XOBE-DB
Entwicklung einer Datenbankprogrammiersprache für XML-Anwendungen

Henrike Schuhart
Institut für Informationssysteme
Universität zu Lübeck
schuhart@ifis.uni-luebeck.de

Zusammenfassung

XML gewinnt immer stärker an Bedeutung für WWW-Anwendungen, insbesondere als Datenaustauschformat. Die spezielle XML-Sprache HTML bildet dabei die Grundlage für u.a. dynamisch generierte WWW-Seiten, die mit heutigen, in der Praxis verwendeten Werkzeugen nicht garantiert fehlerfrei sind. Auch Persistenz für XML-Strukturen ist nicht integriert. Im XOBE-DB-Projekt geht es um die Entwicklung und Implementierung einer Datenbankprogrammiersprache für XML-Objekte. XOBE steht für XML-Objekte. Aufbauend auf der Programmiersprache Java werden konventionelle Zeichenkettenoperationen zur Generierung von XML-Strukturen durch eine objektorientierte Generierung in XML-Syntax ersetzt. Durch eine einfache Deklaration können XOBE-Objekte persistent gemacht werden. XML-Objekte werden, falls sie einem XML-Schema oder einer DTD genügen sollen, bereits statisch auf Typkorrektheit überprüft, so dass nur korrekte XML-Dokumente entstehen. Testläufe sind damit überflüssig. Dieser Typcheck wird für XML-Objekte in linearer Zeit durchgeführt und basiert auf Typinferenzmechanismen. Durch die Integration von XPath- und XQuery-Sprachkonstrukten steht eine Anfragesprache, sowohl für transiente als auch persistente XML-Objekte, zur Verfügung. XQuery soll um UPDATE-Operationen erweitert werden und damit möglichst SQL-entsprechend sein. UPDATEs werden, falls die Validität mit einem Schema gefordert ist, nur dann ausgeführt, wenn das veränderte XML-Dokument schemakonform bleibt. Intern werden XML-Objekte als (persistente)DOM-Objekte abgelegt.

1 Einleitung

In den letzten Jahren hat sich XML mehr und mehr zu der Metasprache für die Strukturierung von Daten jeglicher Art entwickelt. Vor allen Dingen die wachsende Bedeutung des Webs und E-Commerce Applikationen, bei denen XML als de-facto Standard für den Datenaustausch verwendet wird, haben dazu beigetragen. Heutige Web-Anwendungen bekommen es immer häufiger mit einer wachsenden Anzahl dynamisch generierter XML-Dokumente verschiedenster Markup-Sprachen zu tun. Für die Implementierung dieser Anwendungen kommen in der Praxis Werkzeuge zum Einsatz, die die Korrektheit der entstehenden Dokumente nicht garantieren können. Diese fehlende Garantie führt zu zeitaufwendigen Testläufen. Neben nativen XML Datenbanken wie Software AGs Tamino [10] planen oder bieten beinahe alle Anbieter von Datenbankmanagementsystemen XML-Schnittstellen an [7]. Diese XML-Schnittstellen unterstützen das Exportieren, Ansehen der persistenten Daten als XML und zum Teil auch das Importieren von XML-formatierten Daten in die Datenbank. Allgemein weist vieles daraufhin, dass es bald für alle Datenbanken eine standardisierte XML-Sicht geben wird.

In diesem Artikel wird eine Erweiterung der bekannten Programmiersprache Java genannt XOBE(XML Objekte) vorgestellt. XOBE ermöglicht es Programmierern, gültige XML-Dokumente bzw. XML-Fragmente zu erzeugen und zu speichern, sowie lesende und schreibende Anfragen

zu stellen. Die Gültigkeit wird unter Angabe eines zugehörigen XML-Schemas oder wahlweise einer DTD überprüft. Weiterhin beschreiben wir einen Persistenzmechanismus für XOBE, sowie die Anbindung an ein natives XML-Datenbanksystem.

Im Einzelnen liefert die Arbeit folgendes:

- Es wird ein Überblick über die Java-Erweiterung genannt XOBE gegeben.
- Syntax und Semantik für Anfragen an XOBE werden vorgestellt.
- Die statische Überprüfung der Gültigkeit eines XOBE-Programms wird erläutert.
- Es wird gezeigt, wie XOBE-Objekte persistent in einer nativen XML-Datenbank abgelegt werden können.

Der Artikel gliedert sich wie folgt. Zu Beginn in Abschnitt 2 wird ein Überblick über verwandte Arbeiten gegeben. Im dritten Abschnitt erfolgt die Einführung in XOBE. Abschnitt 4 beinhaltet die Beschreibung der Anfragemöglichkeiten in XOBE. In Abschnitt 5 wird ein Persistenzmechanismus für XOBE-Objekte vorgestellt und in Abschnitt 6 auf die genauere Implementierung eingegangen.

2 Verwandte Arbeiten

Während Java Servlets und Java Server Pages XML-Dokumente mittels Zeichenkettenkonkatenation generieren, verfolgen andere Ansätze eine eher strukturbasierte Erzeugung von XML. Zwei Hauptvertreter dieser letzteren Richtung sind JAXB [6] und Castor [4], hier werden spezielle Klassen abgeleitet, auf die die XML-Dokumente dann abgebildet werden. Entscheidend bei diesen Ansätzen ist die strikte Trennung zwischen textueller und objektbasierter Darstellung von XML-Dokumenten. Damit einhergehend sind `Unmarshalling` bzw. `Marshalling` Prozesse, die zwischen den beiden Darstellungen hin und her konvertieren. Andere Projekte wie BIGWIG [2] und dessen Nachfolgerprojekt JWIG [3] verwenden Templates, um XML-Dokumente zu erzeugen. Problematisch bei diesem Ansatz ist vor allen Dingen die ineffiziente statische Programmanalyse. XDUCE [5] ist eine funktionale Sprache, die speziell für die Verarbeitung von XML-Dokumenten entworfen ist. XDUCE produziert ausschließlich korrektes XML. Wie XOBE, so verwendet auch JWIG Java als Basis-Programmiersprache, die Vorteile dieser Vorgehensweise sind eine zu erwartende größere Akzeptanz und es müssen nicht jedesmal die Basisfunktionalitäten einer Programmiersprache neu definiert und implementiert werden. Das hat zur Folge, dass sich XOBE auf XML- und Datenbank- spezifische Fragestellungen konzentrieren kann. Darüberhinaus bietet XOBE die Möglichkeit, XML-Dokumente in natürlicher Syntax zu bearbeiten. Die Gültigkeit der erzeugten XML-Dokumente kann dabei statisch in linearer Zeit mit größtmöglicher Sicherheit garantiert werden. XOBE verwendet als Grundlage für eine Anfragesprache XPath sowie XQuery [12], wobei letzteres vor allen Dingen um UPDATE-Operationen erweitert worden ist. Die Syntax ist aus [11] übernommen worden.

3 XOBE

Durch die Spracherweiterung XOBE wird Java um XML-Objekte ergänzt. XML-Objekte werden sowohl dazu verwendet die baumähnliche Struktur von XML-Fragmenten, als auch die darin eingebette Information darzustellen. XML-Objekte werden wie eingebaute Datentypen konstruiert und können auch äquivalent zu diesen im ganzen Programm benutzt werden. Die Struktur der XML-Objekte muß einer im vorhinein deklarierten Sprachbeschreibung, in Form eines XML-Schemas bzw. einer DTD, genügen. Implizit werden die darin enthaltenen Deklarationen als Definitionen von XML-Objekt-Klassen aufgefaßt und für die statische Typüberprüfung genutzt.

Vergleichbar mit der Konstruktion eines Objekts vom Typ `String` in Java, stellt XOBE einen Konstruktor für XML-Objekte zur Verfügung. Das folgende Beispiel soll das Prinzip verdeutlichen:

```
(1) autor a = <autor>Wassilios Kazakos</autor>;
```

Mittels dieser Zeile wird der Variablen `a` der XML-Objekt-Klasse `autor` ein XML-Element zugewiesen.

```
(2) int euro = 40;
    preis p = <preis>{euro}</preis>;
```

In der letzten Zeile wird der Variablen `p` der XML-Objekt-Klasse `preis` ein XML-Element zugewiesen, ohne dass dabei dessen Inhalt direkt angegeben wird, stattdessen wird der Wert der `int` Variablen eingesetzt.

```
(3) buch b = <buch>
             <titel>Datenbanken und XML</titel>
             {a}
             <autor>Andreas Schmidt</autor>
             {p}
             </buch>;
```

Für Listen von XML-Objekten bietet XOBE ein zusätzliches Klassenkonstrukt an, welche sich besonders in Schleifen und Rekursionen als geeignet erweist, um eine beliebige Anzahl verschiedener Elemente aufzunehmen. Deklarationen solcher Listen sind folgendermaßen möglich:

```
(4) [autor]* a;
    [autor | titel]* at;
```

Bestandteile von XML-Dokumenten, Elemente und Attribute, haben im Allgemeinen unterschiedliche Typen. XOBE benutzt die dort definierten Elemente samt ihrer Inhaltsmodelle, Definitionen von benannten Gruppen sowie komplexen Typen für die Überprüfung der Typkorrektheit. Die Einbeziehung einer Sprachbeschreibung erfolgt in XOBE mittels expliziter Deklaration.

```
(5) ximport <URL>;
```

Weitere Mechanismen, um XML-Klassen in XOBE zu definieren, sind nicht vorgesehen. Die Aufgabe des Typsystems ist es nun, die Typen der im Programm verwendeten Ausdrücke zu inferieren und in einem zweiten Schritt zu überprüfen, ob sie sich an gültigen Positionen befinden. Am Beispiel `(3)` soll dieser Ansatz demonstriert werden. Die Bezeichner `a` und `p` sind als Variablen vom Typ `autor` bzw. `preis` deklariert worden. Demnach ergibt sich auf der rechten Seite folgender inferierter Typ:

```
(6) r = buch[titel[string];autor;autor[string];preis]
```

Der Typ der linken Seite(`s`) im Beispiel `(3)` sei nach Definition der Sprachbeschreibung: `buch[title[string];autor[string]`$*$`;(preis | EW) ]`, wobei `;` für den Sequenzoperator, $*$ für 0 bis unendlich mal und `EW`(empty word) für das leere Wort stehen. Das Typsystem prüft nun, ob die Subtypbeziehung `r`$\leq$`s` gilt. Im Beispiel ergibt sich eine gültige Zuweisung. Das Xobe-Typsystem besteht im Wesentlichen aus den folgenden drei Bestandteilen:

- Durch die Formalisierung wird die Sprachbeschreibung in eine maschinenlesbare Form übersetzt.
- Die Typinferenz, die sich für XPath, XML-Konstruktoren und FLOWR-Ausdrücke unterscheidet, bestimmt die XML-Typen.
- Der Subtypalgorithmus, der die Gültigkeit der inferierten XML-Typen überprüft. Eine detaillierte Beschreibung des Algorithmus, der in XOBE zum Einsatz kommt, befindet sich in [8],[9]. Die Idee basiert auf einem Algorithmus von Antimorov [1].

4 Zugriff

Um Daten aus XML-Objekten zu selektieren, unterstützt XOBE sowohl XPath als auch die SQL-ähnlichen FLOWR-Ausdrücke, die Bestandteil von XQuery sind. Die Semantik der beiden W3C-Vorschläge wird dabei auf das XOBE-Modell übertragen. XPath wird auch in XQuery dazu benutzt, bestimmte Knoten mittels Pfadausdrücken aus XML-Fragmenten zu selektieren. In XOBE werden diese XML-Fragmente ausschließlich durch XML-Objekte repräsentiert, folglich können hier Dateninhalte sowie geschachtelte XML-Objekte angesprochen werden.

```
(7) [autor]* a = b/child::autor;
```

Ein XPath-Pfadausdruck wird immer in Bezug zu einem Kontextknoten ausgewertet, in XOBE ist dies eine XML-Objektvariable. In `(3)` ist `b` als Variable vom Typ `buch` deklariert worden, in `(7)` werden alle Kindknoten namens `autor` selektiert. Das Ergebnis, sowohl eines einfachen XPath-Ausdrucks als auch eines FLOWR-Ausdrucks, ist in XOBE immer eine Liste von XML-Objekten. Mit anderen Worten können diese Ergebnisse wieder in XML-Konstruktoren eingesetzt werden. Für komplexere Anfragen, insbesondere für Anfragen an persistent gespeicherte XML-Objekte in einer Datenbank, bietet XOBE FLOWR-Ausdrücke an. Wesentliche Bestandteile der FLOWR-Ausdrücke sind FOR, LET, WHERE und RETURN-Klauseln. Ein zentraler Punkt von XQuery ist das Konzept der Variablenbindung. Das Ergebnis einer solchen Anfrage wird in der RETURN-Klausel konstruiert und ist wiederum eine Liste von XML-Objekten.

Da XQuery an sich eine rein lesende Anfragesprache ist, für eine vollwertige Datenbankanfragesprache aber auch verändernde Anfragen benötigt werden, ist in XOBE der syntaktische Erweiterungsvorschlag für FLOWR-Ausdrücke aus [11] übernommen worden. Statt der RETURN-Klausel kann nun auch alternativ eine UPDATE-Klausel zum Einsatz kommen. Die UPDATE-Klausel bezieht sich immer auf ein Kontextelement, auf welches folgende Operationen angewendet werden können.

- Löschen eines Kindknotens - `DELETE $child`
- Umbenennen eines Kindknotens - `RENAME $child TO name`
- Ersetzen eines Kindknotens durch einen neuen - `REPLACE $child WITH content`
- Einfügen eines Kindknotens - `INSERT content [BEFORE | AFTER] $child`

```
(8) autor a3 = <autor>Peter Tomczyk</autor>;
    [autor]* a = FOR $i IN b
                 UPDATE $i INSERT {a3} AFTER $i/autor[name()="Andreas Schmidt"];
```

5 Persistenz und Implementierung: XOBE-DB

Die Java-Erweiterung XOBE ist als Präprozessor realisiert, der die XOBE-Programme mit erweiterter Syntax in reine Java-Programme transformiert. Die Validierung findet bereits zur Compilierzeit statt und kann für den Transformationsprozess bereits vorausgesetzt werden. XOBE-intern werden XML-Objekte in DOM-Objekten [13] abgelegt und Anfragen in Anfragen an DOM-Objekte übersetzt. Die DOM-Objekte können dann mittels nativen XML-Datenbanksystemen persistent abgelegt werden. XOBE verwendet zur Zeit das Infonyte-DB-System. In XOBE kann mit persistenten XML-Objekten völlig transparent gearbeitet werden, wie folgender Programmausschnitt demonstrieren soll:

```
XMLDocument document = new XMLDocument(<filename>); //Dokument wird geladen
bibliothek bib = document.getRoot();   //Wurzelknoten wird referenziert
[buch]* b = $bibliothek/buch$; //alle Buecher der Bibliothek werden selektiert.
//Buecher werden veraendert
for(int i=0; i<b.getLength();i++){
 buch bTemp =  b.item(i);
```

```
 int alterPreis = $b/preis$;
 int neuerPreis = alterPreis+20;
 preis preisNeu = <preis>{neuerPreis}</preis>;
 if(alterPreis<10){
  buch neuesBuch = $FOR $i IN b
                   UPDATE $i REPLACE $i/preis WITH {preisNeu}$;
 }
}
document.commit(); //Veraenderungen am Dokument werden gespeichert.
```

Innerhalb der for-Schleife werden alle Bücher der Bibliothek, deren aktueller Preis weniger als 10 Euro beträgt, um 20 Euro teurer gemacht.

Literatur

[1] Valentin Antimorov. Rewriting regular inequalities. *Fundamentals of Computation Theory - Lecture Notes in Computer Science*, 965:116–125, 1994.

[2] Claus Braband, Anders Moeller, and Michael I. Schwartzbach. The <bigwig> project. *ACM Transactions on Internet Technologies*, 2(2):79–114, May 2002.

[3] Aske Simon Christensen, Anders Moeller, and Michael I. Schwartzbach. Extending java for high-level web service construction. *to appear ACM Transactions on Programming Languages and Systems*, 2004.

[4] ExoLab Group. Castor. *http://castor.exolab.org/*, 11 December 2001.

[5] Haruo Hosoya and Benjamin C. Pierce. Xduce: A statically typed xml processing language. *ACM Transactions on Internet Technology*, 3(2):117–148, September 2003.

[6] Sun Microsystems Inc. The java architecture for xml binding(user guide). *http://www.sun.com*, 2001.

[7] Wassilios Kazakos, Andreas Schmidt, and Peter Tomczyk. *XML und Datenbanken.* http://www.springer.de, 2002.

[8] Martin Kempa and Volker Linnemann. Type checking in xobe. *BTW 2003*, pages 240–246, 2003.

[9] S.Martin Kempa. *Programmierung von XML-basierten Anwendungen unter Berücksichtigung der Sprachbeschreibung.* PhD thesis, Universität zu Lübeck, 2003.

[10] Harald Schöning. A dbms designed for xml. In *Proceedings of the 17th International Conference on Data Engineering(ICDE 2001)*, 2001.

[11] Igor Tatarinov, Zachary G. Ives, Alon Y. Halevy, and Daniel S. Weld. Updating xml. *ACM SIGMOD*, pages 413–424, 2001.

[12] W3Consortium. *XQuery 1.0 and XPath 2.0 Formal Semantics.* 2000.

[13] W3Consortium. Document object model(dom). *http://www.w3.org/DOM/*, 2003.