

# Integration relationaler Datenstrukturen in XML-Applikationen – Database to XML Servlet

T. Courvoisier, G. Flach

tc,gf@rostock.zgdv.de

## Zusammenfassung

Strukturierte und semistrukturierte Daten werden zunehmend in Form von XML-Dokumenten gespeichert und übertragen. Neben der Bedeutung von XML als universelles Datenaustauschformat bietet gerade die Trennung von Struktur und Repräsentation neue Anwendungsmöglichkeiten. Dies betrifft insbesondere die Möglichkeiten der Publikation von Datenbeständen. Die Tatsache, daß ein Großteil der Daten von Unternehmen in relationalen Datenbanksystemen gespeichert ist, macht eine Einbeziehung auch dieser Daten in XML-basierte Systeme notwendig. Der Ansatz, der in diesem Beitrag vorgestellt wird, wird es ermöglichen, Ergebnisse von SQL-Anfragen direkt an XML-Applikationen weiterzuleiten.

## 1 Einleitung

Im Rahmen des Verbundforschungsprojektes Business-MV<sup>1</sup> [CGF00, HWH00] realisiert das ZGDV Rostock unter anderem das Produktinformationssystem ProInfo-MV. Dieses Produktinformationssystem ist ein internetbasierter Informationsdienst für Unternehmen und Institutionen, der die datenbankbasierte und anwendungsspezifische Recherche und Verwaltung von multimedialen Produktinformationen, digitalen Archiven und anderen Informationsangeboten erlaubt.

Ziel war unter anderem die personalisierte Präsentation von Produkten, Projekten und Service der beteiligten Unternehmen. Eine vielfältige Transformation und Publikation von Datenbeständen legte den Einsatz von Tools im Publishingbereich nahe. Insbesondere SGML und XML boten sich als Technologie zur Dokumentenproduktion an. Gerade im Bereich von XML gibt es eine große Anzahl meistens frei verfügbarer Tools. Um diese für den Inhalt der Datenbank verwendbar zu machen, war die Schaffung einer entsprechenden Schnittstelle nötig.

Obwohl eine objektrelationale oder objektorientierte Struktur in der Datenbank für die Arbeit mit SGML- oder XML-Dokumenten günstiger wäre als eine rein relationale Struktur, war es nötig, mit relationalen Daten zu arbeiten. Als Basis für den Datenbestand dienten Firmendaten von Kammern und Institutionen, die in relationaler Form vorlagen. Eine Erweiterung dieser Daten durch multimediale Inhalte mußte aufgrund der Wahrung der Kompatibilität zu laufenden Anwendungen ebenfalls eine relationale Datenstruktur haben.

Eine portable XML-Schnittstelle zu relationalen Datenbanken verspricht außerdem einen hohen Wiederverwendungsgrad, da immer mehr Applikationen auch im Datenbankbereich XML in den Gebieten Content-Management, Data-Mining usw. einsetzen.

---

<sup>1</sup>gefördert durch das Wirtschaftsministerium des Landes Mecklenburg-Vorpommern; Projektpartner: Fraunhofer IGD, Universität Rostock, MEDEOCOM GmbH, ZGDV Rostock.

## 2 XML im Überblick

XML [Con98a] wird zunehmend die dominierende Darstellungsform von Dokumenten und Daten im Web sein. Sie erlaubt die Definition einer anwendungsspezifischen Grammatik von Auszeichnungssprachen bestehend aus Elementen und deren Attributen. Aufbauend auf XML werden momentan verschiedene Auszeichnungssprachen, beispielsweise für Formeln, Multi-Mediapräsentationen oder Datenbankanfragen, standardisiert. XML ist keine erweiterbare Auszeichnungssprache - sie ist eine standardisierte Sprache in der sich die Syntax von Auszeichnungssprachen notieren läßt. Formaler ausgedrückt ist XML eine Metagrammatik für kontextfreie Grammatiken - also die Grammatik einer Sprache, mit der sich die Regeln von Grammatiken notieren lassen. Diese Grammatikdefinition wird im XML-Umfeld Document Type Definition (DTD) genannt. Dokumente werden so durch ihre Grammatik typisiert, d.h. die Sätze der durch die DTD definierten formalen Sprache bilden mögliche Dokumente.

In dieser DTD können die Markup-Elemente des Dokumentes definiert werden. Jedes Element kann wieder Elemente oder Zeichendaten enthalten. Außerdem können den einzelnen Elementen Attribute verschiedener Typen zugeordnet werden. Zusätzlich zu Elementen können sogenannte Entities definiert werden.

Die Zeichendaten in Elementen und Entities können als strukturiert (parsed) oder unstrukturiert betrachtet werden. Die Möglichkeit, auch unstrukturierte Daten in ein Dokument aufzunehmen, ermöglicht die Einbindung von Rasterbildern oder Binärobjekten.

Einer der Hauptunterschiede zwischen XML und SGML ist, daß XML-Dokumente auch ohne DTD verwendet werden können. Man unterscheidet deshalb zwischen validierten XML-Dokumenten, die gegen eine DTD geprüft wurden, und wohlgeformten Dokumenten, die lediglich den Syntaxregeln entsprechen. Das hat zur Folge, daß alle Tags in XML selbsterklärend sein müssen, während Tags in SGML unter Umständen ohne Start- bzw. End-Tag geschrieben werden konnten (z.B. `<BR >` in HTML).

Wohlgeformte XML-Dokumente stellen durch die Schachtelung von Tags Bäume dar. Die Darstellung eines solchen Dokuments in einem Browser ist in der Regel die Transformation dieses Baumes in einen anderen Baum: den durch visuell geschachtelte Darstellungsflächen repräsentierten Baum. Die Extensible Stylesheets Language XSL stellt einen allgemeinen Mechanismus für die Transformation von XML-Bäumen bereit und wird momentan als Standard entwickelt [Con99d]. XSL hat zwei Bestandteile. Eine allgemeine Sprache zur Transformation von Bäumen XSLT und eine Sprache zur Notation von in Flächen dargestellten Bäumen, die aus Formatted Objects gebildet werden. Beim Datenaustausch mit XML ist die inhaltliche und die visuelle Komponente eines Dokumentes getrennt. Deshalb kann ein XML-Dokument in einem Browser nur dargestellt werden, wenn dazu ein Stylesheet existiert. In diesem Stylesheet wird beschrieben, wie die einzelnen Tags im Browser dargestellt werden sollen. Ein solches Stylesheet kann auf alle Dokumente einer Klasse angewendet werden. Auf der anderen Seite können mehrere verschiedene Stylesheets für ein und dieselbe Datenmenge verwendet werden. Das ermöglicht eine gleichzeitige Publikation der Daten auf unterschiedlichen Medien.

Über die Form der Stylesheets ist man sich allerdings noch nicht vollkommen einig. Ein Weg besteht darin, Cascading Stylesheets, wie es sie auch für HTML gibt, auf XML-Daten anzuwenden. Der zweite Weg ist, die SGML Stylesprache DSSSL zu nutzen. Diese Methode ist dank kostenloser Softwaretools sehr verbreitet. Allerdings ist die etwas unübersichtliche LISP-ähnliche Syntax gewöhnungsbedürftig. Im W3 Konsortium arbeitet man an der Spezifikation der Extensible Stylesheet Language (XSL). XSL selbst ist eine Anwendung von XML, ist also durch eine DTD beschrieben. Der Internetexplorer in der Version 5 setzt auf XSL und CSS. Im W3 Konsortium ist die Spezifikation von XSL in zwei Teile geteilt. Ein Teil ist bereits als Recommendation verabschiedet und zwar der Transformationsteil von XSL, mit dem es

möglich ist, ein XML-Dokument in ein anderes zu transformieren. Da HTML<sup>2</sup> sowohl eine SGML- als auch eine XML-Anwendung ist, ermöglicht XSLT auch die Transformation eines XML-Dokumentes in HTML.

Der zweite Teil der XSL-Spezifikation beinhaltet die Transformation eines XML-Dokumentes in einen Baum von Formatobjekten (formatting objects). Mit Hilfe solcher Formatobjekte können dann vor allem Printmedien (PDF, PS ...) erzeugt werden.

### 3 XML und Datenbanken

Der ursprüngliche Zweck von XML war es, die Fähigkeit von Applikationen dahingehend zu erweitern, daß Dokumente aus dem Internet besser verarbeitet werden können. Aus der Sicht von Datenbankanwendungen ergeben sich auch noch andere Aspekte der Nutzung von XML. Mit Daten, die in XML gespeichert sind, wäre es möglich, Anfragen an den Inhalt solcher Dokumente zu stellen. Es ist denkbar, Teildokumente aus einer Menge von XML-Dokumenten zu extrahieren, zu synthetisieren und den Inhalt zu analysieren. Deshalb nutzen gerade Anbieter von Content-Management-Systemen den Schub, den die offensive Verbreitung der XML-Technologie bringt. Forschungen auf diesem Gebiet resultieren vor allem in Spezifikationen von semistrukturierten Datenmodellen und Anfragesprachen wie XQL [Con98c], XML-QL [Con98b] und IRQL [HP99].

Die Vorteile von semistrukturierten XML-Dokumenten lassen sich in Datenbanken natürlich am besten nutzen, wenn das entsprechende Datenbanksystem auch intern XML-Strukturen verarbeitet. So verwundert es nicht, daß vor allem nicht-relationale Datenbanksysteme entsprechende Lösungen anbieten. Beispielhaft seien hier die kommerziellen Produkte Tamino [AG99], Excelon [Rub00] und Poet [Sof] genannt. Ein Nachteil bei der Verarbeitung von XML-Dokumenten in Datenbanksystemen ist die aus dem Publishing-Bereich kommende Begrenzung der Dateninhalte auf Zeichendaten. Es existieren aber Bestrebungen, XML durch entsprechende Schemadefinitionen zu einer vollwertigen Datenrepräsentationssprache zu erweitern [Con99c].

Auch im Bereich von e-commerce spielt XML als elektronisches Datenaustauschformat (EDI) eine zunehmende Rolle. Verwiesen sei hier auf Aktivitäten wie BizTalk und OASIS [Wal00]. Datenbanken spielen auf diesem Gebiet aufgrund von Zuverlässigkeit und Transaktionssicherheit eine bedeutende Rolle. Da ein beträchtlicher Teil der Business-Daten in relationalen Datenbanksystemen gespeichert ist, ist ein Transfer zwischen relationalen Datenbanken und XML-Dokumenten ein essentielles Problem.

Die Möglichkeiten der Speicherung von XML-Dokumenten in relationalen oder objekt-relationalen DBMS wurde bereits von mehreren Autoren diskutiert [STH<sup>+</sup>99]. Die Existenz einer DTD ist für die Generierung von relationalen Schemata eine zwingende Voraussetzung. Schwierigkeiten entstehen dabei einmal mit der Komplexität der Element-Spezifikation in der DTD, dem Konflikt zwischen der 2-Ebenen-Natur relationaler Schemata (Tabelle - Attribut) und mit mehrwertigen Attributen und Rekursionen. Objekt-relationale DBMS bieten aufgrund der höheren Mächtigkeit und Flexibilität des Typsystems einige Vorteile [Tim99].

---

<sup>2</sup>HTML ist eigentlich eine reine SGML-Anwendung, kann aber mit ein paar Modifikationen auch mit XSLT erzeugt werden.

## 4 Database to XML Servlet (DaS) - Konzeption

### 4.1 Datenmodelle

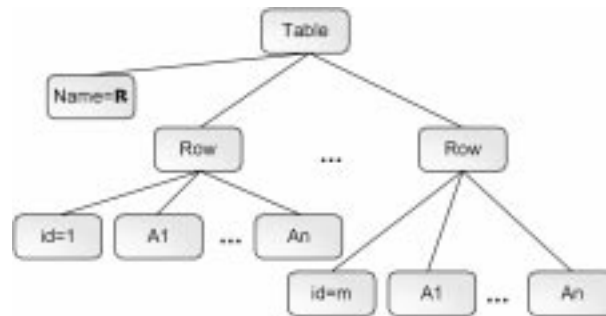
Zur Nutzung relationaler Daten in XML-Applikationen ist es notwendig, Daten, die in relationaler Form vorliegen, in das Datenmodell von XML abzubilden. Die Unterschiedlichkeit der beiden Datenmodelle bereitet dabei Probleme.

Im Relationenmodell werden alle Daten als Teilmenge eines kartesischen Produkts von Mengen dargestellt [Cod70]. Die daraus resultierende zweidimensionale Tabellenstruktur ist nur bedingt in der Lage, semantische und strukturelle Zusammenhänge zwischen den Daten auszudrücken. Diese Zusammenhänge werden vor allem durch die Definition von lokalen und globalen Integritätsbedingungen dargestellt. XML dagegen eignet sich sehr gut zur Darstellung gerade von strukturellen Zusammenhängen von Daten. Auf der anderen Seite kennt XML, aus dem Publishingbereich kommend, keine Datentypen. Lediglich Zeichendaten (character data) können dargestellt werden.

Die einfachste Methode, relationale Daten in XML-Strukturen zu überführen, besteht in der direkten Abbildung der Tabellen. Dabei wird jeder Wertebereich auf Zeichendaten abgebildet und die Tabellenstruktur 1:1 überführt.

$D \rightarrow \#PCDATA|D$  ist beliebiger atomarer Wertebereich (Domäne)

$\mathfrak{R} = \{A_1, \dots, A_n\} \rightarrow$



Diese Methode wird so oder so ähnlich in den meisten Tools zur Umwandlung relationaler Tabellen in XML-Dokumente angewandt. So funktionieren unter anderem auch die XSQL-Pages von Oracle [Cor00] nach diesem Prinzip. Das Problem dabei ist, daß die Tabellen selber, wie oben bereits beschrieben, nur *ein* Teil des Relationenmodells sind.

Um auch die Integritätsbedingungen des Relationenmodells in das Zieldokument einfließen zu lassen, gibt es mehrere Möglichkeiten. Auch hier sei die einfachste zuerst genannt; sie besteht darin, Tabellen und Attribute so in das XML-Dokument abzubilden, daß entsprechende Integritätsbedingungen und Datentypen als Metainformationen bereit gestellt werden. Einen Vorschlag für eine solche Vorgehensweise findet sich auf der Webseite des W3C [Bos97]. Das Java-Tool DB2XML [Tur99] setzt diesen Vorschlag um.

Als beispiel sei Tabelle  $t$  gegeben:

A	B	C
1	'Hallo'	true
2	'Ciao'	false

Die Anfrage  $tquery = 'select * from t'$  wird in folgende DTD gemappt:

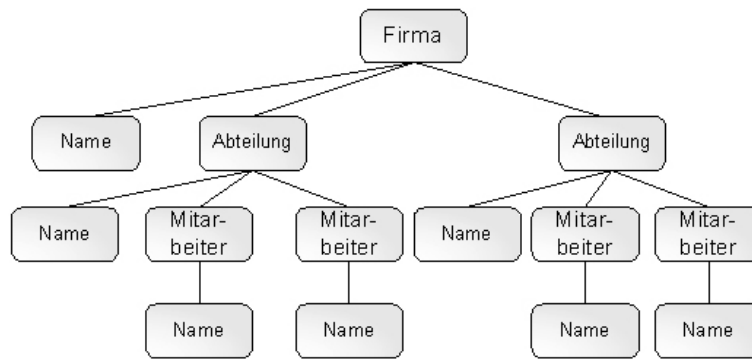


Abbildung 1: Vater-Kind-Beziehungen zwischen XML-Elementen

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE database [
  <!ELEMENT database (t)>
    < !ATTLIST database URL CDATA #REQUIRED>
  <!ELEMENT tquery (tquery_rec*)>
    <!ATTLIST tquery
      QUERY CDATA #REQUIRED
      PKEY CDATA #IMPLIED>
  <!ELEMENT tquery_rec (a, b, c)>
  <!ELEMENT a #PCDATA>
    <!ATTLIST a
      TYPE (TEXT|CURRENCY|DATETIME|LONG) #FIXED "LONG"
      NAME CDATA #FIXED "a"
      NULLABLE (true|false) #FIXED "false"
      ...
    >
  ...
]>

```

Diese Vorgehensweise eignet sich gut, um relationale Daten zwischen Datenbanksystemen auszutauschen. Andererseits bieten XML-Applikationen keine Mechanismen, Integritätsbedingungen von XML-Elementen zu verarbeiten. Es ist also eine Umsetzung von datenbankspezifischen Integritätsbedingungen in XML-spezifische Strukturen anzustreben. Der Fokus liegt dabei auf semantischen und strukturellen Zusammenhängen zwischen Objekten und Eigenschaften.

XML bietet zwei Möglichkeiten zur Darstellung von Zusammenhängen zwischen Objekten. Zum Ersten können durch die Baumstruktur von XML-Dokumenten die Vater-Kind-Beziehungen zwischen den einzelnen Knoten eine Zuordnung zwischen Objekten ermöglichen (Abbildung 1).

Als zweiten Mechanismus zur Darstellung von Beziehungen zwischen Elementen bietet die XML-Spezifikation die Möglichkeit der Definition von Objektidentitäten, die referenziert werden. Dazu können für jedes Element Attribute vom Typ *ID* und *IDREF* bzw. *IDREFS* angegeben werden. Dadurch kann auch der Zusammenhang zwischen Objekten über die Baumstruktur hinweg dargestellt werden (Abbildung 2).

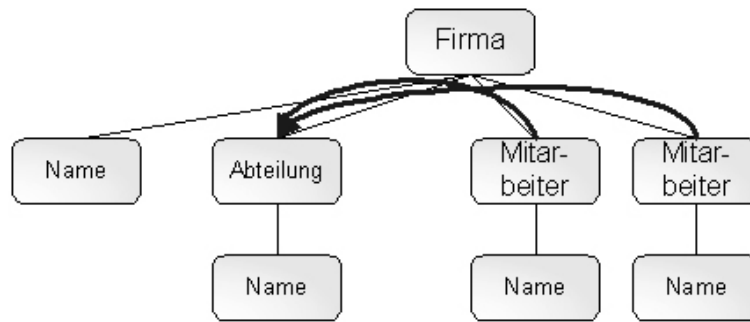


Abbildung 2: Referenzen zwischen XML-Elementen

## 4.2 XML-SQL

Als günstigste Variante zur Überführung von Relationen in XML-Dokumente erweist sich die Spezifikation der Abbildung durch den Anwender bzw. Entwickler selbst. Dazu werden Sprachkonstrukte benötigt, die eine Beschreibung der Vorgehensweise ermöglichen. Solche sogenannten Mapping-Sprachen werden zum Beispiel im XML-DBMS Projekt an der TU Darmstadt [Bou00] und im XML Lightweight Extractor (XLE) von IBM Alphaworks [Alp00] verwendet.

Beim XLE wird eine DTDSA<sup>3</sup>-Datei angelegt, in der das Mapping mittels SQL-Anweisungen beschrieben werden kann. Auf diese Weise können Anfragen an die Datenbank erstellt werden und auf fast beliebige DTDs abgebildet werden. Für das Beispiel in Abbildung 1 sieht eine entsprechende DTDSA, bei Annahme der Tabellen *firma*, *abteilung* und *mitarbeiter* in der Datenbank, folgendermaßen aus:

```

<!DOCTYPE firma[
<!ELEMENT firma (name)
  :: f := SQL("select * from firma where firma_id=$in0")>
<!ELEMENT abteilung (aname)
  :: a := SQL("select * from abteilung
              where firma=$f.firma_id")>
<!ELEMENT mitarbeiter (mname)
  :: m := SQL("select * from mitarbeiter
              where abteilung=$a.abteilung_id")>
<!ELEMENT name (#PCDATA :field(firma, name, f))>
<!ELEMENT aname (#PCDATA :field(abteilung, name, a))>
<!ELEMENT mname (#PCDATA :field(mitarbeiter, name, m))>
]>

```

Wie in der Einleitung beschrieben, war im vorliegenden Anwendungsfall höchste Kompatibilität zu bestehenden Anwendungen gefordert. Das ermöglicht auf der anderen Seite auch die Nutzung eben dieser Anwendungen. Dieses zu ermöglichen und z.B. automatisch generierte SQL-Anweisungen aus anderen Anwendungen zu nutzen, bedurfte einer speziellen Mapping-Sprache, die XML-SQL genannt wurde. Konstrukte dieser Sprache haben die Form

<sup>3</sup>Wofür die Abkürzung DTDSA genau steht, war der Dokumentation leider nicht zu entnehmen.

```

<SQL-Statement>
CONSTRUCT
<XML-Tree>
[ROOT=<rootName>].

```

Dabei steht  $\langle SQL - Statement \rangle$  für jede beliebige SQL-Anweisung. Das Ergebnis einer SQL-Anweisung ist immer *eine* Relation. Mit Hilfe von  $\langle XML - Tree \rangle$  wird aus dieser Relation das Ergebnis konstruiert. Das Konstrukt  $\langle XML - Tree \rangle$  beschreibt dabei den Teilbaum des Zieldokuments, der für genau ein Objekt im Anfrageergebnis angelegt wird. Da jedes XML-Dokument ein einziges Wurzelement haben muß [Con98a], wird dieses standardmäßig mit „*XMLSQLError*“ angelegt. Ist die Variable *ROOT* jedoch definiert worden, so bestimmt sie den Namen des Wurzelements. Die *CONSTRUCT*-Klausel mit dem XML-Teilbaum ist der Spezifikation von XML-QL [Con98b] entlehnt. Eine Besonderheit besteht in der Verwendung der Attribute *ID* und *IDREF*. Jeder Tag, außer die Wurzel des Teilbaumes, kann diese beiden Attribute enthalten. Sie dienen dazu, Teilobjekte und ihre Beziehungen zu kennzeichnen. Die Werte der einzelnen Attribute der Ergebnisrelation können als Variablen unter dem Namen, den sie auch im relationalen Ergebnis haben, angesprochen werden. Zur Kennzeichnung einer solchen Variablen dient das Dollarzeichen.

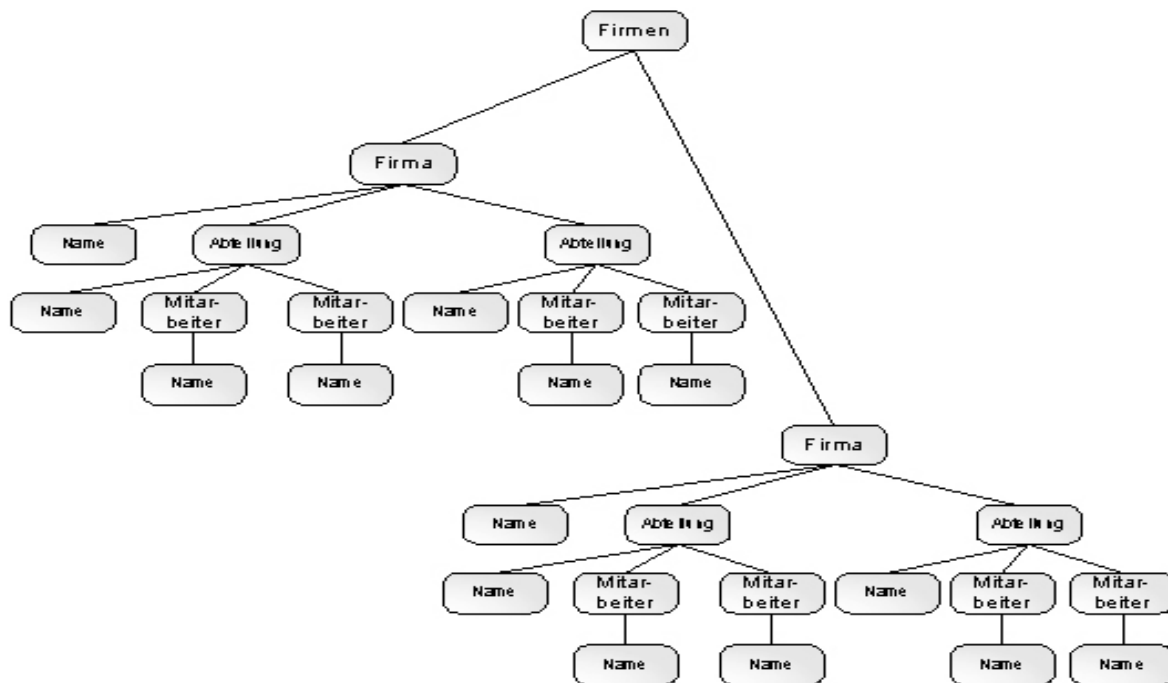
Beispiel:

```

select *      from   firma a, abteilung b, mitarbeiter c
              where  a.firma_id=b.firma_id
                    and b.abteilung_id=c.abteilung_id
CONSTRUCT
  <firma ID="firma$a.firma_id">
    <name>$a.name</name>
    <abteilung ID="a$b.abteilung_id" IDREF="firma$a.firma_id">
      <name>$b.name</name>
      <mitarbeiter IDREF="a$b.abteilung_id">
        <name>$c.name</name>
      </mitarbeiter>
    </abteilung>
  </firma>
ROOT=Firmen

```

Das Beispiel liefert ein Ergebnis dieser Art:



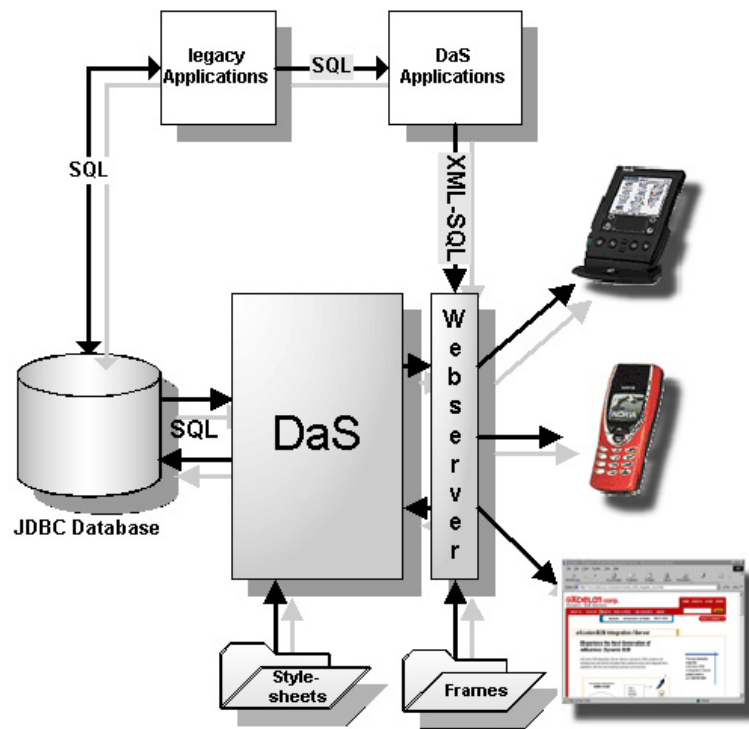


Abbildung 3: DaS - Aufbau

### 4.3 Architektur

Die Abbildung 3 zeigt den Systemaufbau der XML-SQL-Prozesse innerhalb von ProInfo (siehe Einleitung). Innerhalb des Servlets (DaS), das über einen Webserver aufgerufen wird, ist ein XML-SQL-Prozessor und ein XSLT-Prozessor realisiert. Als Parameter akzeptiert das Servlet XML-SQL-Anfragen und XSLT-Stylesheets. Diese Stylesheets können sich einerseits im Stylesheet-Repository auf dem Server befinden oder direkt als Text übergeben werden. Im Servlet wird zunächst die XML-SQL-Anfrage bearbeitet. Auf das Ergebnis wird dann das entsprechende Stylesheet angewendet. Das ermöglicht die Formatierung der Ausgabe für verschiedene Web-Clients. In einem weiteren Repository auf dem Server befinden sich HTML-Frameschablonen. Das hat den Grund, daß das Ergebnis einer XSLT-Transformation nicht geteilt werden kann, wie es aber für die Ausgabe einer HTML-Seite mit Frames notwendig wäre. Zwar kann man mit Hilfe von Javascript auch HTML-Seiten mit Frames erzeugen, die nur aus einer Datei bestehen, aber der Aufwand zur Erstellung von Schablonen ist wesentlich geringer.

Die Übergabe von Verbindungsparametern, die eine JDBC-Verbindung zu einer Datenbank ermöglichen, gestatten den Zugriff auf die entsprechende Datenbank auch von mobilen Geräten aus. Der Webserver stellt in diesem Falle die Verbindung her. Damit können zum Beispiel Daten über jeden Proxyserver für Palm-Geräte abgerufen werden.

### 4.4 Anwendung

Im Rahmen von ProInfo-MV wurde ein Palm-Client für DaS realisiert, der in der Lage ist, Anfragen an die ProInfo-Datenbank zu senden und durch die Ergebnisse zu navigieren. Wei-



Abbildung 4: DaS - Anwendungen

terhin wurde eine WML-Seite eingerichtet, die den Zugriff auf ProInfo mit einem Wap-fähigen Gerät erlaubt. Da das Senden von komplexen Parametern an den Web-Server, ein Wap-Handy überfordert, ist ein kleines CGI-Programm, das Standardanfragen und -transformationen bereitstellt, zwischen Client und DaS geschaltet.

Als dritter Client kann ein normaler Web-Browser verwendet werden, um durch die ProInfo-Webseite zu navigieren. Abbildung 4 zeigt die Screenshots<sup>4</sup> der Clients.

## 5 Zusammenfassung und Ausblick

Die Mapping-Sprache XML-SQL eignet sich gut zur Beschreibung der Abbildung relationaler Datenbestände in XML-Dokumente, unter Wahrung höchstmöglicher Kompatibilität zu bestehenden Anwendungen. Allerdings ist XML-SQL weitestgehend auf das Mapping von Anfrageergebnissen Beschränkt. Um Datenbestände im Ganzen zu erfassen, eignen sich andere Techniken besser. Anfrageergebnisse über mehrere Relationen, wie sie mit XML-SQL ausgewertet werden, erfassen nur Tupel, für die alle Verknüpfungsbedingungen in der Where-Klausel der SQL-Anweisung gelten. Selektiert man zum Beispiel alle Firmen und die dazugehörigen Abteilungen, wobei Firmendaten und Abteilungsdaten in zwei verschiedenen Relationen gespeichert sind, werden keine Firmen ausgegeben, die keine Abteilung haben. Die meisten Datenbanksysteme unterstützen zwar sogenannte Outer-Joins, ein solcher versagt aber auch bei einer dreistufigen Abhängigkeit zwischen Tabellen. XML-SQL ist also keine universelle Mapping-Sprache, sondern eben für die oben genannten Zwecke konzipiert.

Durch den Servletansatz des DaS wird eine Kommunikation zwischen mobilen Geräten und Datenbanken über ein HTTP-Protokoll ermöglicht. Diese Lösung gewährleistet eine hohe Verfügbarkeit der Daten, da mittlerweile jedes mobile Gerät über HTTP<sup>5</sup> kommunizieren

<sup>4</sup>Die Screenshots der mobilen Geräte stammen von Simulatoren.

<sup>5</sup>Bei einem Handy muß natürlich das WAP-Portal des Providers dazwischen liegen.

kann. Der Nachteil des bisherigen Ansatzes liegt einmal darin, daß die Kommunikation mit der Datenbank nur in eine Richtung geht, nämlich hin zum Client. Außerdem ist eine Datenübertragung über HTTP auch nicht sicher. Das System ist im Einsatz also mehr oder weniger auf Informationssysteme beschränkt<sup>6</sup>.

Als Aufgabe für die Zukunft wäre vielleicht ein Update von Datenbankinhalten von mobilen Geräten aus zu nennen. Die Probleme, die sich daraus ergeben werden, liegen hauptsächlich in der Datensynchronisation, der Transaktionsverwaltung und der Datenkonsistenz. Hinzu kommen beschränkte Leistungsfähigkeit und Speicherplatz mobiler Geräte.

Eine Erweiterung von DaS zum mobilen Präsentations- und Informationssystem erscheint günstiger. In Verbindung mit einem Resource Description Framework (RDF, [Con99b]) ist die Recherche über mehrere Datenbanken, Dokumente und Informationsquellen möglich. Der XML-SQL-Prozessor ist dann nur *ein* Teil der Zugriffsmethoden auf entsprechende Informationen.

## Literatur

- [AG99] Software AG. Tamino: The Information Server for Electronic Business. White paper, Software AG, 1999.
- [Alp00] IBM Alphaworks. XML Lightweight Extractor <http://www.alphaworks.ibm.com/tech/xle>, 2000.
- [Bos97] B. Bos. XML representation of a relational database <http://www.w3.org/XML/RDB.html>, 1997.
- [Bou00] Ronald Bourret. XML-DBMS, Version 1.01 <http://www.informatik.tu-darmstadt.de/DVS1/staff/bourret/>, 2000.
- [CBS94] R. H. L. Chiang, T. M. Barron und V. C. Storey. Reverse Engineering of Relational Databases: Extraction of an EER model from a Relational Database. *Data & Knowledge Engineering*, 12:107–142, 1994.
- [CF00] T. Courvoisier und G. Flach. DaS - Database to XML Servlet. *Tagungsband GI-Workshop Grundlagen von Datenbanken*, 12, 2000.
- [CGF00] T. Courvoisier, N. Günther und G. Flach. Business-MV - Internet-based Services for Database Federation, Communication and Cooperation. *Computer Graphics topics*, 2000.
- [Chi95] R. H. L. Chiang. A knowledge based system for performing reverse engineering of relational databases. *Decision Support Syst.*, 1995.
- [Cod70] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13:377–387, June 1970.
- [Con98a] World Wide Web Consortium. Extensible Markup Language (XML), Version 1.0. <http://www.w3.org/TR/1998/REC-xml-19980210>, 1998.
- [Con98b] World Wide Web Consortium. XML-QL: A Query Language for XML. <http://www.w3.org/TR/NOTE-xml-ql/>, 1998.

---

<sup>6</sup>Das rührt aus der Aufgabenstellung her.

- [Con98c] World Wide Web Consortium. XML Query Language (XQL). <http://www.w3.org/TandS/QL/QL98/pp/xql.html>, 1998.
- [Con99a] World Wide Web Consortium. Document Object Model. <http://www.w3.org/DOM/>, 1999.
- [Con99b] World Wide Web Consortium. Resource Description Framework (RDF) Model and Syntax Specification. <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222>, 1999.
- [Con99c] World Wide Web Consortium. XML Schema Part 1: Structures / XML Schema Part 2: Datatypes. [http://www.w3.org/TR/xmlschema-1\(2\)/](http://www.w3.org/TR/xmlschema-1(2)/), 1999.
- [Con99d] World Wide Web Consortium. XSL Transformations (XSLT) Specification. <http://www.w3.org/TR/1999/WD-xslt-19990421.html>, 1999.
- [Cor00] Oracle Corp. XML Developers Kit <http://www.oracle.com/xml/>, 2000.
- [FK99] D. Florescu und D. Kossmann. Storing and Querying XML Data using an RDBMS. *Bulletin of the Technical Committee on Data Engineering*, 22(3):27–34, 1999.
- [HP99] A. Heuer und D. Priebe. IRQL - Yet another language for querying semistructured data? *Universität Rostock, Fachbereich Informatik*, 1999.
- [HWH00] A. Heuer, G. Weber und C. Herzig. Das Zusammenspiel von Agenten- und Föderationstechniken innerhalb von Business-MV. *Tagungsband GI-Workshop Grundlagen von Datenbanken*, 12, 2000.
- [KM99] M. Klettke und H. Meyer. Managing XML documents in object-relational databases. Technical report, Universität Rostock, Fachbereich Informatik, 1999.
- [Por99] B. Porst. *Untersuchungen zu Datentypenweiterungen für XML-Dokumente und ihre Anfragemethoden am Beispiel von DB2 und Informix*. Diplomarbeit, Universität Rostock, Fachbereich Informatik, 1999.
- [Rub00] D. Rubinstein. Excelon Eases B-to-B Integration, Unifies Process. *Software Development Times*, June 2000.
- [Sof] POET Software. Website of POET Software <http://www.poet.de>.
- [STH<sup>+</sup>99] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, und J. Naughton. Relational Databases for Querying XML Documents: Limitations and Opportunities. *Proc. of the 25th VLDB Conference*, 1999.
- [Tim99] J. Timm. *Speicherung von XML-Dateien in Objekt-relationalen Datenbanken*. Diplomarbeit, Universität Rostock, Fachbereich Informatik, 1999.
- [Tur99] V. Turau. Making Legacy Data Accessible for XML Applications. Paper 331, FH Wiesbaden, 1999.
- [Wal00] D. Wall. XML Gets Businesses Talking. *XML Magazine*, 2000.