XML in der Oracle Datenbank „relational and beyond“

Ulrike Schwinn
Oracle Deutschland GmbH
Riesstrasse 25
D-80992 München
Ulrike.Schwinn@oracle.com

Abstract: In Geschäfts- und B2B-Anwendungen wird XML zunehmend als das Format für elektronisches Publizieren und den Austausch von Dokumenten eingesetzt. Richtiges Speichern von XML-Dokumenten ist dabei ein viel diskutiertes Thema. So werden beispielsweise beim nicht nativen Speichern Webinhalte und Applikationsdaten entweder in relationalen Tabellen, im Dateisystem oder in beidem gespeichert. Die besonderen Merkmale von XML können dabei jedoch nicht berücksichtigt werden. Bei großen Mengen von Austauschdaten stellt sich überhaupt die Frage, ob diese Speicherungsform XML-Applikationen überhaupt gerecht werden kann. Eine gute Lösung bieten hingegen die so genannten nativen XML-Datenbanken. In diesem Artikel wird am Beispiel des neuesten Oracle-Datenbank-Release das Konzept einer nativen XML-Datenbank vorgestellt.

1 Einleitung

Der Zunahme von XML-Applikationen (eXtended Markup Language) und die stärkere Nutzung von Webservices (1) erfordert, Werkzeuge zum effektiven Speichern und Verarbeiten von XML-Dokumenten.
Wählt man die Möglichkeit, XML-Dokumente in einer Datenbank zu speichern, kann dadurch sogar der Funktionsumfang der XML-Applikationen erweitert werden. Dazu tragen typische Eigenschaften der Datenbanken (DB) bei, wie zum Beispiel das Management von konkurrierenden Zugriffen, Sicherheit, Sperrverhalten, Datenintegrität mit Constraints und Trigger, sowie eine einfache zentralen Verwaltung oder ein ausgeklügeltes Backup- und Recovery-System.

Wenn aus Sicht der XML-Applikation, zum Beispiel einer Kataloganwendung oder Lastenheftverwaltung, die Entscheidung zur Speicherung in der Datenbank gefallen ist, bedeutet dies, dass die XML-Daten dauerhaft und zentral zur Verfügung gestellt werden sollen. Bei der Auswahl der Speicherung kommt es auf deren Eignung an- sowohl der Applikation als auch der Eigenschaft von XML-Daten an. So bietet sich beispielsweise das rein relationale DB-Modell an, um XML-Dokumente, daten- als auch dokumentzentrisch abzuspeichern. Allerdings stößt man schnell an die Grenzen des rein relationalen Ansatzes. Dieses Modell bildet nur eine kleine Untermenge von XML-Daten ab. XML-Charakteristiken, wie speicherspezifische Informationen etwa die Reihenfolge, dokumentspezifische Informationen oder Kommentar können dabei nicht berücksichtigt werden.

Objektrelationale Datenbanken dagegen erweiterte Möglichkeiten indem sie mit einer Abbildung auf Objekte. Das folgende Beispiel illustriert diese Möglichkeit:

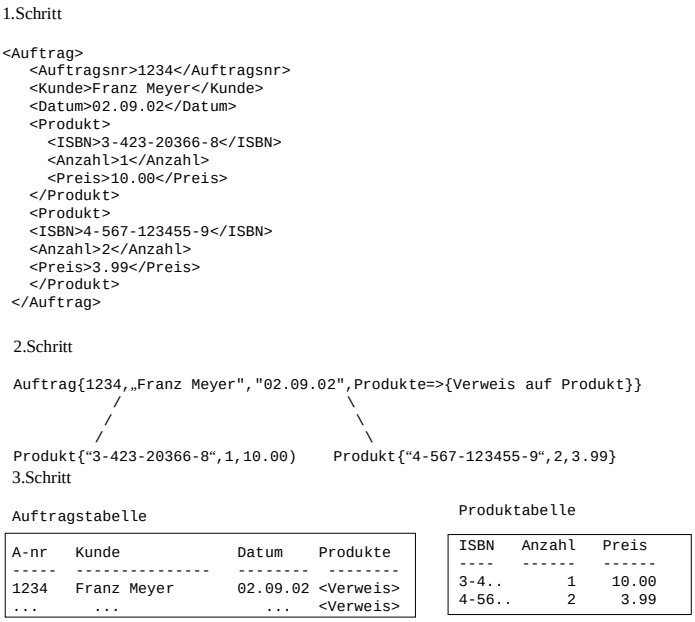

Abb. 1.1: Schrittweises Überführen eines XML-Dokuments in eine objektrelationale Speicherform

Über die reinen Speicherungsformen hinaus müssen die Verarbeitungsmöglichkeiten innerhalb der DB berücksichtigt werden. Das heißt, es müssen Zugriffsmethoden, Mechanismen zum Darstellen oder Wiederherstellen von XML-Dokumenten ergänzt beziehungsweise sichergestellt werden.
Unabhängig von den dargestellten Abbildungsformen sind daher die so genannten nativen XML-Datenbanken seit geraumer Zeit im Gespräch. Die Oracle-Datenbank ist ein Beispiel für solch eine Datenbank. Die Definition von Kimbro Staken (2) gibt vor, welche Eigenschaften eine Datenbank haben muss, um XML nativ abspeichern zu können: Native XML Datenbanken

- definieren ein logisches Datenmodell, nach dem sie die XML-Dokumente speichert und wiederherstellt. Beispiele für solche Modelle sind XPATH, XML-Schema, Modelle, die das „Document Object Model“ (DOM) unterstützen und andere,
- haben das XML-Dokument als eine logische Einheit wie Zeilen einer Tabellen in der Datenbank,
- sind unabhängig von den darunter liegenden Speicherungsformen.

2 So speichert Oracle XML

Trotz der Unabhängigkeit von der Speicherform aus Sicht der XML-Applikation und des Datenzugriffs, ist das Speicherdesign für die Performanz und Konkurrenzfähigkeit der Applikation relevant. Daher wird im Folgenden ein kurzer Überblick zu den unterschiedlichen Speicherformen innerhalb der Oracle-Datenbank gegeben.
Wie im oberen Abschnitt schon erwähnt, werden XML-Dokumente durch ihre Charakteristik in dokument- und datenzentrische Dokumente unterteilt. Die Speicherform als Character Large Object (CLOB) ermöglicht dokumentzentrisches Speichern. Die objektrelationale Speichervariante kommt dem datenzentrischen Dokument-Typ entgegen. Damit die Datenbank zusätzliche XML-Zugriffsweisen bereitstellt, steht der Datentyp XMLTYPE zur Verfügung. Bei einfacher Angabe von XMLTYPE wird dadurch eine dokumentzentrische Speicherung als CLOB möglich.

```
CREATE TABLE auftrag of XMLTYPE;
CREATE TABLE auftrag (auftrags_nr NUMBER,
                      A_Beschreibung XMLTYPE);
```

Abb. 2.1: Beispiele für die dokumentzentrische Speicherung

Um die objektrelationale Speicherung beeinflussen zu können, muss die Datenbank eine Beschreibung der XML-Strukturen erhalten. Dies geschieht durch den Einsatz von standardkonformen annotated XML-Schemas, die die Oracle-spezifischen Beschreibungen als Attribute in den Schema-Annotations mit liefert. Auf diese Weise können zum Beispiel die per Default vergebenen Systemnamen oder Speicherformen für Objekte durch sinnvolle Namen und speziellen Speicherformen ersetzt werden. Damit ein solches XML-Schema nicht proprietär wird, kommt die Technik der Namensräume (Namespace), ein Bestandteil des XML-Standards, zum Einsatz. Durch Angabe eines speziellen Oracle-Namespace werden diese Attribute eindeutig von W3C-Schema-Informationen abgegrenzt. Solche Schemas können problemlos von beliebigen - nicht Oracle spezifischen - Werkzeugen benutzt werden, da der Standard vorsieht, dass unbekannte Annotations einfach ignoriert werden.
Dies hat zur Folge, dass in solchen XML-Schemas, in der Regel mindestens zwei Namensräume angegeben werden – einen für W3C, http://www.w3c.org/2001/XMLSchema und einen für die Oracle-Charakteristiken, http://xmlns.oracle.com/xdb.
Folgende Abbildung zeigt die objektrelationale Speicherung von Abbildung 1.1 in der Datenbank. Die Oracle-spezifischen Annotations sind mit dem Aliasnamen „xdb“ gekennzeichnet.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xdb="http://xmlns.oracle.com/xdb"
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.oracle.com/xsd/auftrag.xsd"
      elementFormDefault="qualified"
     attributeFormDefault="unqualified">
  <xs:element name="Auftrag" xdb:SQLName="AUFTRAG"
     xdb:SQLType="AUFTRAG_T"xdb:defaultTable="AUFTRAG_TAB">
     <xs:complexType>
       <xs:sequence>
        <xs:element name="Auftragsnr"
            type="xs:integer" xdb:SQLName="A_NR"
            xdb:SQLType="INTEGER" />
         <xs:element name="Kunde" type="xs:string"
             xdb:SQLName="KUNDE"
              xdb:SQLType="VARCHAR2" />
         <xs:element name="Datum" type="xs:date"
          xdb:SQLName="DATUM" xdb:SQLType="DATE" />
         <xs:element name="Produkt"
             maxOccurs="unbounded"
             xdb:SQLName="PRODUKTE"
             xdb:SQLType="PRODUKT_T"
             xdb:defaultTable="PRODUKT_TAB"
             xdb:SQLInline="false">
          <xs:complexType>
            <xs:sequence>
                 <xs:element name="ISBN"
                .....
            </xs:sequence>
          </xs:complexType>
             ....
  </xs:schema>
```

Abb. 2.2: Schema auftrag.xsd für das XML-Dokument auftrag.xml

Folgende Annotations sind im Schema auftrag.xsd enthalten:

- SQLName für den Objektnamen,
- SQLType für den Oracle Datenbanktyp,
- SQLInline zur Festlegung, ob ein XML-Dokument in einem oder mehreren Objekten gespeichert wird,
- DefaultTable für den Tabellennamen.

Darüber hinaus können mit diesen Annotations aber auch zusätzliche Constraints, Partitionierungen oder auch Speicherung von XML-Fragmenten in CLOBs (Character Large Objects), in speziellen Feldern und Arrays, in Oracle-Terminlogie auch Varrays oder Nested Tables genannt, mit diesen Annotations definiert werden.

Nach Registrieren des Schemas von Abbildung 2.2, sind die Objekttypen AUFTRAG_T und PRODUKT_T und die zugehörigen Objekttabellen und Arrays durch die Annotations in der Datenbank angelegt worden.
Nachfolgende schemakonforme XML-Dokumente wie z.B. auftrag.xml werden dann in den entsprechenden Objekttabellen wie PRODUKT_TAB und AUFTRAG_TAB

gespeichert. Zur Identifikation des zugehörigen XML-Schemas dient dabei eine eindeutige URL, die beim Registrieren des Schemas festgelegt wurde. Diese URL, wird wie der Standard es vorgibt, im Attribut `xsi:schemaLocation` im XML-Dokument mitgegeben - das erste Argument liefert den zugehörigen Namensraum und das Zweite die eindeutige URL, die beim Registrieren festgelegt wurde.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Auftrag xmlns=http://www.oracle.com/xsd/auftrag.xsd
         xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xsi:schemaLocation="http://www.oracle.com/xsd/auftrag.xsd
                     http://www.oracle.com/xsd/auftrag.xsd">
  <Auftragsnr>...
```

Abb. 2.3: XML-Dokument zum Schema auftrag.xsd

Als dritte Variante lassen sich auch semistrukturierte Dokumente, die weder rein Daten- noch Dokument-zentrisch sind, als Form der objektrelationalen Speicherung realisieren. Möglich wird das durch die Technik, Teilfragmente als CLOB im XML-Schema anzugeben. Ein Beispiel für solch eine Anwendung wäre ein Presseartikel, der teilweise aus einem stark strukturiertem Dokument-Header (Autor, Rubrik, Datum) und einem unstrukturierten Dokument-Body (Text) besteht.

Die Abbildung zeigt einen Ausschnitt aus solch einem Schema:

```
<xs:schema xmlns:xdb=http://xmlns.oracle.com/xdb
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
  <xs:element name="ARTIKEL"
      xdb:defaultTable="ARTIKEL_TAB"
      xdb:tableProps="partition by range
      (xmldata.head.DAT) (partition p1 values less than
      (to_date('1998-12-31','YYYY-MM-DD')),... partition
      p6 values less than (to_date('2003-12-31','YYYY-MM-
      DD')))">
    <xs:complexType xdb:SQLType="ARTIKEL_T">
      <xs:sequence>
         <xs:element name="HEAD"
          type="HEADType" xdb:SQLName="HEAD" />
          <xs:element name="BODY" type="BODYType"
            xdb:SQLName="BODY" xdb:SQLType="CLOB"
            />
        </xs:sequence> ...
```

Abb. 2.4: Beispiel einer semistrukturierten Speicherung

3 Der Funktionsvorrat im Überblick

Neben den verschiedenen Speicherkonzepten bietet die Oracle-Datenbank wichtige Erweiterungen zur Unterstützung von XML-Applikationen.

In einer objektrelationalen Datenbank ist es natürlich möglich, über SQL- Zugriffe Daten abzufragen oder zu verändern. Dabei ist jedoch die Integration der XPATH-Funktionalität (XML Path Language) in den SQL-Abfragen wichtig, um XML-Anfragen standardgemäß implementieren zu können. Um dabei nicht jedes Mal den ressourcenintensiven DOM- Baum aufbauen zu müssen, besteht bei objektrelationaler Speicherung die Möglichkeit, den Zugriff direkt auf die Objekte abzubilden. Diese Methode, auch Query Rewrite genannt, beschleunigt nicht nur die Abfrage, sondern verkleinert auch den Speicherverbrauch. Das folgende Beispiel illustriert solch ein Query Rewrite.

Folgenden Anfrage

```
SELECT EXTRACT (value(a), '/Auftrag/Kunde') FROM auftrag_tab a
```

wird automatisch umgeschrieben auf

```
SELECT a.xmldata.kunde FROM auftrag_tab a
```

Darüber hinaus bietet die Datenbank die Möglichkeit, verschiedene Indizierungsmechanismen einzusetzen, um die Abfragen zu beschleunigen. Der B*-Index zum Beispiel beschleunigt Abfragen von Daten, die objektrelational gespeichert sind. Der Volltextindex, der besonders bei dokumentzentrischen Daten wichtig ist, unterstützt die XPATH-Syntax und kann unabhängig von der Speicherung zur Beschleunigung der Volltextsuche eingesetzt werden.

Nicht zuletzt können ändernde Zugriffe, die konkurrierend stattfinden sollen, entweder durch Änderung und Austausch des gesamten Dokuments oder aber durch Bearbeiten nur eines Teilbaumes erfolgen. Um konkurrierenden Zugriffe auf Teilbäume ohne Sperren des gesamten Dokuments zu ermöglichen, ist speziell für die objektrelationale Speicherung die SQL-Funktion XMLUPDATE hinzugefügt worden. Sie ermöglicht den so genannten „piecewise Update“ auf SQL-Ebene.

Abgesehen von den traditionellen datenbankseitigen Zugriffen durch SQL-Kommandos stehen Protokoll-Schnittstellen wie HTTP, WEBDAV (Web Distributed Authoring and Versioning) und FTP zur Verfügung, um Zugriffe von beliebigen Client-PCs ohne zusätzliche Installation zu ermöglichen. Dazu ist das Datenbankmodell eigens um Funktionen und Strukturen wie Verzeichnisse, Versionierung und ACLs (Access Control Lists) erweitert worden, die in einem Repository in der Datenbank liegen. Dieses Repository, das automatisch in der Datenbank zur Verfügung steht, bildet die hierarchische Verzeichnisstrukturen auf die XML-Ressourcen wie XML- Dateien oder Schemas intern ab. So wirkt ein Oracle- XML-Datenbank-Verzeichnis wie ein natives Betriebssystem-Verzeichnis. Der Anwender kann zum Beispiel Oracle-XML-

Datenbank-Verzeichnisse im Windows- Explorer öffnen und auf XML-Dateien zugreifen oder aber Web-Verzeichnisse im Internet-Explorer nutzen. Auch der FTP-Zugriff unterscheidet sich nicht von einem gewöhnlichen FTP-Server. Veränderungen, die über die Protokolle durchgeführt werden, können allerdings nur über den Austausch des gesamten Dokuments erfolgen.

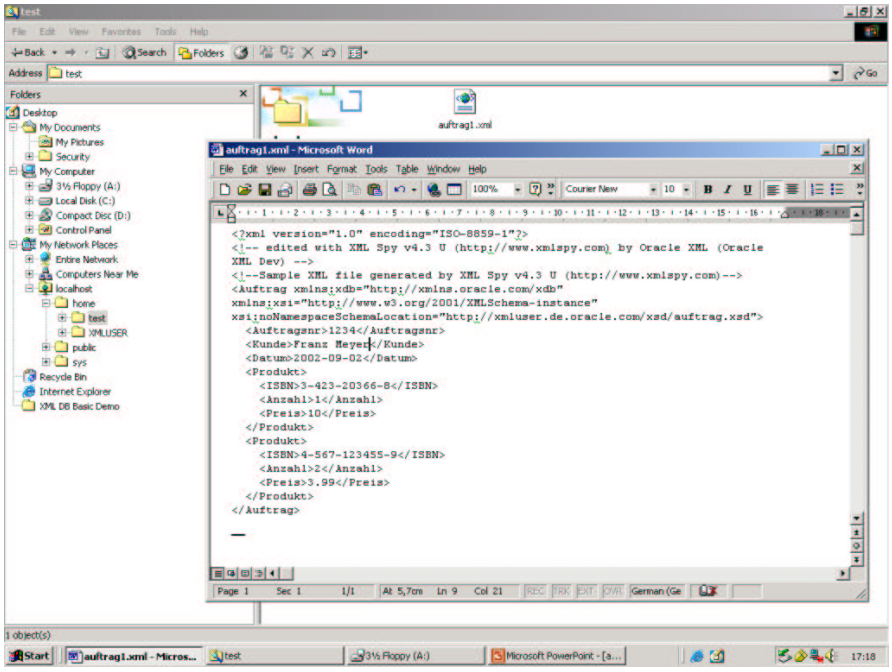

Abb. 3.1: Zugriff auf ein XML-Dokument über dem Windows-Explorer

Alle Dokumente, die auf diese Art und Weise geladen werden, sind automatisch geparst. Dies bedeutet, dass sobald die Datenbank anhand der URL das XML-Dokument als Instanz eines schon registrierten Schemas erkennt, wird das Dokument zerlegt und die Informationen in die entsprechenden Tabellen gespeichert. Sind zusätzliche Integritätsbedingungen wie Constraints und Trigger durch die üblichen Datenbankoperationen hinzugefügt worden, werden diese ebenfalls berücksichtigt.
Insbesondere beim Verstoß gegen ein Constraint kann dies für einen Ladevorgang über das FTP-Protokoll bedeuten, dass XML-Dokumente wegen Integritätsverletzung nicht geladen werden. Dies ermöglicht dem Anwendungsprogrammierer zusätzliche zentrale Funktionalität in die Datenbank zu verlegen.

```
ftp> open mucssu3 2100
Connected to mucssu3.
220 mucssu3 FTP Server (Oracle XML DB/Oracle9i Enterprise
 Edition Release 9.2.0.1.0 - Production) ready.
User (mucssu3:(none)): xmltest
331 pass required for XMLTEST
Password:
230 XMLTEST logged in
ftp> mput auftrag2.xml? y
200 PORT Command successful
150 ASCII Data Connection
550- Error Response
ORA-00604: error occurred at recursive SQL level 1
ORA-00001: unique constraint (XMLUSER.REFERENCE_IS_UNIQUE)
violated
550 End Error Response
ftp: 528 bytes sent in 0,00Seconds 528000,00Kbytes/sec
```

Abb. 3.2: Kopierversuch eines ungültigen XML-Dokuments

Ein weiterer wichtiger Aspekt beim Arbeiten mit XML-Dokumenten ist der Zugang durch das HTTP-Protokoll. Wie wir gesehen haben, unterstützt Oracle bei XML-Ressourcen, die in der Datenbank gespeichert sind, das HTTP- und WEBDAV-Protokoll.
Darüber hinaus ermöglicht ein Servlet, auf Tabellendaten über URLs zuzugreifen. Nur durch Angabe einer URL in einem Browser können somit ohne zusätzliche SQL-Programmierung, relationale Daten angezeigt werden. Das Servlet unterstützt das Generieren von XML- and Nicht-XML-Inhalte wie auch das Transformieren der Resultate durch Stylesheets. Dies erledigt ein Stylesheet-Prozessor, der direkt in der Datenbank liegt. Die Transformation kann durch SQL-Kommando oder durch den Aufruf einer URL, die den Inhalt des XML-Dokuments und des Stylesheets beinhaltet, durchgeführt werden. Im folgenden Beispiel (Abbildung 3.3.) wird durch das Wort „transform“ der Stylesheet-Prozessor aufgerufen. Als Resultat wird das transformierte Ergebnisdokument zurückgeliefert.

```
http://mucssu3:8080/oradb/XMLUSER/AUFTRAG_TAB/ROW[auftrag/kunde='Franz
Meyer'?transform=/public/xsl/auftrag.xsl&contenttype=text/html
```

Abb. 3.3: URL für ein zu transformierendes XML-Dokument

4 Fazit und Ausblick

Da die Oracle-Datenbank in der Lage ist, XML-Daten und relationale Daten in einem Server zu verwalten, ist es möglich, einerseits relationale Sichten auf XML-Dokumente und andererseits XML-Sichten auf relationale Daten zu ermöglichen.
Im ersten Fall kann unter Anwendung der XPATH-Technik und unter Zuhilfenahme von zusätzlichen Oracle-Funktionen, wie zum Beispiel der EXTRACT-Funktion eine relationale Sicht erzeugt werden. So bleibt für den Anwender völlig transparent, ob die

Daten relationalen oder XML-Ursprungs sind. Relationale Reporting-Werkzeuge können somit ohne Veränderungen auf die Daten zugreifen.
Der umgekehrte Weg, relationale Daten im XML-Format anzuzeigen, ist ebenfalls sehr wichtig, da eine große Menge von Business-Daten mittlerweile in Datenbanken relational gespeichert ist. Dabei wird die Nachfrage von Applikationen, diese Daten im XML-Format zur Verfügung zu stellen, immer größer. Oracle bietet für diesen Fall die Möglichkeit, SQL-Daten per Browser im XML-Format anzuzeigen. Auf diese Weise können nicht nur rein relationale Daten und sondern auch XML-Daten angezeigt werden.
Um komplexere XML-Strukturen zu erzeugen, unterstützt Oracle außerdem Abfragen im SQL-Stil. Diese Art der Abfragen, auch bekannt unter dem Namen SQL/XML, wird durch die SQLX-Gruppe (6), in der Oracle ein aktives Mitglied ist, standardisiert. SQL/XML ist eine Erweiterung zu SQL und führt neue Funktionen und Operatoren ein. Mit Funktionen wie zum Beispiel XMLElement, XMLAttributes in SQL-Abfragen, kann der Programmierer XML-Dokumente generieren, die beliebig mit anderen relationalen oder XML-Tabellen kombiniert werden können.
Die W3C Gruppe, arbeitet im Moment an der Standardisierung von XQuery (5), einer reinen XML-Abfragesprache. Da sich diese im Moment noch im Working Draft- Status befindet, bietet Oracle für interessierte Anwender einen XQuery–Prototypen im „Oracle Technology Network“ (4) zum Herunterladen an.

Mit der Oracle-XML-Datenbank Technologie ist ein großer Schritt zur Integration von relationalen Daten und XML-Dokumenten gemacht worden. Die Anschaffung von zusätzlichen XML-Servern sind somit nicht notwendig, um XML-Daten zu nutzen. Analysten sagen eine größere XML-Nutzung voraus; relationale Daten werden ihre Bedeutung aber beibehalten. Die Integration beider Welten in einem zentralen System, mit offenen standardkonformen Zugriffen, ist deshalb ein zukunftsweisender Technologieansatz.

4 Literaturverweise

(1)Bulletin Worldwide XML Database Management Software Forecast, 2002–2006 Analyst Susan Funke
(2)Introduction to Native XML Databases by Kimbro Staken October 31, 2001: http://www.xml.com/pub/a/2001/10/31/nativexmldb.hmtl
(3)Oracle9i XML Database Developer’s Guide – Oracle XML DB Release 2 (9.2) March 2002, Part No. A96620-01
(4)Oracle Technology Network: http://otn.oracle.com
(5)W3C Architecture Domain: http://www.w3.org/XML/Query
(6)SQLX Workgroup: http://www.sqlx.org
(7)Relational and Beyond: Carsten Czarski und Ulrike Schwinn: http://www.oracle.com/global/de/pub/knowhow/index.html?oracle_xml.html