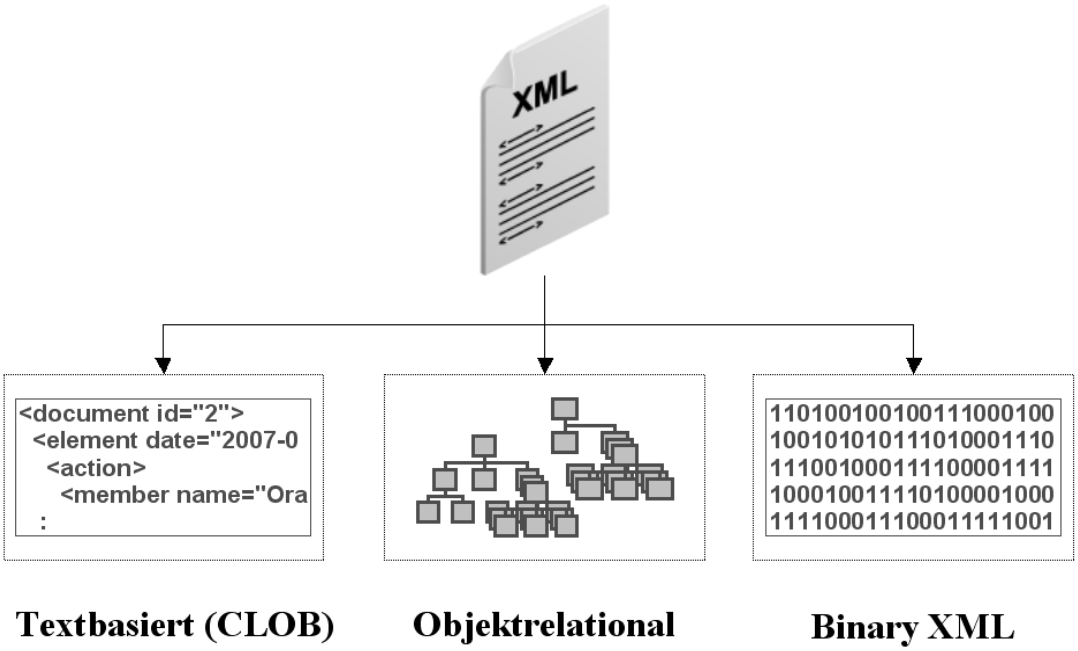


Abbildung 2: Speicherungsformen für XML

Wie immer bei einer Storage-Klausel hat die gewählte Speicherungsform keinen Einfluß auf die zur Verfügung stehende Funktionalität: In einer SQL-Abfrage oder einem PL/SQL-Block können alle XML-Funktionen, unabhängig von der gewählten Speicherungsform, genutzt werden. Auf die Performance der jeweiligen Operation hat die Speicherungsform allerdings einen Einfluß.

Welche die jeweils Richtige ist, hängt von den Anforderungen des Projekts ab. So bietet die textbasierte Speicherungsform Vorteile bei Zugriffen auf das ganze Dokument, die objektrelationale Form ist dagegen sehr gut geeignet bei Zugriffen auf einzelne XML-Tags. Binary XML zeichnet sich schließlich durch seine hohe Flexibilität aus.

Die Funktionen zum Umgang mit XML sind direkt in den Datenbankkern integriert und auf SQL-Ebene verfügbar – das "Zerlegen" eines XML-Dokuments kann mit den SQL-Funktion *XMLTABLE* oder *XMLQUERY* erfolgen. XMLQUERY "schneidet" einzelne Elemente aus und gibt sie zurück – XMLTABLE ist wesentlich mächtiger: Es ist in der Tage, die Inhalte verschiedener XML-Tags als (relationale) Ergebnisliste, wie im folgenden Listing dargestellt, zu projizieren.

```
select
  x.reference, x.username, x.costcenter
from Purchaseorder_tab p, xmltable(
  '/PurchaseOrder'
  passing xml_document
  columns
    reference  varchar2(30) path '/PurchaseOrder/Reference',
    username   varchar2(30) path '/PurchaseOrder/User',
    costcenter varchar2(10) path '/PurchaseOrder/CostCenter'
) x

REFERENCE                      USERNAME   COSTCENTER
------------------------------ ---------- ----------
ADAMS-20011127121051212PST     ADAMS      R20
ADAMS-20011127121044463PST     ADAMS      R20
ADAMS-20011127121044793PST     ADAMS      R20
:                              :          :
```

Bei Verwendung der objektrelationalen Speicherung für XML-Dokumente ist die Indizierung einfach: Alle Elemente oder Attribute können, wie Spalten einer relationalen Tabelle, mit einem Index versehen werden – dies kann ein B-Baum, ein Bitmap- oder ein anderer Indextyp sein. Für die textbasierte Speicherungsform und für Binary XML bietet die XML DB den *XMLIndex* an.

```
create index xml_index on xml_document_tab(object_value)
indextype is xdb.xmlindex parameters (
 'PATHS (INCLUDE (/PurchaseOrder/User)
                 (/PurchaseOrder/LineItems)'
)
```

Ein *XMLIndex* kann für einzelne Dokumentknoten (Elemente, Attribute), Dokumentteile oder das ganze Dokument erstellt werden. Sofern aktuelle Statistiken vorhanden sind, wird der Index vom Optimizer automatisch für Abfragen berücksichtigt. Der XML Index empfiehlt sich also zum gezielten Indizieren von XML-Dokumenten oder einzelner Dokumentknoten.

Volltextsuche in XML: XQuery Fulltext

Das neue Datenbank-Release Oracle12*c* bringt vor allem die Unterstützung neuerer XML-Abfragestandards mit. So werden *XQuery Update* und *XQuery Fulltext* in Oracle12*c* unterstützt. Vor allem XQuery Fulltext ist interessant: Zwar wird die Volltextsuche in XML-Dokumenten auch vor Oracle12*c* durch Oracle TEXT unterstützt – allerdings muss die "normale" Oracle TEXT Abfragesyntax mit der SQL Funktion CONTAINS verwendet werden.

In Oracle12*c* kann man komplett mit der Syntax des XQuery Fulltext-Standard arbeiten. So findet die folgende SQL-Abfrage alle XML-Dokumente, die innerhalb eines XML-Tags "**tag**" das Wort "**text**" enthalten.

```
SELECT id
FROM tab_xml
WHERE XMLExists('declare namespace ns="http://mynamespaces.com/ns2";
                 //ns:tag[. contains text "text"]'
                  PASSING docs)
```

XML erzeugen mit SQL/XML Funktionen

Die SQL/XML-Funktionen zum Generieren von XML sind ebenfalls seit Oracle9*i* in der Datenbank vorhanden und mittlerweile Teil des SQL:2003-Standards. Die Datenbank enthält (aus Gründen der Rückwärtskompatibilität) noch älteren Code zum Erzeugen von XML: PL/SQL-Pakete wie DBMS_XMLQUERY oder DBMS_XMLSAVE entstammen jedoch der Oracle8i-Ära und sollten nicht mehr verwendet werden. Die SQL/XML-Funktionen sind standardisiert und dazu noch wesentlich mächtiger und schneller.

`XMLElement()`	generiert ein XML-Tag
`XMLForest()`	erzeugt mehrere XML-Tags auf einmal
`XMLAttributes()`	erzeugt Attribute innerhalb eines XML-Tags
`XMLAgg()`	generiert eine Hierarchiestufe (1:n-Beziehung)
`XMLPI()`	generiert eine "Processing Instruction"
`XMLComment()`	dient der Erzeugung von XML-Kommentaren
`XMLCDATA()`	erzeugt einen größereren Textbereich in einem XML-Dokument

Tabelle: SQL/XML Funktionen

Die folgende SQL-Abfrage zeigt, wie sich aus der wohlbekannten Tabelle **EMP** XML generieren lässt. Erzeugt man mit diesem SQL eine View, so liefert diese den Datentyp XMLTYPE zurück. Ein Dritter kann dann ohne weiteres nicht unterscheiden, ob die Daten "tatsächlich" gespeicherte oder aus relationalen Daten generierte XML-Dokumente sind.

In der Praxis empfiehlt es sich, das gewünschte XML nicht durch eine einzige, sehr komplexe Abfrage oder View zu erzeugen, sondern per "Bausteinkonzept" zunächst fachlich zusammengehörige Informationen zu Teildokumenten und diese dann in übergeordneten Views zum Gesamtdokument zusammenzusetzen.

```
select
 XMLElement("abteilung",
  XMLAttributes(deptno as "abteilungsnummer"),
  (
   select
    XMLAgg(
     XMLElement("mitarbeiter",
      XMLAttributes(empno as "personalnummer"),
      XMLForest(
       ename as "name",
       hiredate as "einstellungsdatum",
       job as "stellenbezeichnung"
      ),
      XMLComment('generiert am '||sysdate||'.')
     )
    )
   from emp e where e.deptno = d.deptno
  )
) as emps_xml
from dept d
```

XML DB Protokollserver

Die Protokollserver sind ebenfalls seit Oracle9*i* Bestandteil der XML DB. Nach Aktivierung ist zunächst der Zugriff auf das XML DB Repository per FTP oder HTTP möglich – damit ist das Hochladen von XML- oder anderen Dateien in die Datenbank sehr einfach.

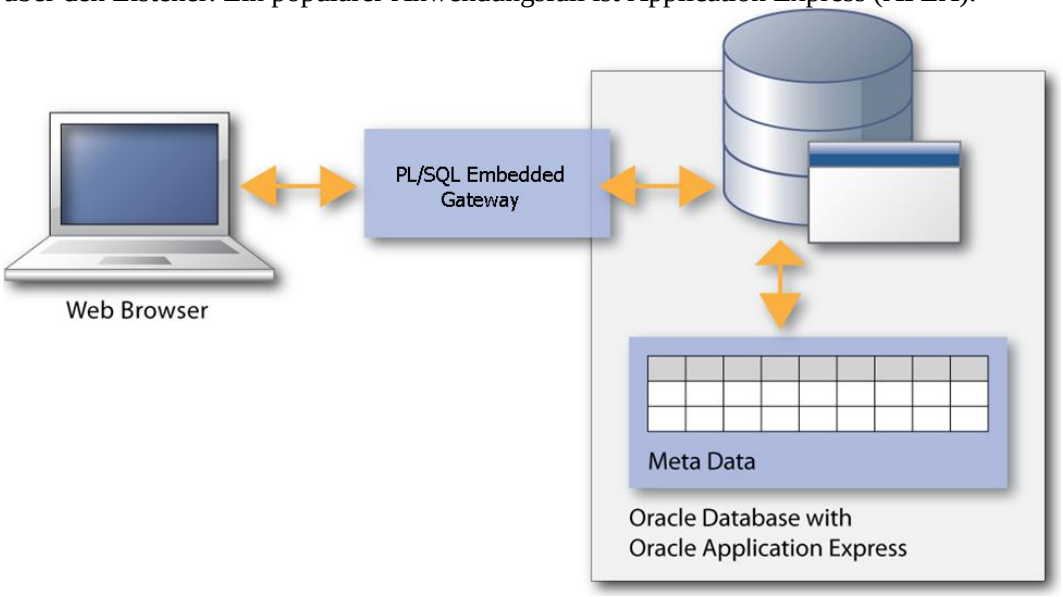
```
D:\>ftp –n
ftp> open databaseserver 2100
Connected to databaseserver.
220- databaseserver
Unauthorised use of this FTP server is prohibited and may be subject to
civil and criminal prosecution.
220 databaseserver FTP Server (Oracle XML DB/Oracle Database) ready.
ftp> user scott tiger
331 pass required for SCOTT
230 SCOTT logged in
ftp> dir
200 PORT Command successful
150 ASCII Data Connection
drw-r--r--   2 SYS      oracle         0 AUG 24 11:52 OLAP_XDS
drw-r--r--   2 SYS      oracle         0 NOV 28 16:54 home'
:
```

Inzwischen gibt es allerdings weit mehr Anwendungsbereiche für die Protokollserver – besonders der HTTP-Protokollserver wird inzwischen von verschiedenen Oracle-Komponenten genutzt:

- Verwendet die Datenbank den *Oracle Automated Storage Manager* (ASM), so enthält das XML DB Repository ab Oracle11*g* unterhalb des Ordners **sys** einen Ordner **asm**. Über diesen kann man (bspw. mit FTP) auf die ASM-Dateien zugreifen.

- Für das mit Version 11.2 eingeführte *Database Filesystem* wird ab Oracle 12*c* ebenfalls ein virtueller Ordner im XML DB Repository bereitgestellt. Damit sind HTTP- und FTP-Zugriffe auf das DBFS möglich.

- Seit Oracle11*g* steht das *PL/SQL Embedded Gateway* bereit. Es basiert auf dem HTTP Protokollserver (wird also mit diesem aktiviert) und erlaubt das Ansprechen von PL/SQL-Prozeduren, welche Ausgaben mit HTP und HTF (PL/SQL Web Toolkit) erzeugen, direkt über den Listener. Ein populärer Anwendungsfall ist Application Express (APEX).



- Das Verwaltungswerkzeug *Oracle Enterprise Manager Express* (eingeführt in Oracle12*c*), mit welchem einfache Administrationsaufgaben möglich sind, verwendet ebenfalls den HTTP-Protokollserver.

- Ab Oracle11*g* bietet die XML DB die Möglichkeit an, PL/SQL-Objekte oder SQL-Abfragen direkt über eine SOAP-Schnittstelle auszuführen (*Database Native Webservices*). Auch diese Funktion basiert auf dem HTTP Protokollserver.

Weitere Informationen

[1] Oracle Dokumentation: XML DB Developers' Guide
http://docs.oracle.com/cd/E16655_01/appdev.121/e17603/toc.htm

[2] Oracle Technology Network: XML DB
http://www.oracle.com/technetwork/database/database-technologies/xmldb/overview/index.html

[3] Blog "SQL und PL/SQL in Oracle"
http://sql-plsql-de.blogspot.com

Kontaktadresse:

Carsten Czarski
ORACLE Deutschland B.V. & Co KG
Riesstr. 25, 80992 München

Telefon:	+49 (0) 89 1430 2116
E-Mail	carsten.czarski@oracle.com
Internet:	www.oracle.de
Blog des Autors	http://sql-plsql-de.blogspot.com
Twitter	@cczarski