SQL-basierte Datenbankzugriffe und XML: Klassifizierung von Anwendungsprogrammen

Thomas Müller
Friedrich-Schiller-Universität Jena
Lehrstuhl für Datenbanken und Informationssysteme
Ernst-Abbe-Platz 2
07743 Jena
mueller@informatik.uni-jena.de

Zusammenfassung

SQL:2003, die neuste Version der SQL-Norm, sieht erstmals die Unterstützung von XML-Daten vor. Damit wird es möglich, Tabellen mit XML-wertigen Spalten anzulegen. Der vorliegende Beitrag geht der Frage nach, welche prinzipiellen Möglichkeiten es für die Übergabe von XML-Werten zwischen Datenbanksystem und Anwendungsprogrammen gibt. Da die Eignung dieser Möglichkeiten von den Charakteristika der Anwendungsprogramme abhängt, wird eine Anwendungsklassifikation vorgeschlagen, auf deren Grundlage die Eignung der Übergabeformen betrachtet wird.

1 Einführung

Seitdem die Meta-Auszeichnungssprache XML (eXtensible Markup Language) 1998 vom World Wide Web Consortium als Empfehlung verabschiedet wurde, hat sie stetig an Bedeutung gewonnen. Als Format zur Darstellung und zum Austausch von Daten ist XML [W3C04b, Sch03] aus der heutigen IT-Landschaft kaum wegzudenken.

Mit der zunehmenden Verbreitung von XML wächst das Bedürfnis, XML-Dokumente effizient speichern, anfragen und verarbeiten zu können. Hieraus ergibt sich die Forderung an Datenbanksysteme, XML zu unterstützen [KM03, Bro04]. Von der Industrie werden derzeit zwei Wege eingeschlagen, eine derartige XML-Unterstützung anzubieten. Zum einen werden sogenannte „native“ XML-Datenbanksysteme [See03] entwickelt, die vollständig auf die Speicherung und Verarbeitung von XML-Daten ausgerichtet sind. Als prominentester Vertreter ist hierbei der von der Software AG angebotene *Tamino XML Server* [Sof03] zu nennen. Zum anderen werden die — seit vielen Jahren zur Speicherung und Verwaltung großer Datenmengen etablierten — relationalen bzw. objektrelationalen Datenbanksysteme um XML-Funktionalität erweitert. Beispiele hierfür sind *DB2* von IBM [IBM02] bzw. *Oracle* [Ora03].

Der Einsatz nativer XML-Datenbanksysteme bietet sich an, wenn ausschließlich XML-Daten zu speichern und zu verarbeiten sind. Der Zugriff auf die Daten erfolgt dabei über eine XML-orientierte Schnittstelle wie beispielsweise DOM [W3C04a] oder XQuery [W3C03]. Soll jedoch auf XML-Daten und objektrelationale Daten gemeinsam über eine SQL-Schnittstelle zugegriffen werden, ergibt sich die Notwendigkeit, ein „XML-fähiges“ objektrelationales Datenbanksystem einzusetzen, da native XML-Datenbanksysteme i. d. R. keinerlei SQL-Unterstützung bereitstellen.

Der vorliegende Beitrag konzentriert sich auf den zuletzt genannten Fall, d. h., den SQL-basierten integrierten Zugriff auf XML-Daten und traditionelle objektrelationale Daten, so dass native XML-Datenbanksysteme hier nicht näher betrachtet werden. Warum ein solcher *SQL*-basierter Ansatz eine hohe Relevanz besitzt, wird im Abschnitt 2 erläutert, wobei auch auf die XML-Erweiterung der SQL-Norm eingegangen wird. In Abschnitt 3 werden anschließend verschiedene Möglichkeiten für die Übergabe von XML-Werten zwischen Datenbanksystem und Anwendungsprogrammen vorgestellt. Da die Eignung dieser Übergabeformen von den Eigenschaften der Anwendungsprogramme abhängt, wird in Abschnitt 4 eine Klassifikation der Anwendungsprogramme vorgenommen, anhand derer die Eignung der Übergabearten besprochen

wird. Abschnitt 5 fasst den Beitrag zusammen und umreißt kurz die künftigen Forschungsschwerpunkte.

2 Relevanz SQL-basierter Ansätze

XML hat innerhalb kürzester Zeit eine extrem hohe Popularität erreicht. Möglicherweise könnte der Eindruck entstehen, dass die SQL-basierte Datenbanktechnologie in absehbarer Zukunft komplett von einer XML-basierten Technologie abgelöst werden wird. Hiergegen sprechen jedoch u. a. folgende Argumente:

- SQL ist eine weit verbreitete, etablierte, mächtige, herstellerunabhängige, normierte Datenbanksprache, in deren Nutzung viele Anwendungsprogrammierer versiert sind.
- SQL wird von den Datenbankmanagementsystemen zahlreicher Hersteller unterstützt. Entsprechende Systeme werden (z. T. bereits seit über zwei Jahrzehnten) stetig weiterentwickelt und kontinuierlich bezüglich Performance und Funktionalität verbessert.
- Die heutige Datenbanklandschaft wird deutlich von (objekt-)relationalen Datenbanksystemen dominiert. Sehr viele Daten sind objektrelational gespeichert und eine hohe Anzahl von Datenbankanwendungen basiert auf dem Zugriff über SQL.

Somit ist — trotz der enorm zugenommenen Bedeutung von XML — zu erwarten, dass der Bedarf an einem *SQL*-basierten Datenzugriff auch in Zukunft bestehen und dominierend bleiben wird. Um dabei auch die Verarbeitung von XML-Daten zu ermöglichen, ist es notwendig, XML-Daten in SQL zu berücksichtigen. Dies wurde von den SQL-Normierungsgremien erkannt. Die neuste Version der SQL-Norm, SQL:2003, sieht — im Gegensatz zu ihrer Vorgängerversion SQL:1999 — die Unterstützung von XML-Daten vor [ISO03b, Tür03, MKF+03]. Im Normteil „SQL/XML“ [ISO03a] wird der SQL-Basisdatentyp „XML“ eingeführt, welcher mit einem Nullwert, einem XML-Dokument oder einer Folge von XLM-Elementen instanzierbar ist. Damit wird es möglich, Tabellen anzulegen, welche Spalten des Typs *XML* besitzen. Zudem definiert die Norm verschiedene Funktionen, die es erlauben, XML-Werte (Instanzen des XML-Datentyps) aus herkömmlichen SQL-Daten bzw. anderen XML-Werten zu generieren.

3 Formen der Übergabe von XML-Werten

Die Nutzung von Tabellen mit XML-wertigen Spalten (Abschnitt 2) wirft die Frage auf, in welcher Art XML-Werte zwischen dem Datenbanksystem und den Anwendungsprogrammen ausgetauscht werden sollten. Die SQL-Norm sieht hierfür zwingend die Übergabe in Form von Zeichenketten vor. So werden XML-Werte beim Auslesen aus der Datenbank implizit oder explizit mittels der SQL/XML-Serialisierungsfunktion `XMLSERIALIZE` in Zeichenketten serialisiert und dann als VARCHAR- oder CLOB-Werte an das Anwendungsprogramm übergeben. Die entgegengesetzte Richtung (das Einbringen von XML-Werten in die Datenbank) erfolgt analog: Das Anwendungsprogramm übergibt die XML-Werte serialisiert als VARCHAR- bzw. CLOB-Werte an das Datenbanksystem, dort werden diese dann implizit oder explizit mittels der SQL/XML-Funktion `XMLPARSE` in Werte des Datentyps XML umgewandelt.

Außer der zuvor beschriebenen Form der Übergabe als Zeichenkette, die im Folgenden als *textbasierte Übergabe* bezeichnet wird, sind prinzipiell auch andere Übergabearten denkbar:

- binäre Übergabe des XML-Wertes „als Ganzes“ in seiner „verzeigerten Gestalt“ (z. B. als Baum im Fall eines XML-Dokuments bzw. als verkettete Liste von Bäumen im Fall einer Sequenz von XML-Elementen)

- getrennte Übergabe von Strukturinformationen und eigentlichen Daten, wobei beispielsweise boolesche und numerische Daten in ihrer natürlichen Form — und nicht als Zeichenketten — übergeben werden

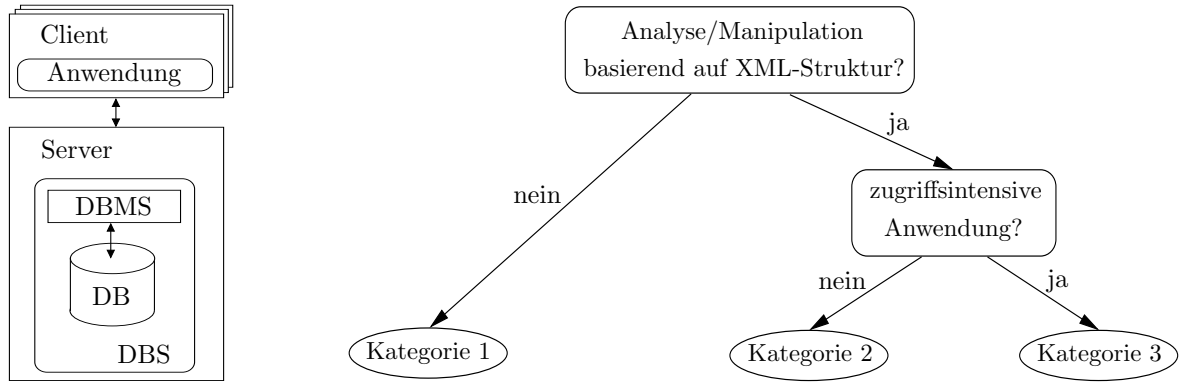
Diese (und gegebenenfalls weitere) *nicht-textbasierten* Übergabearten sollen in künftigen Forschungsarbeiten näher betrachtet und evaluiert werden. Solche nicht-textbasierten Übergabeformen unterscheiden sich von der textbasierten Übergabe u. a. durch folgende beiden Eigenschaften:

1. kompaktere Übertragung
   - Boolesche und numerische Werte werden nicht in die (i. d. R. speicherintensivere) textuelle Darstellung überführt.
   - Während in der Zeichenkettenrepräsentation eines XML-Elements der Elementtypname redundant im Start-Tag und End-Tag angegeben wird, treten derartige Redundanzen hier nicht auf.
2. kein Parsen der textuellen Repräsentation
   - Das Anwendungsprogramm kann direkt auf den Daten des XML-Wertes arbeiten, ohne zuvor seine Zeichenkettenrepräsentation parsen und Datentypkonvertierungen vornehmen zu müssen.

Ob eine textbasierte oder eine nicht-textbasierte Übergabe genutzt werden sollte, hängt von der Charakteristik des Anwendungsprogramms ab. Im nächsten Abschnitt wird eine Klassifizierung von Anwendungsprogrammen eingeführt, anhand derer die Eignung beider Ansätze betrachtet wird.

4 Klassifizierung von Anwendungsprogrammen

Als Ausgangspunkt der vorzunehmenden Klassifizierung dienen Anwendungen, die auf Tabellen mit Spalten des SQL-Datentyps XML zugreifen, um XML-Werte aus der Datenbank auszulesen bzw. in diese einzubringen. Dabei wird eine einfache Client/Server-Architektur [Dad96] vorausgesetzt, welche aus einem Server mit Server-Datenbanksystem (DBS) sowie Clients mit Anwendungsprogrammen besteht (Abbildung 1a).



a) Architektur b) Klassifizierung

Abbildung 1: Client/Server-Architektur sowie Klassifizierung der Anwendungsprogramme

Die Abbildung 1b zeigt die in diesem Beitrag vorgeschlagene Klassifizierung der Anwendungsprogramme. Für eine gegebene Anwendung wird zunächst betrachtet, ob sie auf den ausgelesenen/einzubringenden XML-Werten Analysen bzw. Manipulationen durchführt, welche auf der zu Grunde liegenden XML-Strukur beruhen. Dies ist beispielsweise der Fall, wenn auf den XML-Werten XQuery-Ausdrücke ausgewertet werden oder wenn navigierend (z. B. mittels DOM-Operationen) auf die XML-Werte zugegriffen wird. Anwendungen, auf die dies *nicht* zutrifft, werden der *Kategorie 1* zugeordnet. Derartige Anwendungen sind i. A. dadurch gekennzeichnet, dass sie die ausgelesenen XML-Werte unverändert an den Nutzer oder andere Anwendungen weiterreichen bzw. von anderen Anwendungen entgegengenommene XML-Daten unverändert in die Datenbank einbringen. Da der mit den anderen Anwendungen vorgenommene Datenaustausch i. d. R. über eine *textuelle* Repräsentation der XML-Daten abgewickelt wird — genau dies ist das Grundprinzip des XML-basierten Austauschs von Daten — bietet sich hier die *textbasierte* Übergabe von XML-Werten zwischen Datenbanksystem und Anwendungsprogramm an.

Anwendungen, die auf den XML-Werten XML-basierte Analysen oder Manipulationen durchführen und somit nicht in die Kategorie 1 fallen, lassen sich dahingehend weiter unterscheiden, wie häufig sie bei ihren Analysen bzw. Manipulationen auf die entsprechenden XML-Daten zugreifen. „Zugriffsarme“ Anwendungen werden der *Kategorie 2* zugeordnet, die verbleibenden Anwendungen bilden die *Kategorie 3*. Wo genau die Grenze zwischen zugriffsarmen und zugriffsintensiven Anwendungen zu ziehen ist, oder ob eine noch feinere Einteilung zweckmäßig wäre, soll hier (auf Grund der Platzbeschränkung) nicht näher betrachtet werden.

Um dem Grundgedanken zu entsprechen, Datenauswertungen und -manipulationen möglichst zentral vom DBMS ausführen zu lassen, ist zur Unterstützung der Anwendungsprogramme aus Kategorie 2 zu untersuchen, inwieweit XML-Funktionalität (XML-basierte Analyse- und Manipulationsmöglichkeiten) direkt in SQL integriert werden kann. Falls die so angestrebte Verlagerung der XML-basierten Verarbeitung ins DBMS nicht vollständig möglich ist, sollte wegen der im Abschnitt 3 genannten Vorteile eine *nicht-textbasierte* Form der Übergabe von XML-Daten zwischen Datenbanksystem und Anwendungsprogramm gewählt werden.

Für Anwendungsprogramme der Kategorie 3 kommt eine komplette Verlagerung der XML-basierten Analysen und Manipulationen ins Server-DBMS aus Last- bzw. Skalierungsgründen meist nicht in Frage (vgl. [HR01]). Für diese zugriffsintensiven Anwendungen ist es stattdessen notwendig, eine lokale — d. h. clientseitige — Verarbeitung der XML-Daten zu ermöglichen. Diese Verarbeitung könnte einerseits durch die Anwendung selbst oder andererseits durch ein mit XML-Funktionalität ausgestattetes client-seitiges Datenbanksystem erfolgen. Im letzteren Fall müsste die in der Abbildung 1a gezeigte Architektur um ein entsprechendes „XML-fähiges“ Client-DBS erweitert werden. Sofern damit eine komplette Verlagerung der XML-basierten Verarbeitung vom Anwendungsprogramm ins Client-Datenbanksystem jedoch nicht möglich ist, sollte, wie auch im zuerst genannten Fall, (analog zur Betrachtung der Kategorie 2) eine *nicht-textbasierte* Übergabeform bevorzugt werden.

5 Zusammenfassung und Ausblick

Die SQL-Normierungsgremien haben die Notwendigkeit erkannt, XML-Daten in SQL zu berücksichtigen. Der in SQL:2003 neu eingeführte Basisdatentyp *XML* erlaubt die Definition von Tabellen mit XML-wertigen Spalten. Dies wirft zwangsläufig die Frage auf, wie XML-Werte zwischen Datenbanksystem und Anwendungsprogrammen zu übergeben sind. Die von der SQL-Norm vorgesehene textbasierte Übergabeform wurde im Beitrag vorgestellt. Ferner wurden Ideen für eine alternative, nicht-textbasierte Übergabe skizziert. Die Eignung beider Übergabeansätze hängt wesentlich von den Anwendungscharakteristika ab. Aus diesem Grund wurde im Beitrag eine Klassifizierung der Anwendungsprogramme in drei Kategorien vorgeschlagen, auf deren Grundlage die Eignung der Übergabeformen betrachtet wurde.

Die Anwendungsklassifizierung sollte in künftigen Forschungsarbeiten konkretisiert und gege-

benenfalls weiter verfeinert werden. Die nicht-textbasierten Übergabeformen sind zu präzisieren und auf Basis der (weiterentwickelten) Klassifizierung zu evaluieren. Zudem ist zu untersuchen, inwieweit XML-Funktionalität in SQL integriert werden kann bzw. wie die Client/Server-Architektur zur Unterstützung zugriffsintensiver Anwendungen zu erweitern ist.

Literatur

[Bro04] M. Brosemann. Vergleich und Bewertung der XML-Funktionalität heutiger Datenbanksysteme. Studienarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena. In Vorbereitung, 2004.

[Dad96] P. Dadam. *Verteilte Datenbanken und Client/Server-Systeme: Grundlagen, Konzepte und Realisierungsformen*. Springer-Verlag, Berlin, Heidelberg, 1996.

[HR01] T. Härder und E. Rahm. *Datenbanksysteme — Konzepte und Techniken der Implementierung*. Springer-Verlag, Berlin, Heidelberg, 2. Auflage, 2001.

[IBM02] IBM Corporation. *IBM DB2 Universal Database — XML Extender Administration and Programming (Version 8)*, 2002.

[ISO03a] International Organization for Standardization, Genf. *Information technology – Database languages – SQL – Part 14: XML–Related Specifications (SQL/XML)*, Dezember 2003. ISO/IEC 9075-14:2003, International Standard (IS).

[ISO03b] International Organization for Standardization, Genf. *Information technology – Database languages – SQL – Part 2: Foundation (SQL/Foundation)*, Dezember 2003. ISO/IEC 9075-2:2003, International Standard (IS).

[KM03] M. Klettke und H. Meyer. *XML & Datenbanken : Konzepte, Sprachen und Systeme*. dpunkt-Verlag, Heidelberg, 1. Auflage, 2003.

[MKF+03] J.-E. Michels, K. Kulkarni, C. M. Farrar, A. Eisenberg und N. Mattos. The SQL Standard. In *it – Information Technology*, 1/2003, Seiten 30–38. Oldenbourg Wissenschaftsverlag, München, Februar 2003.

[Ora03] Oracle Corporation. *Oracle XML DB Developer's Guide, 10g Release 1 (10.1)*, Dezember 2003.

[Sch03] H. Schöning. XML – Editorial. In *it – Information Technology*, 3/2003, Seiten 121–122. Oldenbourg Wissenschaftsverlag, München, Juni 2003.

[See03] M. Seemann. *Native XML–Datenbanken im Praxiseinsatz*. Software & Support Verlag, 1. Auflage, 2003.

[Sof03] Software AG. *Tamino Version 4.1.4 — Introducing Tamino*, 2003.

[Tür03] C. Türker. *SQL:1999 & SQL:2003 : objektrelationales SQL, SQLJ & SQL/XML*. dpunkt-Verlag, Heidelberg, 1. Auflage, 2003.

[W3C03] World Wide Web Consortium. *XQuery 1.0: An XML Query Language*, November 2003. W3C Working Draft.

[W3C04a] World Wide Web Consortium. *Document Object Model (DOM) Level 3 Core Specification, Version 1.0*, Februar 2004. W3C Proposed Recommendation.

[W3C04b] World Wide Web Consortium. *Extensible Markup Language (XML) 1.0 (Third Edition)*, Februar 2004. W3C Recommendation.