Zugriff auf multimediale XML-Dokumente in Objekt-relationalen Datenbanksystemen

Diplomarbeit

Universität Rostock, Fachbereich Informatik
Lehrstuhl für Datenbank- und Informationssysteme

vorgelegt von: Christoph Dittberner
geboren am 05.11.1973 in Rostock

Betreuer: Prof. Dr. A.Heuer
Prof. Dr. B. Urban
Dr. H. Meyer
Dipl.-Inf. G. Flach
Dipl.-Inf. T. Courvoisier

Abgabedatum: 19.03.2001

Zusammenfassung

In den letzten Jahren hat die Thematik wireless-orientierter, multimedialer Content Management Systeme zunehmend an Bedeutung gewonnen. In diesem Kontext kann die Extensible Markup Language (XML) zur Verbesserung einer großen Bandbreite möglicher Anwendungen beitragen. Die abgeleiteten Anwendungen reichen vom vereinfachten Dokumentenmanagement über die effizientere Gestaltung web-basierter Dienste, der einheitlichen Aufbereitung von Parametern und Repository-Daten, der Unterstützung von Semantic Data Streams zum Nachrichtenaustausch zwischen Business Objects bis hin zur Vereinfachung typischer Aufgaben aus dem Bereich E-Commerce z.B. der Automatisierung zwischenbetrieblicher Kommunikation. XML als erweiterbare, offene Trägertechnologie kann somit in den folgenden Bereichen eingesetzt werden: Anwendungen von Datenintegration, Datenaustausch zwischen Anwendungen, Client-orientierte Datenpräsentation und –manipulation sowie Content Management und Publishing.
Diese Arbeit widmet sich insbesondere den beiden letztgenannten Anwendungsbereichen.
Benötigt wird in diesem Zusammenhang eine Schnittstelle zum jeweiligen Datenbanksystem, die eine Nutzung der objekt-relationalen Multimedia-Fähigkeit zuläßt und die Ableitung der jeweiligen anwendungs- und endgeräteabhängigen Präsentation ermöglicht. Die zugehörige XML-basierte Request-Bearbeitungs- und Anfrage-Transformations-Strategien werden diskutiert. Die vorgestellten Konzepte wurden praktisch realisiert und verifiziert. Die Ergebnisse der Laufzeittests der erstellten Prototypen bestätigen die in dieser Arbeit vorgestellten Konzepte für den Zugriff auf multimediale XML-Dokumente in objekt-relationalen Datenbanksystemen.

Abstract

The subject of wireless-oriented multimedia **C**ontent **M**anagement **S**ystems (CMS) gained increasingly in importance in the recent years. In this context, the E**X**tensible **M**arkup **L**anguage (XML) can contribute to an improvement for a large bandwidth of practicable applications. The derived applications reach from the simplified document management, via the more efficient designing of web-based services, the homogeneous preparation of parameters and repository-data and the supporting of semantic data streams for information exchange between business objects, to the simplification of typical tasks from the field of E-commerce, for instance the automatization of the inter-operating communication. Consequently, XML as an extensible open-carrier technology can be used in the following areas: applications of data intergration, data exchange between applications, client-oriented data presentation and manipulation as well as content management and publishing.

This paper devotes especially to the two last-named application fields. In this connection it is reqiured an interface to the corresponding database system that allows the usage of object-relational multimedia ability and the derivation of the respective application- and end-device-based presentation. The corresponding XML-bases request processing and inquiry transformation strategies are discussed. The presented concepts were practically implemented and virtually verified. The running-time test results of the produced prototyps confirm the in this paper presented concepts for the access to multimedial XML-documents in object-relational database systems.

CR-Klassifikation

E.2 Data Storage Representations

H.2.0 Database Management General

H.2.4 Database Management Systems

H.2.5 Heterogeneous Databases

H.2.8 Database applications

H.3.3 Information Search and Retrieval

H.5.1 Multimedia Information Systems

H.5.2 User Interfaces

Keywords

Object-Relational Database Managemaent Systems, Multimedia Information System, XML, Content based Retrieval, Query Transformation, Request processing, Data Visualization, Mobile Computing

Danksagung

An dieser Stelle möchte ich all jenen danken, die mich während des Studiums und beim Verfassen dieser Diplomarbeit unterstützt haben.

An erster Stelle ist meine Mutter zu nennen, die mir das Studium überhaupt ermöglicht hat. Meiner Freundin Steffi danke ich für die Motivations- und Aufbauarbeit, die sie während meines Studiums und vor allen Dingen in der Diplomarbeitsphase geleistet hat.

Steffen Nowacki half mir mit seinen Erfahrungen bei der Java-Programmierung. Dr. Holger Meyer, Guntram Flach und Thomas Courvoisier haben diese Arbeit betreut. Guntram gilt mein ganz besonderer Dank, hat er doch viel dazu beigetragen, daß diese Arbeit so und rechtzeitig fertiggestellt wurde.

Schließlich danke ich Prof. Andreas Heuer und Prof. Bodo Urban für die Begutachtung dieser Diplomarbeit.

Inhaltsverzeichnis

1. Einleitung 8

1.1 Motivation 8

1.2 Vorgehensweise 8

2. Grundlagen & Begriffe 10

2.1 ORDBMS 10

2.2 XML 11

2.2.1 Historische Entwicklung 11

2.2.2 Sprachliche Konzepte 17

3. Weiterführende XML-Standards 24

3.1 Xlink, Xpointer 24

3.2 DOM 24

3.3 XSL/ XSLT 25

3.4 XQL und XML-QL 25

3.5 XML-Schema 26

3.6 RDF 26

3.7 Multimediapräsentation 27

3.8 WAP/WML 27

3.8.1 WAP 28

3.8.2 WML 29

4. DB-Systeme mit XML-Unterstützung 32

4.1 Objekt-Relationale DBMS und XML 32

4.1.1 IBM DB2 (UDB) 32

4.1.2 Informix 34

4.1.3 Oracle8i 38

4.2 XML-Server 38

4.2.1 Tamino 38

4.2.2 ObjectDesign eXcelon 39
4.2.3 POET Content Management Suite 39
4.2.4 Bluestone XML-Server 40
4.3 Fazit 40
5. System-Konzeption 42
5.1 Einordnung 42
5.1.1 Movi 42
5.1.2 iViP 43
5.2 Anwendungsszenario & Zielsetzung 44
5.3 Architektur & Realisierung 46
5.4 Prototypen 52
6. Zusammenfassung und Ausblick 54
Anhang A Metadatenstruktur 58
A.1 RDF-Graph für den Application Context 58
A.2 Zusätzliche Namespaces (Auszug) 59
Anhang A Content-Mediator 61
A.1 Application Context-DTD 61
A.2 SQL Anweisung-DTD 62
A.3 Client Request (Testimplementierung) 62
Abbildungsverzeichnis 8
Literaturverzeichnis 65

1. Einleitung

1.1 Motivation

Im Laufe der letzten fünf Jahre hat sich das World Wide Web (WWW) sehr stark weiter entwickelt. Besonders in den Bereichen „E-Commerce“ und „Recherche“ hat sich eine Menge getan. Werden heutzutage webbasierte Anwendungen entwickelt, so ist es wichtiger denn je geworden, nicht nur den herkömmlichen WWW-Browser zu unterstützen, sondern auch die Benutzer mobiler Endgeräte wie Palm-Handhelds und WAP-Handys zu berücksichtigen.

Auf der anderen Seite werden die zu verwaltenden Daten mehr und mehr in objekt-relationalen Datenbankmanagementsystemen (ORDBMS) gespeichert, da hier die Vorteile der objektorientierten Konzepte in bestehende relationale DBMSs integriert werden. Durch diese Integration ist es nun möglich, mit entsprechenden medienspezifischen Erweiterungen neue Datentypen in das DBMS einzubringen. Diese Erweiterungen –DataBlades (Informix) oder InterMedia (Oracle) – definieren gleichzeitig neben den neuen Datentypen eine neue Anwendungslogik sowie eigene Indexstrukturen.

Hauptschwerpunkt bei der Entwicklung von neuen Anwendungen liegt nun in der Nutzung dieser neuen Möglichkeiten bzgl. der Speicherung, Verwaltung und Recherche auf diesen hauptsächlich multimedialen Datentypen sowie auf der anwendungsspezifischen Darstellung und Repräsentation der multimedialen Dokumentstrukturen auf verschiedenen Endgeräten.

Im Rahmen dieser Diplomarbeit werde ich die Möglichkeit, XML-Dokumente (XML - eXtensible Markup Language) als Datenaustauschformat zwischen Datenbank und Anwendung zu nutzen, näher untersuchen. Außerdem stelle ich eine generische Schnittstelle vor, durch welche die Integration bestehender Daten in eine Anwendung unter Berücksichtigung des Endgerätes umgesetzt werden kann.

1.2 Vorgehensweise

Als erster Schritt werden im Kapitel 2 wesentliche Grundlagen und Begriffe eingeführt. Den Schwerpunkt bildet hierbei die Extensible Markup Language (XML) als Grammatik mit ihren Bestandteilen als Auszeichnungssprache bestehend aus Elementen und deren Attribu-

ten. Aufbauend auf der in Kapitel 2 eingeführten Begriffsterminologie werden in Kapitel 3 weiterführende Auszeichnungssprachen vorgestellt, die allesamt auf XML basieren und teilweise schon standardisiert sind oder kurz davor stehen. Das anschließende Kapitel 4 stellt die Ansätze verschiedener XML-Server und XML-Datenbanksysteme vor und beschreibt diese.

Im Kapitel 5 wird ein Überblick über das entwickelte, neuartige Konzept einer intelligenten und generischen Zugriffssteuerung auf multimediale, XML-gesteuerte Dokumentstrukturen in objekt-relationalen DBMS gegeben. Dabei werden die Anforderungen des MOVI- und iViP-Projektes im vollem Umfang berücksichtigt. Es wird nicht nur auf allgemeine Konzepte eingegangen, sondern auch Implementierungseinzelheiten erläutert. Die beschriebenen Anwendungsbeispiele sind in keiner Weise als vollständig, sondern lediglich als illustrativ für den jeweiligen Gegenstand anzusehen.

In Kapitel 6 wird eine Zusammenfassung durchgeführt. Ein Ausblick auf eine mögliche Fortsetzung dieser Arbeit schließt dieses Kapitel ab. An dieser Stelle wird darauf hingewiesen, daß von den Lesern dieser Arbeit Grundkentnisse der Datenbank-Technologie sowie des Client-Server-Ansatzes gefordert werden.

2. Grundlagen & Begriffe

In diesem Kapitel möchte ich einige grundlegende Begriffe klären, wobei ich als erstes die Objektrelationalen Datenbankmanagementsysteme (ORDBMS) kurz vorstellen möchte sowie dann eine Grobübersicht zum Thema XML geben werde. Weiterführende Informationen zu der viel und oft besprochenen Thematik Multimedia-Datenbanksysteme sind [Ditt00, AbKl97, ABH97] zu entnehmen.

2.1 ORDBMS

Die objektrelationalen Datenbankmanagementsysteme [StBr99, Ston96, FlLo01] bauen auf die Technologie relationaler Konzepte auf und haben die Fähigkeit, komplexe Objekte durch objektorientierte Konzepte zu verwalten. Die Objektorientierung steckt dabei in den benutzerdefinierten Datentypen, weil hier die Methoden und die Vererbung wirksam werden. So läßt sich ein ORDBMS folgendermaßen charakterisieren:

Benutzerdefinierter Typen (UDT für „**U**ser **D**efined **T**ype“) können aus Standarddatentypen und bereits vorhandenen UDT unter Benutzung von Typkonstruktoren (`SET OF`, `TUPLE OF`, `LIST OF`) gebildet werden. An solche UDT können Methoden (UDR für „**U**ser **D**efined **R**outines“) gebunden werden, die nur auf diesem Typ ausführbar sind. Der Benutzer sollte nicht von außerhalb auf die interne Struktur der komplexen Typen zugreifen können. (*Einkapselung*).

Bei der Tabellendeklaration können benutzerdefinierte- und Standarddatentypen gleichermaßen verwendet werden und es werden folgende Konzepte unterstützt.

(1) *Vererbung*: UDT können Strukturinformationen und Methoden von andern benutzerdefinierten Datentypen erben und somit eine Typhierarchie aufbauen. Eine Relation (*Klasse*) kann als Unterklasse einer anderen Relation (in diesem Fall Oberklasse) definiert werden; die Objekte der Unterklasse sind dann eine Teilmenge der Oberklasse. Dadurch wird eine Klassenhierarchie aufgebaut.

(2) *Objektidentität* (OID) sichert die Trennung von Schema und Werten.

(3) Die *Anfragesprache* sollte einerseits den Standard von SQL erfüllen und andererseits die Möglichkeiten der neu hinzugekommenen komplexen Objekte ausnutzen.

(4) Navigation über Referenzen, Zugriff auf geschachtelte Objekte / Relationen durch „Pfadausdrücke“.

(5) *Overriding* sollte es ermöglichen, ererbte Methoden zu redefinieren oder zu überschreiben, so daß diese Methoden dem neuen Datentyp angepaßt bzw. verfeinert werden können.

Abschließend ist noch zu erwähnen, daß man wie bei den Programmiersprachen auch bei den objekt-relationalen Datenbanksystemen dazu übergeht, vordefinierte Typ-Bibliotheken für die verschiedensten Anwendungsbereiche anzubieten, die dann einfach in das System integriert werden können. Informix bietet zum Beispiel sogenannte *DataBlades* an, die das Datenbanksystem um z.B. Multimediadatentypen für Text-, Bild-, Videodaten u.a. erweitern und zahlreiche spezielle Funktionen auf diesen Datentypen bereitstellen. Diese Erweiterungen gibt es nicht nur vom Hersteller sondern auch von Drittanbietern.

2.2 XML

In diesem Abschnitt möchte ich eine Grobübersicht über XML [Wor98a, Brad00, BeMi98, HaMe01] geben und erläutern, warum sich XML so hervorragend als Austauschformat für strukturierte Daten eignet.

2.2.1 Historische Entwicklung

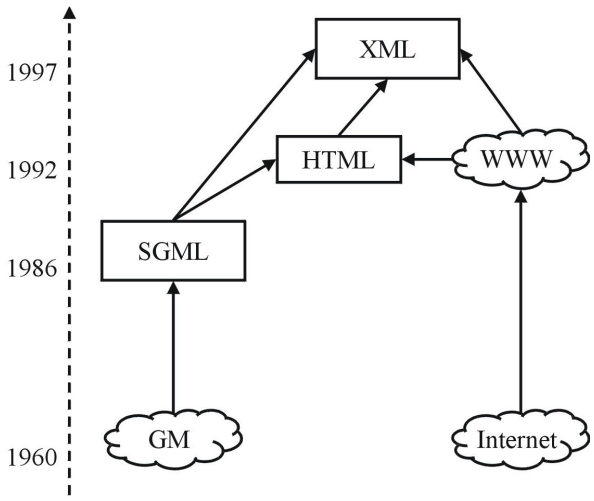

Abbildung 2.1 Die geschichtliche Entwicklung von XML

Anfang der 60er Jahre entstand die Idee, die Struktur eines Dokumentes zu kennzeichnen. Dieses Konzept konnte sich allerdings erst mit der Einführung von SGML (**S**tandard **G**eneralized **M**arkup **L**anguage) durchsetzen, deren Standardisierung 1986 durch die ISO bestätigt

wurde. SGML wurde vorrangig eingesetzt, um „sehr große“ Datensammlungen wie Dokumentationen im Flugzeugbau, Enzyklopädien oder mehrbändige Gesetzesbücher anzulegen und zu verwalten. Die Komplexität und der Umfang von SGML erwies sich jedoch als Hemmschwelle bei deren Verbreitung und Anwendung. Die Entwicklung von SGML-nutzenden Programmen war ein sehr großer Arbeitsaufwand, so daß entsprechende Anwendungen recht spärlich auf den Markt kamen, fehlerbehaftet waren und im Vergleich mit proprietären Lösungen zudem noch sehr zeit- und rechenaufwendig arbeiteten.

Mit dem Wachstum des Internet und zunehmender Bedeutung des WWW als Medium für den Austausch und die Suche von Informationen wurde 1992 der HTML-Standard verabschiedet (**H**yper**t**ext **M**arkup **L**anguage). Dadurch war es nun durch spezielle Auszeichnungselemente („Markup Tags“; im weiteren als Tags bezeichnet) möglich, den Dokumenten eine Struktur zu geben, den Text für die Ausgabe zu formatieren und diese Dokumente über Verweise mit anderen Dokumenten oder Multimediadaten wie Bildern, Videos und Ton miteinander zu verknüpfen. Als Auszeichnungssprache enthält HTML daher Befehle zum Auszeichnen spezieller Bereiche eines Dokuments, wie Überschriften, Textabsätze, Listen, Tabellen oder Grafikreferenzen. Dabei geht das HTML-Beschreibungsschema von einer hierarchischen Gliederung aus und beschreibt Dokumente, welche wiederum globale Eigenschaften wie beispielsweise einen Titel oder eine Hintergrundfarbe haben, wobei der eigentliche Inhalt aus Elementen besteht, zum Beispiel einer Überschrift 1. Ordnung, einem Textabsatz und einem Bild. Einige dieser Elemente haben wiederum Unterelemente. So könnte ein Textabsatz eine als fett markierte Textstelle enthalten und eine Aufzählungsliste aus einzelnen Listenpunkten bestehen.

```
<HTML>
      <HEAD>
            <TITLE>HTML Beispiel</TITLE>
      </HEAD>
      <BODY>
            <P>Ein Absatz</P>
            <P>Ein zweiter Absatz mit <B>fett geschriebenen</B> Text
            und einem <A HREF=“anderesDok.html“>Verweis</A> auf
            ein anderes Dokument.</P>
      </BODY>
</HTML>
```

Beispiel 2.1 HTML Beispiel 1

HTML ist eine einfach zu benutzende Sprache, um Daten im WWW zu veröffentlichen und darzustellen. Diese „Einfachheit“ ist auch der Grund, warum sich diese Beschreibungsspra-

che so schnell als Grundlage des WWW durchgesetzt hat. Trotz der Möglichkeit, Daten aus unterschiedlichsten im WWW verteilten Datenquellen darzustellen, resultieren gerade aus dieser „Einfachheit“ auch schwerwiegende Nachteile, die eine Erschließung und Verwendung durch computergesteuerte Anwendungen erschweren:

- **HTML ist *Präsentationsorientiert***: Wie man an Beispiel 2.1 schon sehen kann, ist HTML darauf ausgerichtet, wie die Daten eines Dokuments **präsentiert** werden, die Bedeutung der Daten geht jedoch dabei verloren. Es ist z.B. nicht feststellbar, ob die beiden Absätze in irgendeinem Zusammenhang miteinander stehen. Beispiel 2.2 produziert in einem gängigen Browser (Microsoft Internet Explorer 5.5) dieselbe Darstellung wie Beispiel 2.1, obwohl die Daten völlig anders strukturiert sind. Durch die enge Verknüpfung des Inhalts mit den Darstellungsinformationen gibt es mit HTML nur sehr ineffiziente Möglichkeiten, dieselben Daten auf unterschiedlichen Medien oder für unterschiedliche Personengruppen zugänglich zu machen.

```
<HTML>
      <TITLE>HTML Beispiel</TITLE>
      <BODY>
            Ein Absatz<BR><BR>
            Ein zweiter Absatz mit <B>fett geschriebenen</B> Text
            und einem <A HREF=anderesDok.html>Verweis</A> auf
            ein anderes Dokument.
```

Beispiel 2.2 HTML Beispiel 2

- **HTML ist *nicht erweiterbar.*** Das größte Problem bei HTML ist die fehlende Erweiterbarkeit der verfügbaren Tags. HTML hat eine kleine feste Menge an Auszeichnungselementen. Eine anwendungsspezifische Erweiterung, um beispielsweise den Absatz in einem bestimmten Abschnitt eines Buches – z.B. ein „Vorwort“ – zu kennzeichnen, ist in HTML nicht vorgesehen. (siehe Beispiel 2.3)

```
<HTML>
      <HEAD>
            <TITLE>Buch Beispiel</TITLE>
      </HEAD>
      <BODY>
            <H1>Vorwort</H1>
            Hier koennte der Text fuer ein Vorwort stehen.
            <H1>Kapitel 1</H1>
            Der Text des 1. Kapitel kommt hier rein.
            <H2>Abschnitt 1</H2>
            <P>Das waere der 1. Abschnitt im 1. Kapitel.
            <H1>Kapitel 2</h1>
            Dies ist ein Kapitel ohne Unterabschnitte.<BR><BR>
      </BODY>
</HTML>
```

Beispiel 2.3 Buch-Beispiel in HTML

- **HTML fehlt die Grundlage für die *Validierung des Inhalts:*** Es gibt in HTML keine Überprüfungsmöglichkeiten, ob die Daten, die eingegeben oder importiert wurden, gültig sind, d.h. ob sie einem speziellen korrekten Format entsprechen. Als einzige Möglichkeit läßt sich das HTML-Dokument in seine HTML-spezifischen Bestandteile wie Kopf, Titel und Dokumentkörper zerlegen. Auch hier bezieht sich die Zerlegung wieder nur auf die präsentativen Merkmale. Ein Programm, welches Beispiel 2.1 auswerten kann, würde mit Beispiel 2.2 schon einige Probleme bekommen, da in diesem Beispiel die schließenden Tags **`</BODY>`** und **`</HTML>`** bewußt weggelassen wurden. Außerdem wird statt des Paragraph-Tags (**`<P>`**) ein doppelter Zeilenumbruch (**`<BR>`**) verwendet. Trotz dieser Mängel erfolgt die Darstellung im Browser korrekt und mit demselben Ergebnis wie in Beispiel 2.1. Diese hohe Fehlertoleranz der Browser ist einerseits die Stärke von HTML, weil dadurch jeder Laie HTML anwenden und Webseiten auch mit eher mangelhaften HTML-Kenntnissen erstellen und ins WWW einbringen kann. Andererseits ist das aber auch eine der Schwächen von HTML, da computergestützte Anwendungen, die diese Daten auswerten und analysieren sollen, sehr umständlich zu implementieren und schwierig zu warten sind.

Während in HTML die sprachlichen Konzepte von SGML übernommen wurden, hielt man sich aber nicht an die zugrundeliegenden Richtlinien. Erst durch spätere HTML-Standards wurde dies behoben, so daß man heute HTML als Anwendung von SGML betrachten kann. Allerdings halten sich die führenden Browser-Hersteller wie Microsoft und Netscape nicht an diese Standards, sondern benutzen inkompatible eigene Tags.

Der Druck, SGML zu vereinfachen, wuchs mit dem Bestreben, HTML erweiterbar zu gestalten und eine klare Trennung von Inhalt und Darstellung zu erreichen. So wurde 1996 durch das „World Wide Web-Konsortium“ (W^3C), welches unter anderem auch für den HTML-Standard verantwortlich ist, beschlossen, einen neuen Standard zu entwickeln – die E***X***tensible **M**arkup **L**anguage (XML). Im ersten Schritt wurden 10 Designziele der neuen Sprache entworfen, um sicher zu stellen, daß die endgültige Spezifikation letztendlich auch den Anforderungen gerecht wird, für die sie eigentlich entworfen wurde:

1. XML soll direkt über das Internet nutzbar sein.
2. XML soll durch viele Anwendungen nutzbar sein.
3. XML soll kompatibel mit SGML sein.
4. Es soll einfach sein, Anwendungen zu entwickeln, die XML-Dokumente verarbeiten können.
5. Die Anzahl der optionalen Features von XML soll minimal sein, idealerweise Null.
6. XML-Dokumente sollen für den Menschen lesbar und verständlich sein.
7. Der Entwurfsprozeß von XML soll schnell vorbereitet werden.[1]
8. Der Entwurf soll auf formalen Grundlagen basieren und präzise sein.
9. Es soll einfach sein, XML-Dokumente zu erzeugen.
10. Knappheit ist für die Auszeichnung von XML-Elementen von minimaler Bedeutung.

XML setzt dabei auf den Grundlagen und den Vereinbarungen von SGML und HTML auf und basiert grundlegend auf SGML, übernimmt aber einige Besonderheiten von HTML und enthält neue zusätzliche Möglichkeiten, um so auch dem Gebrauch im Internet gerecht zu werden. Die XML-Spezifikation in der Version 1.0 wurde im Februar 1998 veröffentlicht.

[1] Wenn man bedenkt, daß der Entwicklungsprozeß von XML 1996 begann und 1998 die endgültige Spezifikation verabschiedet wurde, ist es eher subjektiv, ob dieses Ziel erreicht wurde oder nicht.

```
<?xml version="1.0"?>
<Book Title="Das Buch-Beispiel mittels XML">
      <Chapter Title="Vorwort">
            <Paragraph>
                  Hier koennte der Text fuer ein Vorwort stehen.
            </Paragraph>
      </Chapter>
      <Chapter Title="Kapitel 1">
            <Paragraph>
                  Der Text des 1. Kapitel kommt hier rein.
            </Paragraph>
            <Section title="Abschnitt 1">
                  <Paragraph>
                        Das waere der 1. Abschnitt im 1. Kapitel.
                  </Paragraph>
            </Section>
      </Chapter>
      <Chapter Title="Kapitel 2">
            <Paragraph>
                  Dies ist ein Kapitel ohne Unterabschnitte.
            </Paragraph>
      </Chapter>
</Book>
```

Beispiel 2.4 Buch-Beispiel in XML

Im Gegensatz zu HTML erfolgt bei XML eine klare Trennung von Inhalt und Präsentation. Wie sehr gut an Beispiel 2.4 zu sehen ist, dient XML selbst „nur“ der Beschreibung des Inhalts und von „Daten über diese Daten“ (Metadaten). XML ist somit eine Metasprache, die nicht die benötigte Art und Weise der Darstellung eines Dokumentes kennzeichnet, sondern die Struktur eines jeden Textbausteins beschreibt. Während bei HTML letztendlich Informationen über die Formatierung der Daten speichert (z.B. „**fett“** geschriebener Text, Beginn und Ende eines Absatzes, Attribute über die Formatierung eines Bildes), werden bei XML semantische Informationen über die zu beschreibenden Daten wie beispielsweise „Vorwort“, „Kapitel“, „Abschnitt“ oder „Zitat“ abgelegt. Die Aufgabe der Inhaltsdarstellung wird mittels anderer Werkzeuge gelöst – **C**ascading **S**tyle **S**heets (CSS), E***X***tensible **S**tylesheet **L**anguage (XSL) oder XHTML.

Das zweite Problem von HTML, die fehlende Erweiterbarkeit, ist bei XML ebenfalls gelöst. Dem Anwender steht es dank XML frei, eigene Tags einzuführen, um so die Struktur seiner Dokumente festzulegen. Die in einem Dokument zu verwendenden Tags können auch mit anderen Anwendern geteilt werden. Verschiedenste Anwender bzw. Firmen können so über

das WWW miteinander verbunden, dieselben Tag-Definitionen für ihre Datenbestände verwenden, um eine globale Zusammenarbeit zu erreichen.

XML-Dokumente können über einfache Mechanismen auf Gültigkeit schlechthin oder auf Gültigkeit bezüglich einer speziellen **D**okument **T**yp **D**eklaration (DTD) überprüft werden. Dadurch wird auch das letzte signifikante Problem von HTML (die fehlende Validierungsmöglichkeit) gelöst.

2.2.2 Sprachliche Konzepte

„XML ist dafür bestimmt, den Inhalt von elektronischen Dokumenten in einer Art und Weise zu speichern, die sowohl für Mensch als auch für den Computer lesbar ist.“ ([Brad00] S.15)

Im vorherigen Abschnitt habe ich die Notwendigkeit aufgezeigt, die zu der Entwicklung von XML geführt hat. In diesem Abschnitt möchte ich nun die sprachlichen Konzepte vorstellen, die hinter XML stehen.

XML-Dokumente

XML-Dokumente enthalten grundsätzlich Textdaten, die mit jedem Texteditor bearbeitet werden können. Das Dokument kann dabei im Dateisystem abgelegt sein, dynamisch durch ein CGI-Programm als Ergebnis einer Browser-Anfrage generiert werden oder ein Wert in einem Datenbankfeld sein. Es spielt keine Rolle, wo dieses Dokument liegt bzw. wie es erzeugt wird.

Ein XML-Dokument besteht sowohl aus einer logischen- als auch aus einer physischen Struktur. Die logische Struktur unterteilt dabei ein Dokument in verschiedene logische Einheiten (**Elemente**), die eine bestimmte Bedeutung haben. Die physische Struktur ermöglicht die getrennte Benennung und Speicherung von Komponenten des Dokuments (**Entities**). Dadurch wird einerseits die Einbindung von separat gespeicherten Daten (z.B. Bilddaten) sowie die Wiederverwendung von Dokumentkomponenten (z.B. ***D**okument **T**ype **D**efinition* – DTD (siehe dort)) ermöglicht.

XML-Elemente

Sie bilden quasi den „Befehlssatz“ eines XML-Dokumentes und dienen der Strukturierung der Daten. Dabei wird ein Element durch das *öffnende Tag* und das *schließende Tag* begrenzt. Alle Daten, die zwischen diesen beiden Tags liegen, werden als Elementinhalt bezeichnet, der seinerseits wieder aus weiteren Elementen oder Zeichendaten oder aus beidem

(**`mixed content`**) bestehen kann. Der Elementinhalt des ersten Paragraph-Elements in Beispiel 2.4 ist die Zeichenkette „Hier koennte der Text fuer ein Vorwort stehen.“.

XML-Tags haben das fast gleiche Format wie HTML-Tags. Öffnende Tags beginnen mit einem **`<`** und schließende Tags mit **`</`**. Dann folgt der Elementname und ein abschließendes **`>`**. Der Elementname ist dabei im Gegensatz zu HTML frei wählbar und sollte den Inhalt des Elements beschreiben (**`<Chapter>,<Section>`**) und nicht die Darstellung (**`<H1>,<H2>`**).

Einen Spezialfall bilden die ***leeren Elemente***, die keinen Inhalt beschreiben. Sie sind vergleichbar mit dem <**Img**>- oder dem <**br**>-Tag in HTML. Werden diese Elemente durch ein öffnendes und schließendes Tag gebildet, dürfen zwischen beiden Tags **keine** Daten stehen. Im Normalfall werden öffnendes und schließendes Tag miteinander verschmolzen, so daß das Element durch **`<Elementname/>`** gebildet wird.

Elemente können drei Inhaltsarten haben:

- Inhalt besteht nur aus Elementen. („Element- Inhalte“)
- Inhalt besteht nur aus Daten. (CDATA, PCDATA)
- Inhalt besteht aus Daten und Elementen. (Mischform)

XML-Bäume

XML-Dokumente stellen genaugenommen Baumstrukturen dar. Ein Baum wird dabei durch die Schachtelung der XML-Elemente gebildet. Jedes Element muß ein Elternelement haben und darf sich nicht mit anderen Elementen überlappen. Wie anhand der beiden folgenden Grafiken gut zu erkennen ist, muß jedes Element innerhalb eines anderen Elementes liegen und darf nicht andere Elemente überschneiden. Ausgehend von dem Begriff „Stammbaum“ spricht man bei den übergeordneten Elementen auch von „Elternelementen“, wobei die Elemente, welche auf derselben Hierarchieebene liegen, als „Geschwister“ und deren nachfolgenden Elemente als „Kinder“ bezeichnet werden. Die dadurch gebildete *logische Struktur* läßt sich am ehesten durch eine Baumstruktur oder ein Blockschema darstellen.

Ausgehend von dieser Struktur muß jedes Element genau ein Wurzelelement haben, welches auch Dokumentelement genannt wird und als einziges Element kein Elternelement haben darf.

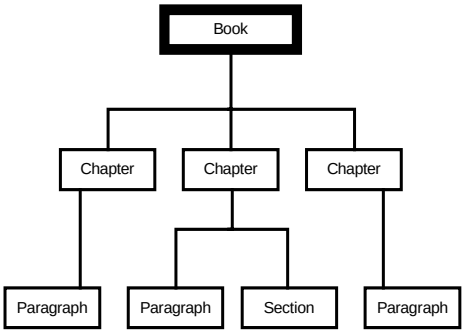

Abbildung 2.2 Baumstruktur

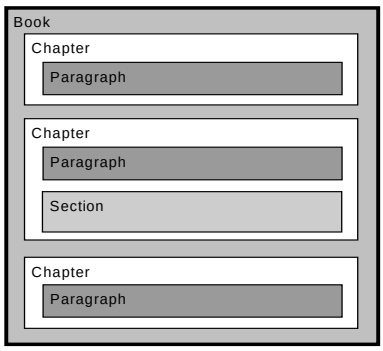

Abbildung 2.3 Blockschema

Attribute

Als Attribute werden die ein Element beschreibenden zusätzlichen Informationen bezeichnet. Sie werden in das Start-Tag des zugehörigen Elements eingebettet, bestehend aus dem Attributnamen und dem dazugehörigen Attributwert.

Durch das Präfix „xml:“ im Attributnamen werden durch den XML-Standard reservierte Attribute gekennzeichnet. Derzeit sind es zwei Attribute: „xml:lang“ für länderspezifische Werte und „xml:space“ für die Behandlung der Darstellung der nichtdruckbaren Zeichen, die durch diesen Standard definiert sind.

Deklarationen

Durch Deklarationen wird die Verarbeitung der XML-Dokumente im XML-Prozessor gesteuert. Dabei kann mittels Deklarationen der Typ eines Dokumentes festgelegt und Elemente, Attribute und Entities definiert werden.

Gekennzeichnet werden sie durch „<!“ und „>“. Deklarationen können unter Verwendung von „[“ und „]“ auch geschachtelt angegeben werden.

Processing Instructions

Durch die Verwendung von **P**rocessing **I**nstructions (PI) können Anweisungen an spezielle Programme, die dieses Dokument verarbeiten, in das XML-Dokument eingebunden werden. Mittels „<?**PITarget** *instruction*?>“ werden sie in das Dokument eingebunden. Gelangt das entsprechende Programm während Dokument-Verarbeitung auf solch eine PI und „kennt“

das *PITarget*, so kann es entsprechend der in *instruction* stehenden Anweisungen reagieren. Kennt es das *PITarget* nicht, so wird der Inhalt von *instruction* reagiert.

```
<?sql query="select title from booktable"?>
<?xml version="1.0"?>
```

Beispiel 2.5 Processing Instructions

Eine Anwendung der PI ist die *XML Deklaration*. Sie hat zu Beginn eines XML-Dokumentes zu stehen und ist durch das *PITarget* "**`xml`**" gekennzeichnet, welches durch den XML-Standard reserviert ist. Durch den Parameter "version" wird dabei die Versionsnummer des für dieses Dokument verwendeten XML-Standards angegeben. Über zusätzliche Parameter wie "encoding" und "standalone" wird angegeben, welcher Zeichensatz benutzt wird und ob die Definition des Dokumenttyps in einer externen Datei erfolgt.

Physische Struktur

Wie anfangs erwähnt, erlaubt die physische Struktur die separate Benennung und Speicherung von Komponenten des XML-Dokuments, wodurch es dann möglich ist, dieselben Komponenten in unterschiedlichen Dokument-Instanzen wieder zu verwenden.

Entities

Jede physische Einheit wird als ***Entity*** bezeichnet. Entities werden vor allem dann eingesetzt, wenn

- dieselbe Information in verschiedenen Dokumenten bzw. an verschiedenen Stellen des Dokuments genutzt werden soll,
- große Dokumente verwaltet werden müssen,
- die Informationen nicht im ASCII-Format vorliegen (z.B. Multimediadaten)

oder

- wenn die Informationen über verschiedene inkompatibel Systeme verteilt sind.

Entities müssen in der Dokumenttypdefiniton stehen und werden durch die Deklaration **`<!Entity`** *`Entity-Name "Entity-Wert"`***`>`** definiert. Entities werden durch den XML Prozessor auf unterschiedliche Art und Weise je nach dem Typ des Entities ausgewertet.

```
<!ENTITY DATITEL "Zugriff auf multimediale XML-Dokumente in
Objekt-relationalen Datenbanksystemen">
...
Diese Diplomarbeit hat den Titel &DATITEL; ...
```

Beispiel 2.6 Interne Text-Entities und deren Anwendung

So können *Interne Text-Entities* verwendet werden, um längere im Dokument häufig benutzte Textpassagen abzukürzen. Dabei muß der Ersetzungstext in Anführungszeichen ("" oder ‘‘) eingefaßt sein und wird über **`&Entity-Name;`** referenziert, wobei dann automatisch der XML-Prozessor die Referenz durch den Ersetzungstext ersetzt.

Jeder XML Prozessor verfügt über die *vordefinierten Entities* **`<`** für **`<`**, **`>`** für **`>`**, **`&`** für **`&`**, **`&apos;`** für ‘ und **`&quot;`** für **`"`** , die durch den XML-Standard spezifiziert sind.

```
<!ENTITY apos "&#39">
<!ENTITY quot "&#34²>
```

Beispiel 2.7 Zeichen-Entities

Zeichen-Entities werden durch die Angabe von „&#“ gefolgt vom Zeichencode definiert. Dabei stehen Zahlenwerte im Bereich von 0-256 für ISO-8859-1 und im Bereich von 256-65535 für Unicode/ISO-10646.

Mittels *Externer Entities* können vorhandene außerhalb des Dokuments gespeicherte Daten in das Dokument eingebracht werden. Jedes Entity welches nicht intern ist, gilt automatisch als externes Entity. Dabei wird durch das Schlüsselwort **`SYSTEM`** ein sogenannter *System-Identifier* bestimmt, der den Speicherort der jeweiligen Information angibt und einen relativen oder absoluten **U**niform **R**esource **I**dentifier (URI) darstellt. Über das Schlüsselwort **`PUBLIC`** kann auch ein sogenannter *Public-Identifier* verwendet werden. Dabei wird durch einen Entity-Manager der Public-Identifier über einen Katalog auf einen System-Identifier abgebildet. Bezieht man sich auf Entities, die unternehmensweit genutzt werden, deren Speicherort sich aber häufig ändert, so müßten bei jeder Änderung in sämtlichen XML-Dokumenten, die dieses Entity benutzen, die System-Identifier angepaßt werden. Dieses umständliche Verfahren läßt sich mittels des Public-Identifiers auf ein Minimum reduzieren. Für den Fall, daß der XML-Prozessor nicht in der Lage ist den Public-Identifier aufzulösen, muß zusätzlich noch der System-Identifier angegeben werden.

```
<!ENTITY chapter1 SYSTEM "/book/chapters/chap1.xml">
<!ENTITY chapter1 PUBLIC ²-//myCorp//Books//XML//Chap1//En"
      "/books/chapters/chapter1.xml">
```

Beispiel 2.8 Externe Entities

Werden Entities eingebunden, die nicht im Textformat vorliegen, so muß das Datenformat erst einmal über eine *Notations-Deklaration* definiert werden. Dies erfolgt über die Deklaration **<!NOTATION** *Notation-Name External-Notation-Identifier***>**[2]**.**

```
<!NOTATION JPEG PUBLIC "-//myCorp//NOTATION JPEG//EN"
      "/usr/bin/xview“>
```

Beispiel 2.9 Notationen

Das zusätzliche Schlüsselwort NDATA bestimmt bei der Deklaration des externen Entities das verwendete Datenformat indem es auf den zugehörigen Notation-Namen verweist.

```
<!ENTITY picture-transform SYSTEM "/imgs/transformation.jpg" NDATA JPEG>
```

Beispiel 2.10 Binäre Entities

Die DTD

Die **D**okument **T**ype **D**efinition (DTD) ist sozusagen das Kernstück eines XML-Dokumentes. In ihr werden die strukturellen Gemeinsamkeiten von gleichartigen Dokumenten beschrieben und Regeln für die Definition der Elemente, Attribute, Entities und anderer XML-Daten wie Processing Instructions und Notationen deklariert. Sie beschreibt somit die logische Struktur einer ganzen Klasse von Dokumenten. Einer der wesentlichen Unterschiede zwischen SGML und XML ist, daß es in XML nicht zwingend erforderlich ist, eine DTD anzugeben.

Validierung

Bei XML-Dokumenten unterscheidet man zwischen **wohlgeformten** und **gültigen** Dokumenten. Ein XML-Dokument ist *wohlgeformt*, wenn es den XML-Syntaxregeln entspricht und alle Elemente ordnungsgemäß verschachtelt sind. Ein gültiges XML-Dokument entspricht der Struktur, die durch seine DTD vorgegeben wird. Die DTD eines Dokuments kann sowohl extern als auch extern oder als Kombination von beidem angegeben werden, wobei bei letzterem die Deklarationen der internen DTD Vorrang haben und die externen Deklarationen überschreiben.

[2] Wie bei denen externen Entities kann auch hier der „External-Notation-Identifier“ einen System-/Public-Identifier darstellen.

Über die Definition der Elemente und Element-Inhalte wird in der DTD angegeben, in welchem Kontext die Elemente vorkommen dürfen. Dabei stehen Elementsequenzen, Alternativen sowie Quantitätsangaben zur Verfügung, über die bestimmt wird, welche und wieviele Elemente ein anderes Element enthalten darf.

Die genaue Spezifikation des XML-Standards Version 1.0 ist [Wor98a] zu entnehmen, wobei eine sehr gute deutsche Übersetzung in [BeMi98] zu finden ist.

3. Weiterführende XML-Standards

Die XML-Technologie wird die Basis für zukünftig dominierende Darstellungsformen von Dokumenten und Daten 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, Multimediapräsentationen oder Datenbankanfragen standardisiert. Für die Verarbeitung von XML-Dokumenten steht ein Objektmodell zu deren Repräsentation bereit und eine Transformationssprache zu deren Übersetzung. In diesem Kapitel möchte ich nun einen Überblick über die für die Zielstellung dieser Arbeit relevanten Techniken und die Zusammenhänge erläutern.

3.1 Xlink, Xpointer

Die Möglichkeit, Dokumente mit Hilfe von Verweisen (Links) untereinander zu vernetzen, ist bereits von HTML bekannt. Diese Möglichkeit bietet auch XML. Es liegen dem W3C zwei Entwürfe vor, nämlich **XML Linking Language** (Xlink) und **XML Pointer Language** (XPointer). Xlink definiert zwei Typen von Verweisen, den einfachen und den erweiterten Verweis. Der einfache Link entspricht dem von HTML. Mit einem erweiterten Link kann ein Verweis zwischen zwei Ressourcen errichtet werden, ohne daß eine von ihnen irgend etwas über diese Relation weiß. In diesem Fall existiert der Link in einer dritten Ressource. Man kann außerdem einen Link zwischen mehr als nur zwei Dokumenten definieren. Einrichten von Links von Medien oder Dateien aus, die selbst nicht beschreibbar sind (z.B. CD-Roms) sind ebenso möglich.

Mit XPointer kann auf eine Stelle oder einen Bereich innerhalb eines XML-Dokuments verwiesen werden. Absolute Adressierung ist mit Hilfe von vordefinierten Schlüsselwörtern möglich. Ist mit Hilfe eines der Schlüsselwörter der Ausgangspunkt eines Verweises festgestellt, kann man mit relativen Angaben auf beliebige Stellen eines Dokuments verweisen.

3.2 DOM

XML-Dokumente liegen bekanntlich in Textform vor. Eine Anwendung, die ein XML-Dokument bearbeiten will, muß zuerst dieses analysieren und den Inhalt des Dokuments in

eine Baumstruktur umwandeln. Um eine einheitliche Sicht auf XML-Dokumente zu haben, hat das W3C das **D**ocument **O**bject **M**odel (DOM) spezifiziert. DOM ist eine plattform- und sprachunabhängige Schnittstelle, die Programmen einen dynamischen Zugriff auf Inhalt und Struktur der Dokumente ermöglicht.

3.3 XSL/ XSLT

XML ist eine Strukturbeschreibungssprache und kann unter anderen zum Datenaustausch in heterogenen Umgebungen verwendet werden. Im Gegensatz zu HTML beinhaltet ein XML-Dokument keine Informationen über die Darstellung der beschriebenen Daten.

XML-Architekturen nutzen dazu im allgemeinen XSL als Transformationssprache. XSL stellt dabei einen allgemeinen Mechanismus für die Transformation von XML-Bäumen bereit und wird momentan als Standard vom W3C-Consortium entwickelt. XSL hat zwei Bestandteile. Eine allgemeine Sprache zur Transformation von Bäumen (XSLT) und eine Sprache zur Notation von in Flächen dargestellten Bäumen, die aus Formated 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 Transformations-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 unterschiedliche Medien und Endgeräte.

3.4 XQL und XML-QL

XML soll die für den Nutzer sichtbare Datenrepräsentation darstellen. Dementsprechend wird der Nutzer Datenbankanfragen in einer auf XML-basierenden Anfragesprache stellen wollen, die eine Rolle ähnlich SQL für relationale DBMS haben würde. Momentan ist durch das W3C ein Standard für eine solche Sprache in Erarbeitung. Es gibt derzeit zwei konkurrierende Entwürfe: XQL und XML-QL.

XML-QL (Query Language for XML) [Wor98b, DFF+98] definiert eine Syntax und Semantik für die Auswahl von Daten aus XML-Dokumenten und die Erzeugung neuer XML-

Dokumente daraus. Bei dieser Sprache wird ein Mechanismus eingesetzt, der dem „Query-by-Example“ ähnlich ist. Das Grundmuster einer Anfrage sind WHERE-CONSTRUCT-Klauseln, wobei die WHERE-Klausel spezifiziert, wonach gesucht wird, und die CONSTRUCT-Klausel angibt, wie das Ergebnis gebildet wird. Es ist dabei möglich Variablen einzusetzen,. Über die dann Anfragen an mehrere Dokumente gestellt werden können.

XQL [Wor98c, RLS98] dagegen hat einen etwas anderen Ausgangspunkt. XQL nimmt die Beschreibung der in XSL verwendeten Muster-Sprache als Ausgangspunkt und erweitert sie zu einer Abfragesprache. Die Verarbeitung der Abfrage entspricht dann dem Auffinden eines auf das Muster passenden Teildokuments. Das Grundmuster von XQL-Anfragen besteht aus dem Selektieren zu verarbeitender Elemente mittels Pfadausdrücken. Innerhalb eines Iterators kann dann auf die entsprechenden Werte zugegriffen werden. Diese können dann umgeformt und eingebettet in andere, zusätzliche Elemente als Ergebnis ausgegeben werden. Mit XQL hat man eine Möglichkeit, mit einer recht kompakten Syntax Anfragen an einzelne Dokumente zu stellen. Aufgrund dessen wird XQL gegenwärtig in den meisten Systemen als Anfragesprache eingesetzt. Anfragen, deren Ergebnisse aus mehreren Dokumenten zusammengesetzt werden müssen, lassen sich jedoch nicht realisieren.

Es ist offen, welcher der beiden Entwürfe sich durchsetzen wird. Dabei ist zu beachten, daß XQL sehr starke Unterstützung durch einige Softwarehersteller hat.

3.5 XML-Schema

Zugleich werden auch Standardisierungsbemühungen unternommen, XML durch entsprechende Schemadefinitionen zu einer vollwertigen Datenrepräsentationssprache zu erweitern, beispielsweise mit ersten Dokumenten zu XML-Schema. [Wor99a, Worl97]

3.6 RDF

Mit dem **R**esource **D**escription **F**ramework (RDF) versucht das W3C [Wor99e], ein Rahmenwerk zur Bearbeitung von Metadaten zur Verfügung zu stellen. RDF unterstützt die Interoperabilität zwischen Anwendungen, die maschinenverständliche Informationen über das Internet austauschen. Das um Aussagen höherer Ordnung ergänzte Datenmodell von RDF ähnelt stark einem Entity-Relationship-Modell. Was ihm lediglich fehlt sind Möglichkeiten zur Festlegung und Typisierung der Eigenschaften von Ressourcen.

Für die Festlegung eines solchen Schemas erarbeitete das W3C einen entsprechenden Standard mit der RDF-Spezifikation. Mit diesen Hilfsmitteln können nun standardisierte Schemata für Metadaten festgelegt und auf der Basis von RDF spezifiziert und verarbeitet werden. Während RDF lediglich einen Rahmen für die Notation von Metadaten bildet, legen diese Schemata die tatsächlich beim Nutzer sichtbaren Informationen fest.

3.7 Multimediapräsentation

Die **S**ynchronized **M**ultimedia **I**ntegration **L**anguage (SMIL) soll die Integration einer Menge von „unabhängigen“ Multimediaobjekten in eine synchronisierte Multimediapräsentation erlauben. Die in SMIL definierten Elemente erlauben die Markierung des Ablaufs einer Präsentation. Die wichtigsten Elemente legen paralleles und sequentielles Abspielen von Strömen zusammen mit in Attributen festgelegten Zeitbedingungen fest.

Auf XML-Basis erarbeitet das W3C die Spezifikation **S**calable **V**ector **G**raphics (SVG). SVG definiert eine Reihe von Elementen zur Auszeichnung von Koordinatensystemen, für einen Satz grundlegender Zeichnungselemente und deren Attribute, für die Einbindung von Pixelgrafiken in Zeichnungen und für die Verwendung von Text. SVG ist derzeit noch nicht standardisiert und es existieren nur erste experimentelle Darstellungsprogramme. Es ist aber abzusehen, daß ein solches Zeichnungsformat erfolgreich sein muß, da der Bedarf nach höherwertigen Darstellungen z.B. für technische Dokumentationen enorm ist.

3.8 WAP/WML

Ein Schlagwort, das in den letzten zwei Jahren das Internet maßgeblich geprägt hat, ist die „Mobilität“. Eine neuartige Entwicklung im Bereich der drahtlosen Telefonie trägt dabei die Hauptverantwortlichkeit - die Digital Wireless User Agents, z.B. Handys, die über das Wireless Application Protocol (WAP) auf Angebote aus dem Internet Zugriff haben. Marktforscher prognostizieren, daß bis 2003 weltweit mehr als 530 Millionen WAP-fähige Telefone genutzt werden. Die Hersteller können mittlerweile den durch die Werbung geweckten Bedarf nicht mal annähernd decken. Lange Wartezeiten für interessierte Kunden stehen fest.

Die Möglichkeit, das Handy als Kommunikationshilfe zwischen Laptop und dem World Wide Web zu verwenden, besteht schon lange. Auch werden Handys mehr und mehr für Applikationen jeglicher genutzt. Warum geht man also nicht gleich mit dem Handy in das Internet und stellt die abgerufenen Informationen direkt auf dem Display dar? Was dem im

Wege stand, waren fehlende Protokolle, die speziell auf die Besonderheiten dieser Kommunikationsmittel abgestimmt waren.

3.8.1 WAP

Das WAP-Forum[3], welches aus mittlerweile über 500 Firmen aus der Hardware-, Software- und Kommunikationsbranche gebildet wird, stellte deshalb das **W**ireless **A**pplication **P**rotocol[4] (WAP) vor. Die hinter WAP stehende Technologie ähnelt in ihren Grundzügen sehr der Web-Technologie. Im Prinzip handelt es sich dabei „nur“ um eine Erweiterung. Der einzige jedoch wichtige Unterschied besteht in einem zwischengeschalteten WAP-Gateway bzw. WAP-Proxy, der für die Kodierung bzw. Dekodierung der ankommenden Requests und ausgehenden Responses in binäres WML[5] sorgt. Abbildung 3.1 zeigt, wie WAP-Inhalte zwischen Client und Server über das Gateway übertragen werden.

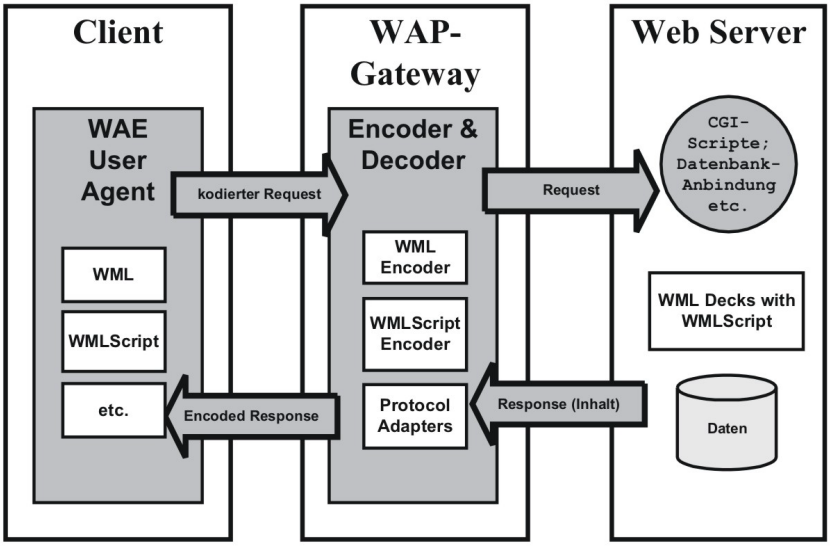

Abbildung 3.1 Wireless Application Environment Model

Wie bei anderen Protokollen üblich, ist auch WAP schichtenförmig modelliert (siehe Abbildung 3.2). In der *Anwendungsschicht* liegt die **W**ireless **A**pplication **E**nvironment

---

[3] http://www.wapforum.org

[4] Die Spezifikation der WAP-Architektur 2.0 ist am 12.03.2001 der Öffentlichkeit zugänglich gemacht worden.

[5] Bedingt durch die begrenzten Ressourcen auf Digital WUA wird als Resultat von Optimierungsbemühungen binäres anstatt textuelles WML verwendet.

(WAE), die auf WWW- und Telefonie-Technologien basierend als Ausführungsumgebung von WAP-Anwendungen dient. In der *Session-Schicht* sorgt das **W**ireless **S**ession **P**rotocol (WSP) für die Bereitstellung von einem verbindungsorientierten und einem verbindungslosen Service. In der *Transaktionsschicht* sorgt das **W**ireless **T**ransaction **P**rotocol (WTP) für die Ausführung von zuverlässigen und unzuverlässigen Transaktionen. Die *Sicherungsschicht* (WTLS für **W**ireless **T**ransport **L**ayer **S**ecurity) dient der Sicherung der Datenintegrität, Privatsphäre sowie der Authentifizierung und bietet zusätzlich einen Schutz vor den „berühmt-berüchtigten" Denial-of-Service-Attacken. In der *Transportschicht* ist das **W**ireless **D**atagram **P**rotocol (WDP) als allgemeiner Transportmechanismus für die Kommunikation zwischen Bearer[6] und den darüberliegenden Schichten verantwortlich.

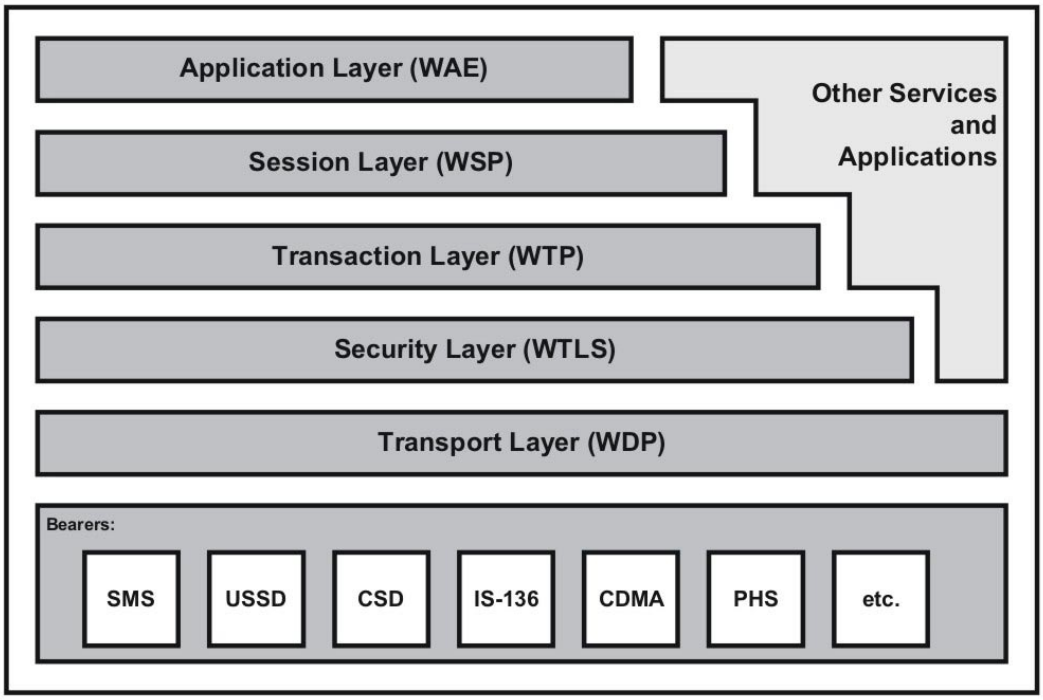

Abbildung 3.2 Das WAP-Schichtenmodell

3.8.2 WML

Nachdem das Problem des fehlenden Übertragungsprotokolls geklärt ist, fehlt jetzt noch eine geeignete Auszeichnungssprache, um die zu übertragenden Informationen in eine darstellba-

[6] Bearer können als drahtlose Schnittstellen zwischen Transportschicht und physikalischen Netzen aufgefaßt werden.

re Art und Weise überführen zu können. Dabei müssen folgende Einschränkungen beachtet werden. Einerseits verfügt das Zielgerät über deutlich weniger CPU-Leistung, hat weniger RAM und ROM zur Verfügung und ist im Vergleich mit einem PC in der Regel nur mit kleineren Displays mit geringer Farbtiefe (meistens 1 Bit Farbtiefe) ausgerüstet. Außerdem sind die Eingabemöglichkeiten für Daten nicht sehr komfortabel gehalten, so entspricht eine Drei- bzw. Vierfachbelegung der Tasten, um Text und Zahlen eingeben zu können, der Norm. Andererseits müssen zusätzliche Probleme, die sich aus geringer Verbindungsstabilität, erhöhten Warte- und Verzögerungszeiten und geringer Bandbreite ergeben, schon bei der Erstellung der zu übertragenden Daten berücksichtigt werden.

Das Resultat der Bemühungen des W3C im Kontext mobiler Anwendungen findet sich in der Spezifikation der neuen, auf XML basierenden **W**ireless **M**arkup **L**anguage (WML) sowie der zugehörigen Scriptsprache *WMLScript* wieder, die in der WAP-Anwendungsschicht abgearbeitet werden (siehe Abbildung 3.2).

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
      <card id="cardOne" ontimer="#cardTwo" title="Demo Page 1">
            <timer value="50"/>
            <p align="center">
                  <b>Hello World !!</b>
            </p>
      </card>
      <card id="cardTwo" title="Demo Page 2" newcontext="true">
            <do type="accept">
                  <go href="#cardThree"/>
            </do>
            <p align="center">
                  <b>This is WML talking to you.</b>
                  <br/><br/>
                  Input your name: <input name="Name"/><br/>
            </p>
      </card>
      <card id="cardThree" title="Demo Page 3">
            <p align="center">
                  <b>Welcome $(Name)!</b>
                  <br/><br/>
                  Thanks for visiting!<br/>
            </p>
      </card>
</wml>
```

Beispiel 3.1 „Hello World“-Beispielanwendung

WML verfügt ähnlich wie HTML über einige grundlegende syntaktische Konstrukte zur Darstellung, Interaktion und Navigation mit dem Benutzer. *Elemente* sind alle Markup- und strukturellen Informationen innerhalb eines WML-Dokuments, wobei sich der Aufbau sowie die zugeordneten Attribute durch die in „http://www.wapforum.org/DTD/wml1_1.1.xml“ spezifizierte DTD definiert sind. Zusätzlich sind Variablen- und Eventhandler-Konzepte integriert, sowie Fähigkeiten zur Timer-gesteuerten Programmierung integriert. Spezielle Eingabe-Tas sowie WMLScript sorgen für Interaktionsmöglichkeiten mit dem Benutzer.

Basiselement der WML-Syntax ist das **Card**-Element (vergleichbar mit dem Layerkonzept in HTML). In ihr werden alle weiteren darzustellenden Inhalt untergebracht. Mehrere dieser Card’s lassen sich wiederum zu einem **Deck** zusammenfassen, welches als logische Einheit betrachtet werden kann und das oberste Element der WML-Struktur darstellt. Bei Empfang eines solchen Decks wird defaultmäßig durch das Endgerät die erste Card angezeigt. In Beispiel 3.1 wird eine „Hello World“-Anwendung vorgestellt, die einen kleinen Ausschnitt der WML-Möglichkeiten geben soll.

Ich diesem Abschnitt habe ich einige weiterführende XML-Techniken beschrieben. Im nächsten Kapitel werden nun sogenannte XML-Server oder XML-Datenbanken vorgestellt. XML-Datenbanken ermöglichen den Element-genauen Zugriff auf zum Teil unheitlich strukturiert vorliegende XML-Dokumente und erlauben damit das Potential von XML als Auszeichnungssprache gegenüber herkömmlichen Technologien besser auszunutzen.

4. DB-Systeme mit XML-Unterstützung

XML-Daten sind regelmäßig hoch komplexe Baumstrukturen, die besondere Anforderungen an die sie verarbeitenden Anwendungen stellen. Relationale Datenbanken erfordern grundsätzlich ein Mapping zwischen den XML-Strukturen und den Tabellen der Datenbank. Die Komplexität ihrer Anwendung wird hierdurch unnötig erhöht. Da sich XML-Bäume mit Objekten sehr gut modellieren lassen, sind hier objektorientierte Datenbanken eindeutig im Vorteil. Das Ablegen von XML-Applikationen in einer Objektdatenbank erfordert keinen mit dem relationalen Ansatz vergleichbaren Paradigmen-Wechsel. Darüber hinaus entstehen beim Einsatz einer relationalen Datenbank Geschwindigkeitseinbußen, zum Beispiel durch zusätzliche relationale Join-Operationen und Index-basiertes Suchen beim Traversieren des XML-Baumes. Kommende Datenbankgenerationen werden zudem in der Lage sein müssen, XML-konforme Schema-Beschreibungen zu verarbeiten. Auf diese Weise wird sichergestellt, daß die bereits erwähnte Flexibilität im Umgang mit neuen Anforderungen tatsächlich erreicht wird. Für Änderungen in der XML-Struktur müssen bislang bei relationalen Datenbanken ständig die Tabellendefinitionen nachgezogen werden.

In Objekt-Datenbanken läßt sich das mit generischen Objektmodellen vermeiden, für die Änderung einer DTD sind dann keine Anpassungen am Schema der OO-Datenbank notwendig. Es ist allerdings unvermeidlich, daß diese Flexibilität zu einem Trade-off führt – je flexibler das Objektmodell, desto geringer ist in aller Regel die Performance. Für die nahe Zukunft ist es daher wünschenswert, die Flexibilität eines generischen Objektmodells mit der Geschwindigkeit eines spezialisierten Ansatzes zu verbinden. So sollte z.B. zur Validierung von XML-Dokumenten ein DBMS integrierte Unterstützung für XML-Parser bieten.

Im nachfolgenden sollen einige Datenbanksysteme, die sich der Speicherung von XML-Dokumenten angenommen haben, vorgestellt werden.

4.1 Objekt-Relationale DBMS und XML

4.1.1 IBM DB2 (UDB)

Durch die Firma IBM wird für das objekt-relationale DBMS DB2 (UDB) ein spezieller XML-Extender angeboten, durch den die Extraktion, das Ablegen und der Zugriff auf und

von XML-Dokumenten im DBMS unterstützt wird [Pors00]. Dabei kann das gesamte XML-Dokument entweder als User-Defined Datatype abgespeichert oder durch anwendungsspezifische Transformationsmechanismen auf relationale Datentypen abgebildet werden. In der **D**ata-**A**ccess-**D**efinition-Datei (DAD) werden Informationen darüber gespeichert, wie die Zuordnung von Elementen und Attributen des Dokuments auf die Attribute der Datenbank erfolgen soll. Auf der Grundlage dieser Spezifikation werden die Dokumente durch den XML-Extender in der Datenbank abgelegt.

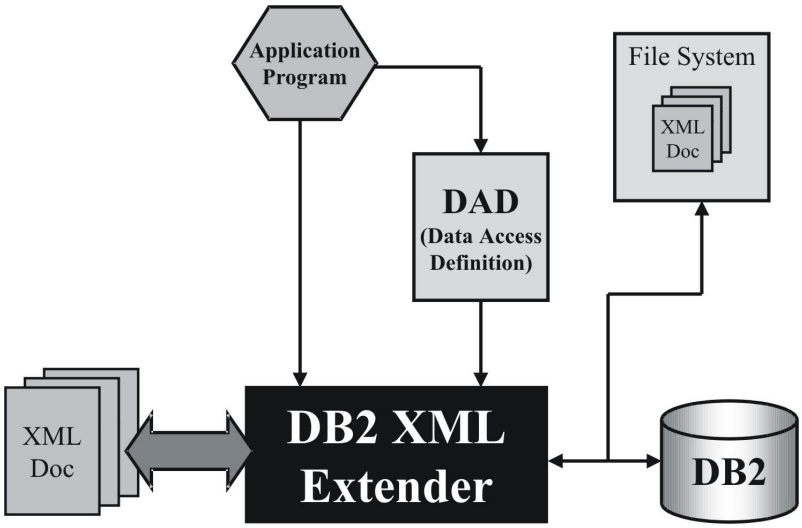

Abbildung 4.1 XML Extender Überblick

Auch der umgekehrte Weg, die Erzeugung von XML-Dokumenten aus der DB2-Datenbank, ist möglich. Auch hier kann der Benutzer das Aussehen der Dokumente beeinflussen, indem er die Zuordnung zwischen den Datenbank-Attributen und Elementen und Attributen eines XML-Dokuments festlegt. Entsprechend dieser Beschreibung werden aus dem Datenbestand der Datenbank Dokumente generiert.

Darüber hinaus wird in einem gewissen Umfang die Anfrageunterstützung und damit verbundene Ableitung von XML-Dokumenten aus der DDB2-Datenbank mittels des Tools XPERANTO gewährleistet. Dabei werden die XML-Dokumente mit Hilfe von XML-QL automatisch aus den objekt-relationalen Daten unter Zuhilfenahme von Informationen aus dem Data Dictionary erzeugt.

4.1.2 Informix

Überblick

Mit dem seit September 1999 verfügbaren „**I**nformix **I**nternet **F**oundation 2000“-**F**ramework (IIFF) hat sich Informix voll und ganz dem Internet als Plattform für zukünftige Entwicklungen im Datenbanksektor verschrieben. Mit Schlagworten wie „Sei als Erster dort!“, „Vergrößere Dich schnell!“ oder „Ständige Verfügbarkeit“ wird für das neue Framework auf den Webseiten geworben. [IIFF01].

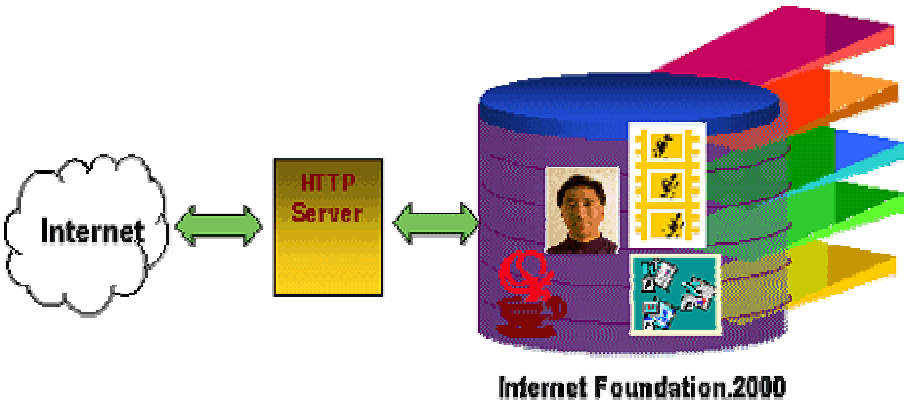

Abbildung 4.2 Informix DataBlade-Technologie

Die für die Entwicklung von Internet-Anwendungen beiden wichtigsten Komponenten des IIFF‘s möchte ich im folgenden kurz vorstellen.

- **Informix J/Foundation**: Dadurch wird in den **I**nformix **D**ynamic **S**erver (IDS) eine **J**ava **V**irtual **M**achine (JVM) integriert, so daß Java Anwendungen nun direkt auf dem Datenbank Server ausgeführt werden können. Dabei wird im Gegensatz zu anderen Anbietern keine „hauseigene“ Realisierung eingebunden, sondern auf Standard JVM’s gesetzt, um so Kompatibilität und eine leichte Migration bestehender Java-Anwendungen zu garantieren.
- **Informix Web DataBlade Module Version 4.0**: Mit der neuen Version des Web DataBlades erweitert Informix die Funktionalität des IDS und des IIFF um wichtige Aspekte, die ein einfaches Entwickeln, Verwalten und Verteilen von Datenbankanwendungen für das Web gewährleisten. Durch die eingebaute Unterstützung von HTML-Datenstrukturen und Multimediainhalten in einer transaktionsbasierten Umgebung (siehe [Ditt00]) wird das Erzeugen von Web-

basierten Anwendungen auf eine einfache Art und Weise ermöglicht. Neu hinzugekommen ist ein erweitertes Session-Management, Browser-basierte Installations- und Adminstration-Tools, sowie volle XML-Unterstützung durch erweiterte MISQL-Tags.

Zusammenfassend kann man sagen, daß im IIFF die XML-Unterstützung durch das Web-Datablade erreicht wird. Durch die Möglichkeit, strukturierte XML-Dokumente im DBMS abzuspeichern, werden dem Anwender umfangreiche Möglichkeiten für Import, Export, Speicherung, Indizierung und Manipulation durch Datenbankabfragen bereitgestellt. XML greift weiterhin in die Bereiche Workflow-Management, was für E-Commerce-Anwendungen von Bedeutung ist, und Metadaten-Speicherung, wobei letzteres einen einheitlichen Zugriff auf die Metadaten gewährleistet, der unabhängig von den beschriebenen Daten ist.

In dem folgenden Teilabschnitt möchte ich nun etwas intensiver auf das Web-Datablade und die damit verbundenen Möglichkeiten, XML in Datenbank-basierten Anwendungen zu nutzen, eingehen.

Web-Datablade

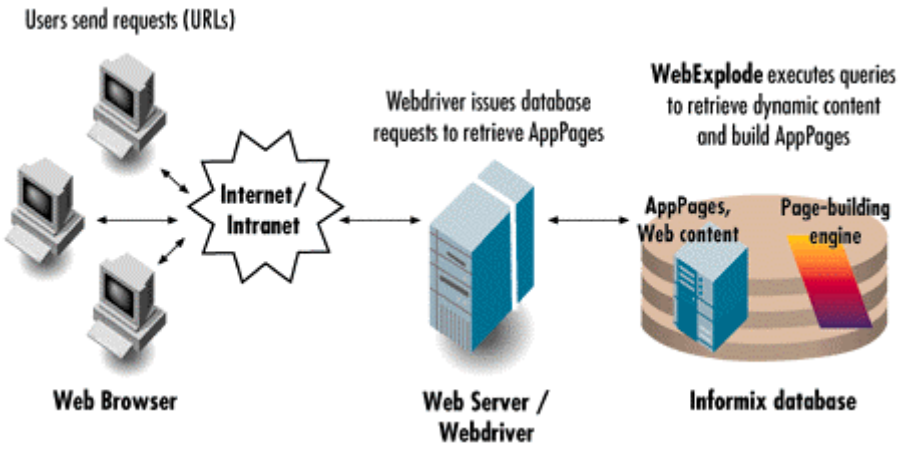

Abbildung 4.3 Aufbau und Funktionsweise des Web-DataBlades

Abbildung 4.3 gibt einen Überblick über die Arbeitsweise des Web-DataBlades. Über das Web-Datablade wird die Funktionalität bereitgestellt, Objekte vom Datentyp HTML anzulegen und ihn durch DataBlade-Tags (**`MIVAR, MIBLOCK, MISQL, MIERROR`**) zu erweitern. (siehe [Ditt00]).

Ziele

1. Der Datenbank-Server muß in Lage sein, XML-Dokumente zu „verstehen“, d.h. analysieren zu können.
2. Es wird ein Transaktionskonzept für Insert-, Update- und Delete-Operationen auf XML-Objekten gefordert.
3. Beim Empfang von XML-Dokumenten muß das DBMS aus der XML-Struktur Teilbereiche extrahieren und in Relationen speichern können.
4. Die XML-Informationen müssen in ihrer hierarchischen Struktur verarbeitet werden können, auch wenn intern mit Relationen gearbeitet wird.
5. Das DBMS muß XML-Dokumente als Ergebnis einer Anfrage-Operation auf hierarchisch, in Relationen oder als Hybrid von beidem gespeicherte XML-Dokumente zurück liefern können.

Unterstütze Speicherungsarten

XML-Struktur: Das XML-Dokument wird dabei in seiner ursprünglichen XML-Struktur atomar als Objekt gespeichert. Das Objekt kann wahlweise als BLOB oder XML-Datentyp abgelegt werden.

Relationale Form: Basierend auf einem „object-to-relational“-Mapping, wird der Inhalt des Dokuments extrahiert und in relationalen Tabellen gespeichert.

Hybrid: Ist es notwendig, nur Fragmente des Dokuments als XML-Struktur zu speichern und die restlichen Informationen in Relationen abzulegen, so ist eine Hybrid aus beiden Speicherverfahren möglich.

Analyse der XML-Dokumente (Datenimport)

Das Zerlegen eines XML-Dokuments in seine Bestandteile wird durch ein XSL-Dokument beschrieben und durch XSLT, in Zusammenarbeit mit dem Web-DataBlade, durchgeführt. So kann mittels XSLT ein XML-Dokument in eine Reihe von Insert-, Update- oder Delete-Update-Operationen zerlegt und die Ergebnis-SQL-Anweisungen über das Web-DataBlade direkt auf der Datenbank ausgeführt werden (siehe Abbildung 4.4).

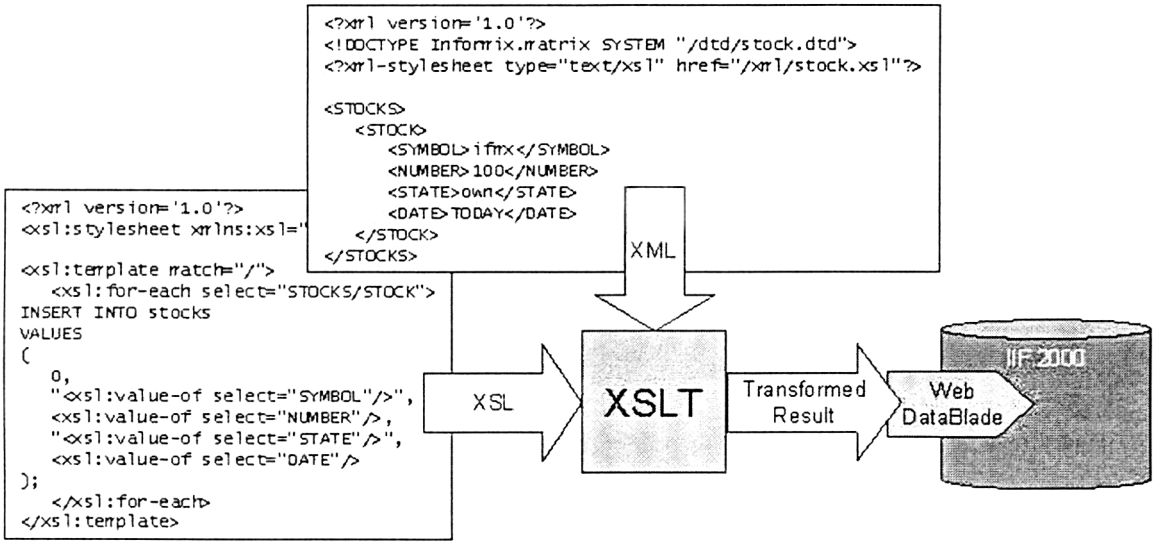

Abbildung 4.4 Datenimport über XSL/ XSLT

XML über das Web-DataBlade

Damit der Browser, die über das Web-DataBlade erzeugten XML-Dokumente auch als solche erkennt, ist notwendig mittels der Anweisung **<?MIVAR>$(HTTPHEADER,content-type,text/xml)<?/MIVAR>** den Mime-Type für den generierten XML-Datenstrom zu setzen. Geht man direkt über das „Application-Page-Builder"-Tool kann man dort beim Anlegen eines neuen Objekts den Mime-Type explizit festlegen.

Durch die Möglichkeit, innerhalb des Web Datablade SQL-Tags (**<?MISQL><?/MISQL>**) Formatierungsanweisungen für die selektierten Daten festlegen zu können, kann ein wohlgeformtes XML-Dokument generiert werden.

```
<?MIVAR>$(HTTPHEADER,content-type,text/xml)<?/MIVAR>
<?xml version="1.0" ?>
<DISPLAY>
<?MISQL SQL="select lastname, firstname from personTable">
      <ROWS>
            <lastname>$1</lastname>
            <firstname>$2</firstname>
      </ROWS>
<?/MISQL>
</DISPLAY>
```

Beispiel 4.1 Beispiel Skript, um ein XML-Dokument aus der Datenbank zu generieren

Das MISQL-Tag kann zusätzlich noch die Attribute XMLDOC (bestimmt das Wurzel-Element), XMLVERSION und XMLROW (dem hier angegebenen Element wird automatisch eine fortlaufende Numerierung für jedes Ergebnis zugewiesen). Beispiel 4.1 könnte damit durch folgende Zeilen realisiert werden.

```
<?MIVAR>$(HTTPHEADER,content-type,text/xml)<?/MIVAR>
<?MISQL XMLDOC="DISPLAY" SQL="select lastname, firstname from
personTable"><?/MISQL>
```

Beispiel 4.2 Eine etwas kürzere Realisierung von Beispiel 4.1

4.1.3 Oracle8i

Mit Hilfe des Oracle XML Developer's Kit (XDK) steht ebenfalls ein Werkzeugsystem zur Verfügung mit dessen Hilfe die Speicherung und Generierung von XML-Dokumenten in bzw. aus Oracle-Datenbanken möglich ist [Orac00]. Dieses bietet XML-Parser an, die den Benutzer unterstützen, XML-Dokumente zu parsen und in Oracle-Datenbanken abzubilden. Dabei sind grundsätzlich drei Varianten unterscheidbar. Dies bezieht sich auf die Speicherung der strukturierten Anteile in Relationen und Attribute, für unstrukturierte Anteile als CLOB oder BLOB und als drittes eine Kombination aus diesen.

Weitere Informationen über die Speicherung von XML-Daten in ORDBMS sind unter [Timm99, KlMe00] nachzulesen.

4.2 XML-Server

4.2.1 Tamino

Der von der Software AG entwickelte XML-Server dient der Speicherung, Verwaltung und Verarbeitung von Text-, Bild- und Audiodaten. Tamino speichert im Gegensatz zu den beschriebenen ORDBMS die XML-Dokumente direkt im XML-Format, wobei Dokumente mit und ohne DTD gespeichert werden können. Anfragen werden in einem XQL-Dialekt realisiert. Als zusätzliche Funktionalität wird der Zugriff auf externe relationale Datenquellen unterstützt.

Herzstück von Tamino ist die X-Machine, ein vollständiges DBMS für XML-Dokumente und andere Daten. XML-Dokumente werden in der Tamino-eigenen XML-Datenbank ge-

speichert, während andere Daten (z.B. aus relationalen Tabellen) auch in originären Datenquellen abgelegt werden können. Die Metadaten zu Struktur und Objekten von XML-Dokumenten sowie zu den Daten aus externen Datenquellen sind in der Data Map gespeichert, dem Repository von Tamino. Weitere Elemente der X-Machine sind der XML-Parser, der XQL Query Interpreter, der Object Processor, der Object Composer und die Utilities.

Der Object Processor ordnet abzuspeichernde Objekte entweder der XML-Datenbank, der relationalen Datenbank oder über die X-Nodes externen Datenquellen zu. Das Gegenstück hierzu ist der Object Composer, der aus den genannten Quellen die nachgefragten Daten selektiert und in XML-Dokumente zusammenfügt. Unter den Utilities sind verschiedene Dienstprogramme z.B. für das schnelle Laden von XML-Dokumenten, die baumartig strukturiert sind. Weitere Bausteine von Tamino sind die SQL Engine, der Tamino Manager, Tamino Server Extensions und Tamino X-Nodes.

4.2.2 ObjectDesign eXcelon

eXcelon ist eine vollständige Entwicklungsumgebung für XML-basierte Anwendungssysteme. Es besteht aus den Bausteinen Toolbox, Data Server und Xconnetcs. Die Toolbox umfaßt Werkzeuge zur Modellierung und Entwicklung von XML-Anwendungen sowie zur Verwaltung der Datenbasis und des Systems selbst. Der Data Server ist das DBMS von eXcelon und damit das Herzstück des Systems. Mit der Xconnect-Funktionalität können Daten aus einer Vielzahl externer und unterschiedlicher Daten-Quellen importiert und exportiert werden. Die Toolbox enthält vier Werkzeuge (Studio, Explorer, Manager, Stylus) mit denen überwiegend grafisch XML-basierte Web-Applikationen und Datenstrukturen entwickelt und verwaltet werden können.

XML-Dokumente werden intern als XML-Datentypen in der ObjectStore-Datenbank abgelegt. Als Anfragesprache dient XQL, zusätzlich besteht eine grafische Unterstützung bei der Anfrageformulierung. Durch die Verwendung von XSL-Stylesheets können die Dokumente oder Teile der Dokumente sichtbar gemacht werden.

4.2.3 POET Content Management Suite

Mit der Content Management Suite (CMS) wurde durch POET ein System bereitgestellt, das auf dem POET Object Server aufsetzt und geeignet ist, komplexe Medientypen zu speichern und zu verwalten. Dazu gehören neben XML und SGML auch Grafik-, Audio- und Videoformate. Gespeicherte XML-Dokumente werden auf objektorientierte Datenbankstrukturen

abgebildet. Hervorzuheben ist hier, daß eine Volltext-Suchfunktionalität bereitsteht, die für viele Dokumenttypen, die größere Textanteile enthalten, erforderlich ist.

4.2.4 Bluestone XML-Server

Dieser XML-Server ist vollständig in eine Java-Umgebung eingebettet. Die zentrale Komponente des Systems ist der Dynamic XML Application Server, der als Middleware für die Integration von Unternehmensanwendungen konzipiert ist. Im folgenden werden die einzelnen Bausteine des XML Application Server kurz vorgestellt:

Anfragen eines Clients werden direkt an den Application Server gerichtet. Die Communication Services sind als Framework implementiert, das bereits eine Vielzahl von Schnittstellen zu gängigen Protokollen bereitstellt. Ein Document Handler implementiert die Dokumentspezifische Anwendungslogik, die für die Verarbeitung verschiedener XML-Dokumente notwendig ist. Ein solcher Handler übernimmt die übergebenen Daten, wandelt sie ggf. in XML um und verarbeitet sie entsprechend der in Java implementierten Verarbeitungsregeln. Die Document Handler stützen sich funktional auf den XML Core Services ab, in denen alle Funktionen zur Verarbeitung von XML zur Verfügung gestellt werden. Hierzu gehört das Parsen von XML-Dokumenten, die Verarbeitung von XSL und XQL sowie von DOMlets. Weitere Funktionen des Dynamic XML Application Servers sind das Session Managements, Live Object Cache, Security, State Server.

Mit Visual XML ist eine Entwicklungsumgebung für XML-Applikationen auf Basis des XML-Servers verfügbar. Weiterhin können Instanzen des Dynamic XML Application Server konfiguriert, generiert und administriert werden.

4.3 Fazit

Aus den oben beschriebenen XML-Datenbanksystemen sind eine Reihe von Gemeinsamkeiten erkennbar, aus der sich eine verallgemeinerte Refernzarchitektur ableiten läßt. Als wesentliche Architektur-Bausteine sind folgende zu nennen:

- *Basisdienste*: Als Basisdienste stehen XML-Parser sowie XQL- und XSL-Interpreter zur Verfügung.
- *XML-Datenbank*: Die XML-Dokumente und XSL-Stylesheets werden in einer nativen XML- oder objektorientierten Datenbank verwaltet. Alternativ sind auch ORDBMS mit objekt-relationalen Konzepten denkbar (mit Einschränkungen).

- *Management-Dienste*: Benutzer, Zugriffsrechte, Hauptspeicher und sonstige Konfigurationen werden über Verwaltungsdienste administriert.
- *Interfaces zu externen Datenquellen*: Die Nutzung von externen Datenquellen wie SQL-Datenbanken oder ERP-Systemen wird über ein Schnittstellensystem ermöglicht.
- *Extensionen*: Die Funktionalität kann durch individuelle Programme erweitert werden.

XML-Datenbanksysteme und spezialisierte XML-Server, die über derartige Bausteine verfügen, sind als Werkzeuge und Frameworksysteme grundsätzlich für die Entwicklung und den Betrieb von XML-basierten eBusiness- und Content Management-Applikationen geeignet. Sie verbinden konventionelle Daten-Management-Funktionalität mit der XML-Basistechnologie und bilden somit die Grundlage für zukünftige multimediale Informationssysteme (siehe auch [Wate01, STH+99, FloKo99, Cour00, AbBs00]).

5. System-Konzeption

In den bisherigen Kapiteln dieser Diplomarbeit wurden neben den verschiedenen XML-Standards auch die jeweiligen Möglichkeiten der Speicherung und Generierung von XML-Dokumenten in XML-Datenbanksystemen bzw. spezialisierter XML-Server untersucht.

In diesem Kapitel möchten wir nun einen Überblick über das entwickelte, neuartige Konzept einer intelligenten und generischen Zugriffssteuerung auf multimediale, XML-gesteuerte Dokumentstrukturen in objekt-relationalen DBMS geben. Ziel war u.a. die anwendungs- und endgerätespezifische Darstellung und Präsentation multimedialer Dokumentstrukturen (z.B. für mobile Umgebungen). Für das Experimentalsystem und die praktischen Realissierungsarbeiten wurde eine projektspezifisches Anwendungsszenario zu Grunde gelegt.

5.1 Einordnung

Die in diesem Abschnitt vorgestellte Realisierung ist Teil eines Prototypen, den ich mit anderen Mitarbeitern am Zentrum für Graphische Datenverarbeitung Rostock e.V. (ZGDV) im Rahmen der Projekte MOVI (**Mo**bile **Vi**sualisierung) und iVip („Integrated Virtual Product Development“) erarbeitet und umgesetzt habe.

5.1.1 Movi

Das Projekt MOVI [HKK+95, HeLu96] ist ein Gemeinschaftsprojekt des ZGDV Rostock, der Universität Rostock, des IGD Rostock und der TH Darmstadt, welches von der Deutschen Forschungsgemeinschaft gefördert wird. Das ZGDV Rostock bearbeitet im Rahmen dieses Projektes die folgenden vier Schwerpunkte.

Im Arbeitspaket *Austauschstrategien* werden Strategien zum optimierten Austausch multimedialer Informationen untersucht. Ein wichtiges Kriterium ist hierbei die Beachtung wechselnder Netze und unterschiedlicher Übertragungsbandbreiten.

Gerade für den Bereich mobiler Infrastrukturen ist eine spezielle Anpassung bzw. Entwicklung von Datenbankverfahren für die Verwaltung und Handhabung der multimedialen Datentypen außerordentlich wichtig. Das Arbeitspaket *Multimedia-Datenmanagement* beschäftigt sich mit diesem Gebiet.

Der Bereich *Content-Based-Retrieval* ist auf die Einbettung von Techniken des Content-Based-Retrieval in die mobile Anfragebearbeitung und die mobile Visualisierungsarchitektur ausgerichtet [BöLu97, BöFl98, Fla99a, Fla99b, FlGü00].

Einen besonderen Schwerpunkt bildet das Arbeitspaket *Datenreduktion*. Hier wird untersucht, wie sich neue datenbankorientierte Reduktionsmechanismen mit im Rahmen von MOVI entwickelten datenbankexternen Reduktionsmechanismen (z.B. **L**evel-**o**f-**D**etail für Bilddaten; LoD) in das übergreifende LoD-Modell integrieren lassen, um so den quellentransparenten LoD-orientierten mobilen Zugriff auf Multimedia-Datenstrukturen zu unterstützen

5.1.2 iViP

Im Rahmen des iViP-Projektes geht es um die Bereitstellung von Gestaltungswerkzeugen, den Aufbau und die Verwaltung virtueller Produkte und Werkzeuge für die virtuelle Fertigungserprobung. Sie sollen die durchgängige digitale Abbildung des entstehenden Produktes im gesamten Produktlebenszyklus ermöglichen. Das ZGDV Rostock ist dabei in vier Teilprojekte eingebunden [FHK+98]. So wurde u.a. der Zugriff aus der iViP-Integrationsplattform heraus auf die ORDBMS-Erweiterungen (z.B. Volltextsuche auf techn. Dokumente und Normen) realisiert. Dazu erfolgt zunächst eine Registrierung dieser DBMS-Funktionalität (Anmeldung an die Integrations-Plattform) über entsprechende CORBA-Services. Das User-Interface wurde als JavaBean realisiert (Siehe Abb 2). Der Zugriff auf das Informix-System, erfolgt über eine JDBC-Schnittstelle.

Im Rahmen eines weiteren Teilprojektes wurde in Kooperation mit einem Projektpartner aus der Automobil-Industrie von mir der Prototyp „DigTeD“ („**Dig**itales Archiv **te**chnischer Text-**D**okumente“) entwickelt, durch den eine integrierte Speicherung, Verwaltung und Volltextsuche über PDF-Dokumente (Normbeschreibungen, techn. Prüfbestimmungen etc.) ermöglicht. Ziel war der Aufbau eines datenbankbasierten, digitalen Archives (Siehe Abbildung 5.3). Als technische Basis wurde das Oracle8i-System (Version 8.1.5) in Kombination mit der Oracle *Inter*MediaText-Erweiterung [Orac99] genutzt.

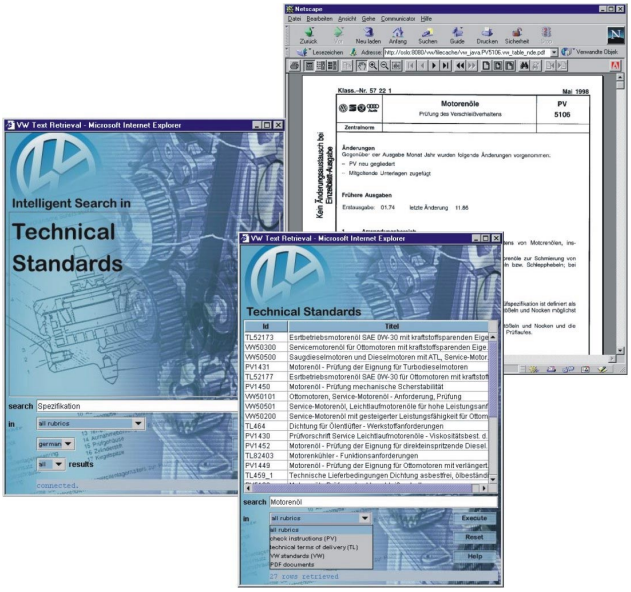

Abbildung 5.1 Prototyp für „DigTeD“

5.2 Anwendungsszenario & Zielsetzung

Im Rahmen dieser Diplomarbeit wurde der in obigem Projekt verwendete Datenbestand weiterverwendet, um einen Prototypen zu entwickeln, der

1. den Datenaustausch über eine XML-Schnittstelle realisiert, um so nach außen hin offen zu bleiben für die Anbindung neuer Anwendungen,
2. über geeignete Schnittstellen verfügt, mit denen die Anbindung neuer Datenquellen ermöglicht wird

und

3. spezielle Transformationsverfahren benutzt, um so ein Mapping der Anfrageergebnisse auf die visuellen Möglichkeiten der Endgeräte zu gestatten.

In dem entwickelten Prototyp werden dabei Konzepte umgesetzt, die sich vor allen auf den Bereich der mobilen Endgeräte wie Palm-Handhelds oder WAP-fähige Handys ausrichten. In Bezug auf das MOVI-Projekt läßt sich die Realisierung bei den Teilaspekten II, III und IV einordnen.

Leider zeigte es sich, daß die zur Verfügung stehende Datenbasis eine sehr ungünstige Mischung aus in PDF-Dokumenten gespeicherten Bilddaten war, die sich so für den Einsatz von Content-based-Retrieval-Techniken [ANSI95, BaRi99] nicht eignete. Die Daten wurden

deshalb in der Vorbereitungsphase mittels OCR-Software (Caere OmniPage 9.0) in das ASCII-Format überführt und zusätzlich die vorhandene Datenbasis semi-automatisch um eine Kurzbeschreibung (Abstract) ergänzt.

Unter dem Aspekt der Portabilität wurden die Server-Komponenten der Architektur auf der Backend-Seite in JAVA (JDK 2.0) implementiert. Speziell für die spätere Realisierung auf dem Palm IIIc wurde für die Client-Implementierung die J2ME CLDC **K**ilobyte **V**irtual **M**achine Version 1.0 (KVM) von SUN. Bei der KVM handelt es sich um die erste Referenzimplementierung der **C**onnected **L**imited **D**evice **C**onfiguration (CLDC) von Sun. Im Gegensatz zum vollen JAVA-Standard verfügt die KVM über ein reduziertes Klassenmodell. Das hängt damit zusammen, daß sie hauptsächlich auf Plattformen mit wenig Speicher und reduzierten bzw. eingeschränkten Ressourcen ausgelegt ist. Abbildung 5.2 gibt einen Überblick, wo die kVM im Java-Spektrum einzuordnen ist.

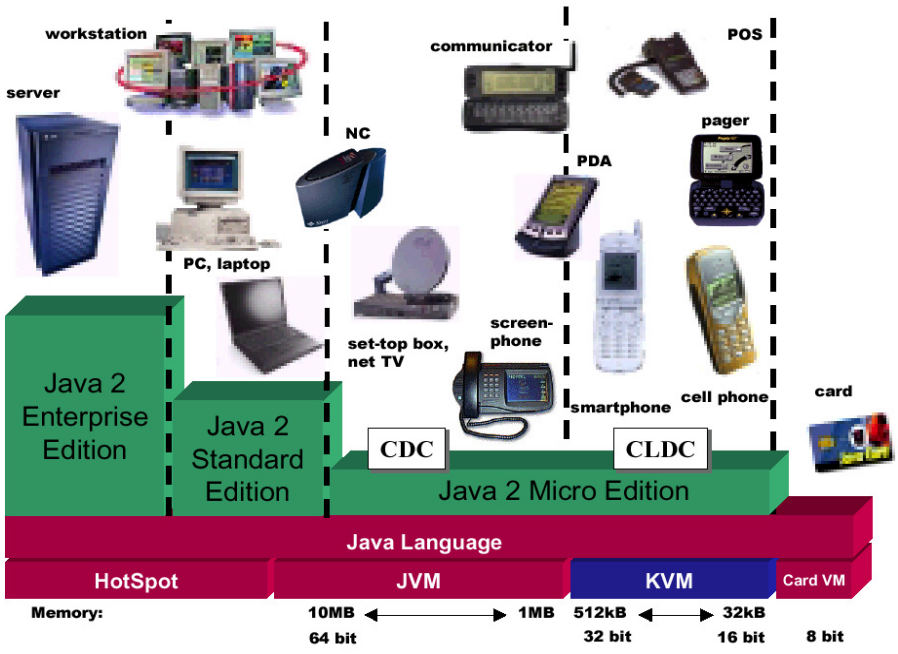

Abbildung 5.2 Java-Pakete und Ihre Zielmärkte (Quelle: SUN)

Durch das reduzierte Klassenmodell der KVM kommt es allerdings einigen Inkompatibilitäten in Bezug auf die Java Userinterface Komponenten und eingeschränkte Eventhandling-Möglichkeiten, die die angestrebte Portabilität bezüglich des Clients erschwerten. Wir haben uns deshalb entschieden , zusätzlich zur KVM noch das KAWT (**k**ilo-

byte **A**bstract **W**indow **T**oolkit) Version 0.81 der „Kroll & Haustein GbR“[7] zu verwenden. Durch das KAWT werden diese Nachteile insoweit ausgeglichen, als das stellt eine Erweiterung der KVM um GUI[8]-Komponenten dar, die leider von SUN weder berücksichtigt worden sind noch nachgereicht werden sollen.

Aufbauend auf einer Client-Server-Architektur mußte ein XML-Framework konzipiert und realisiert werden, mit dem es möglich sein sollte, verschiedene Datenbankquellen in das System zu integrieren. Voraussetzung für die Integration der Datenquellen in das System ist das „Einbringen“ von Metadaten, über die einerseits die interne Datenbankstruktur (Tabellenstruktur, Beziehungen und Zusammenhänge zwischen den einzelnen Tabellen) sowie zusätzliche Informationen (verwendete JDBC-Treiber, URL der Datenbank, Benutzerinformationen, Datenbankfähigkeiten) ermittelt werden können.

5.3 Architektur & Realisierung

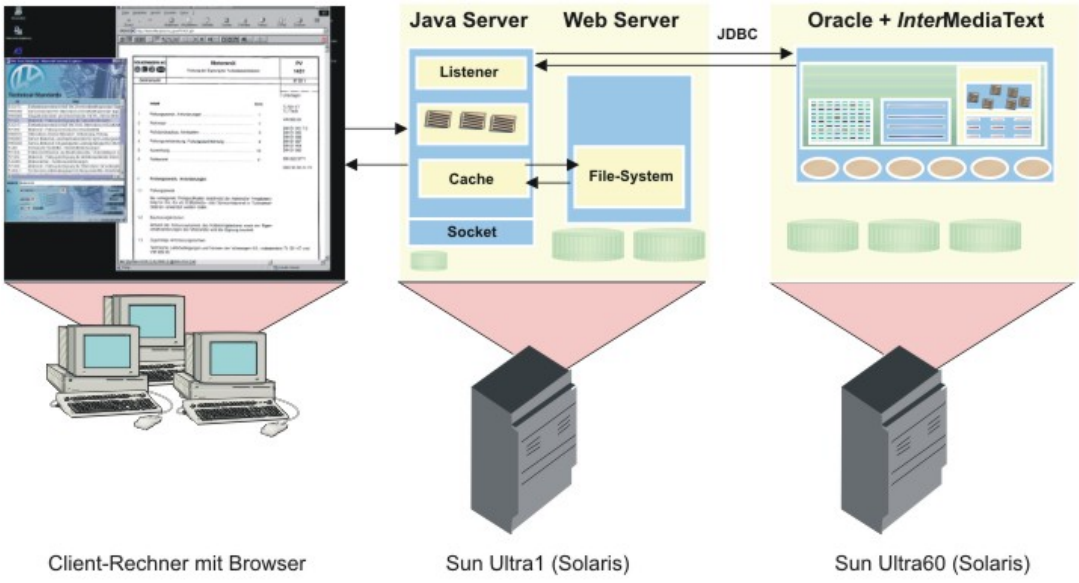

Abbildung 5.3 im Rahmen von iViP entwickelte Client-Server Lösung

Entsprechend der zuvor erläuterten Anwendungszielstellungen (MOVI-, iViP-Projekt als Anwendungskontext) wurden innerhalb dieser Arbeit auch die erforderlichen Bausteine und

[7] http://www.trantor.de/kawt/

[8] Graphical User Interface

Funktionen realisiert, um die system- und geräteunabhängige Verarbeitung der Daten zu gewährleisten. Die auf Client- sowie auf Server-Seite realisierte XML-Schnittstelle (siehe Abbildung 5.4) bietet dafür die Möglichkeit der Transformation von verschiedenen Informationsquellen in eine Zielstruktur. Unter Nutzung dieses Mechanismus wurden im Rahmen dieser Diplomarbeit vordefinierte XSL-Stylesheets (Templates) der integrierten DaS-Komponente [CoFl00] zur Verfügung gestellt, die die Publikation auf unterschiedliche Endgeräte und Ausgabeformate im Anwendungskontext ermöglicht. Der Client wurde dabei so realisiert, daß er auf den unterschiedlichsten Endgeräten ablaufen kann. Es ist einerseits möglich, daß auf Geräten wie einem Palm-Handheld oder dem PC der Client als selbständige Anwendung ausgeführt wird. Andererseits kann der Client aber auch als Prozeß auf einem WWW-Server laufen, der Schnittstellen für WAP-Geräte oder herkömmliche HTML-Browser bietet. Für diese beiden Aspekte wurden Prototypen entwickelt, die in Abschnitt 5.4 näher erläutert werden.

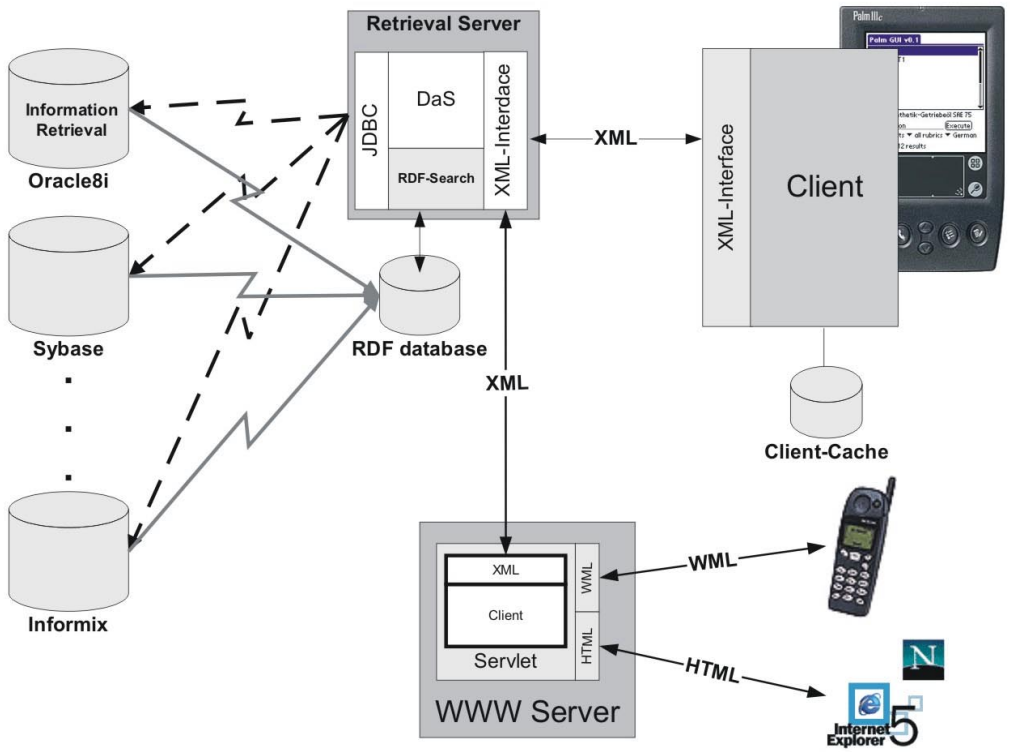

Abbildung 5.4 XML Framework Architektur

Für die Anwendung in mobilen Umgebungen ist der Leistungsumfang und das Konzept von XML/XSLT jedoch nicht ausreichend. Mobile Geräte sind hinsichtlich der Größe der Do-

kumente und der darstellbaren Datentypen eingeschränkt. Eine Selektion der darstellbaren Datentypen aus einem XML-Dokument reicht zur Präsentation des Inhaltes oft nicht aus. Darum ist es notwendig, die Systemlösung so intelligent zu gestalten, daß neben der Selektion der Inhalte auch eine automatische Transformation auf den entsprechenden Datentypen erfolgen kann (Daten-Preprocessing). So kann z.B. ein WAP-Gerät als Bild nur Dateien im WBMP-Format (Wireless Bitmap) anzeigen. Befinden sich wichtige Informationen in Bilddateien, muß diese für die Anzeige in einem WAP-fähigen Gerät in Größe und Format angepaßt werden. Hier können nun in einer ersten Stufe, bereits verfügbare Multimediaerweiterungen z.B. der Objekt-relationalen Datenbanksysteme (ORDBMS) wie Informix und Oracle8i genutzt werden. Diese versprechen entscheidende Fortschritte hinsichtlich des Daten-Preprocessings z.B. Skalierung, Konvertierung, Komprimierung der verschiedenen Multimediadatentypen. Hauptgrund dafür ist, daß es mit Hilfe benutzerdefinierten Datentypen nun möglich ist, die Semantik von multimedialen Daten zu erfassen. Multimedia-Objekte brauchen also nicht mehr in Form von uninterpretierbaren BLOBs abgespeichert zu werden. In realen ORDBMS wird multimediale Funktionalität zumeist in Form vorgefertigter Module (Informix DataBlades, Oracle Cartridges) integriert. Dabei handelt es sich um eine Sammlung von neuen Datentypen, zusammen mit den dazugehörigen Funktionen, Integritätsbedingungen, Indexmechanismen etc. Die interne Struktur der neuen Datentypen bleibt nach außen völlig verborgen, der Zugriff ist nur über die entsprechenden Funktionen möglich. Es wird also das objektorientierte Prinzip der Kapselung ausgenutzt, indem eine Schnittstelle zur Verfügung gestellt wird, welche ausschließlich auf benutzerdefinierten Funktionen basiert. Innerhalb des Vorhabens werden einzelne Funktionsbausteine (Datenbankserver-Erweiterungen) bereitgestellt, die medienspezifische Transformationsfunktionalität und Algorithmen für ausgewählte Dokumenttypen enthalten.

So kann im Rahmen der vorher erläuterten Konzeption die Content Transformation auf Serverseite auf eine äußerst effektive Weise unterstützt werden. Die Systemlösung basiert somit auf einem zweistufigen Ansatz bei der die Multimedia Content Transformation sowohl auf struktureller Ebene (XML/XSLT) als auch auf den Datenelementen selbst stattfindet. Die Funktionsbausteine des Content-Mediators werden dabei direkt durch die Device- und Kontext-Informationen des RDF-Content-Repositories gesteuert.

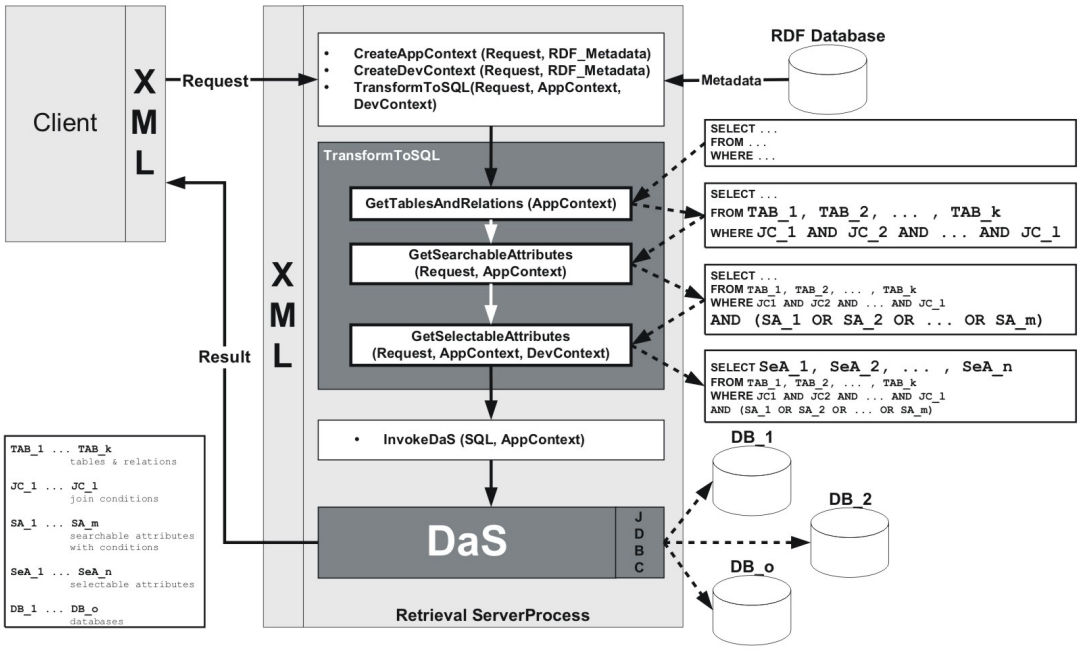

Abbildung 5.5 Anfragebearbeitung und Transformation

Abbildung 5.5 gibt einen Überblick darüber, wie die Client-Server-Anfrageverarbeitung abläuft und welche Teilschritte notwendig sind, um aus dem Client-Request die SQL-Anfrage zu generieren, mit der dann auf die Datenbank zugegriffen wird.

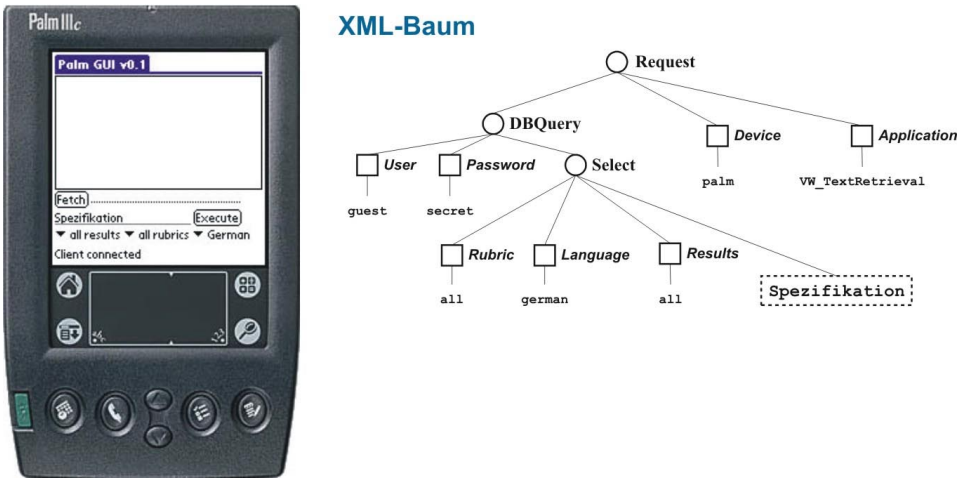

Abbildung 5.6 Requestaufbau auf dem Client

Entsprechend der Abbildung 5.5 wird zu Beginn vom Client aus den zu übertragenden Parametern (Abbildung 5.6) ein XML-Dokument erzeugt, welches für den späteren Datenaustausch mit dem Server-Prozeß verwendet wird. Dafür wird auf Client-Seite das benutzte Endgerät ermittelt, welches entweder über einen Parameter den auf Server-Seite bekannten

Gerätekontext bestimmt (Szenario: Client läuft als eigenständige Applikation auf dem Endgerät) bzw. über zusätzliche Mechanismen Informationen über die Display-Auflösung und –Farbtiefe, Arbeitsspeicher und graphische Fähigkeiten der jeweiligen Umgebung erhält (Szenario: Client läuft als Prozeß auf dem Web-Server und kann wahlweise über die WML bzw. HTML-Schnittstelle angesprochen werden).

Sind alle relevanten Daten durch den Client aufbereitet worden (siehe Abbildung 5.6), wird der Request an den Applikations-Server (z.B. „Retrieval-Server“) gesendet. Auf Client- sowie auf Server-Seite sind dafür XML-Schnittstellen notwendig, über die der Datenaustausch vollzogen wird. Die eingehende Anfrage des Clients wird Server-seitig analysiert und mit einer speziellen „Kommando-DTD“ (siehe Anhang) validiert. Ist dies erfolgreich geschehen, werden verschiedene Prozesse gestartet, in denen die Anfrage verarbeitet und eine entsprechende Ergebnis-XML-Struktur erzeugt wird.

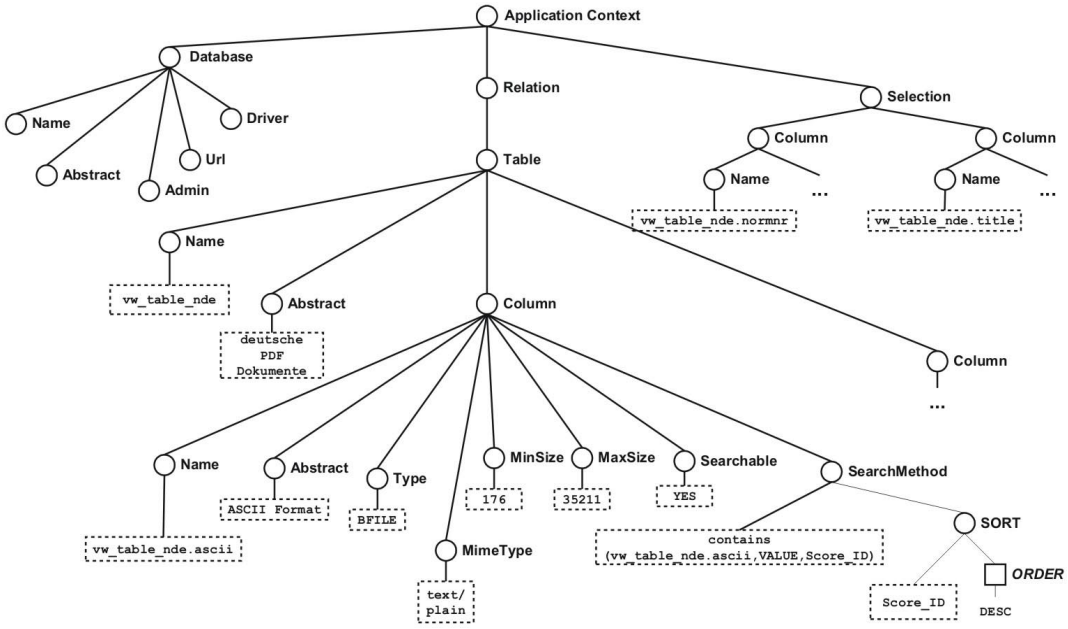

Abbildung 5.7 Anwendungs-Kontext

Abhängig von der eingehenden Anfrage des Clients wird nun im ersten Schritt der Anwendungskontext erzeugt. Dafür müssen die jeweiligen Informationen aus dem RDF-Content-Repository gewonnen werden, um den Datenbankinhalt für die SQL-Anfrage reproduzieren zu können. Die daraus gewonnene Struktur (siehe Abbildung 5.7) stellt wiederum ein XML-Dokument dar und wird so den weiteren Teilprozessen zugänglich gemacht. In dieser Struk-

tur sind Informationen (Metadaten) über die betroffene Datenbank wie z.B. Datenbankname, benötigter JDBC-Treiber und Datenbankbeschreibung enthalten. Außerdem werden Daten über die zu benutzenden Tabellen oder Joins abgelegt, um darüber später die SQL-Anfrage erzeugen zu können. Zu den Informationen, die im Tabellen-Zweig gesammelt werden, gehören der Name der jeweiligen Tabelle, eine kurze abstrakte Beschreibung sowie Informationen über die für die Anfrage relevanten Attribute/ Spalten der Tabelle. Die Spaltenbeschreibung enthält wichtige Informationen über die Daten wie interner Datentyp, Mime-Typ und Minimum und Maximum bzgl. der gespeicherten Datengröße. Dabei werden Minimum und Maximum verwendet, um zu bestimmen, ob Attributwerte auch bei geringer Bandbreite übertragen werden können oder ob gegebenenfalls noch Transformationsmethoden angewendet werden müssen, welche die Daten reduzieren (Daten-Preprocessing) oder im schlechtesten Fall „ausblenden“ (Projektion). Die im DBMS verfügbaren Transformationsmethoden werden dabei im Database-Zweig des Anwendungskontextes gespeichert.

In einem ähnlichen Prozeß wird danach die XML-Struktur für den Gerätekontext unter Zuhilfename der in der RDF-Datenbank gespeicherten Daten erzeugt. Er enthält die oben beschriebenen Informationen über Geräte-spezifische Parameter.

Nachdem diese drei Strukturen (Request, Anwendungskontext, Gerätekontext) erzeugt worden sind, kann jetzt die Transformation des Requests in eine äquivalente SQL-Anfrage erfolgen (siehe Abbildung 5.5). So werden im ersten Teilschritt aus den im Anwendungskontext gespeicherten Informationen über die zu verwendenden Tabellen und deren Beziehungen untereinander die entsprechende FROM-Klausel und die WHERE-Klausel gebildet, die benötigt werden, um die jeweilige Datensicht (etwaige Joins) nachzubilden. Der zweite Schritt ergänzt die WHERE-Klausel um die Vergleichstermini, die aus dem Requestkontext und dem Anwednungskontext gewonnen werden. Der abschließende dritte Schritt fügt die durch die vollständige SQL-Anfrage zu selektierenden Attribute ein. In diesem Teilschritt erfolgt dann auch die Entscheidung in welcher Art und Weise einen Datenreduktion in Abhängigkeit vom Endgerät erfolgen muß.

Nachdem die SQL-Anfrage generiert und wiederum in einen XML-Baum überführt worden ist, wird die DaS-Schnittstelle aufgerufen. Innerhalb dieser Komponente wird die SQL-Anfrage nun an die Datenbank versendet und die Ergebnisse mittels XSL-Stylesheets transformiert in eine Ergebins-XML-Struktur umgewandelt und zurück an den Client übergeben

und dort visualisiert oder im Falle des Webservers über einen nochmaligen Transformationsprozeß in das entsprechende Zielformat (HTML, WML) konvertiert.

Einige interessante weitergehende Realisierungsapekte, die insbesondere die Kopplung von XML mit der Information-Retrieval-Technologie betreffen, sind in [SiTW01, SDT+97, StDe01] weiter beschrieben. Sie wurden jedoch nicht weiter im Rahmen der Diplomarbeit berücksichtigt, da in dem hier verfolgten Ansatz die dem DBMS zur Verfügung stehende Information-Retrieval-Funktionalität genutzt werden sollte.

5.4 Prototypen

Im Rahmen der Diplomarbeit wurde wie in Abschnitt 5.3 aufgezeigt eine Client-Server Architektur entwickelt, die durch die Verwendung von XML als Datenaustauschformat vom verwendeten Endgerät getrennt und andererseits Transformations- und Datenreduktionsfunktionalität implementiert wurde.

Abbildung 5.8 Palm-Lösung

Neben Teilentwicklungen bei der Konzeption und Realiserung des Server-Backends wurden von mir der Client entwickelt, der sowohl in den eigenständigen Applikationen als auch auf dem Webserver benutzt wird. Neben der herkömmlichen Browserunterstützung kann die Clientanwendung nun zum Beispiel auf einem Palm (siehe Abbildung 5.8) oder auf einem WAP-fähigen Handy (siehe Abbildung 5.9) zum Einsatz kommen und kann über entspre-

chende XSL-Stylesheets relativ einfach und schnell auf jeweilige Zielgerät angepaßt werden..

Abbildung 5.9 WAP-Lösung

6. Zusammenfassung und Ausblick

Ziel dieser Arbeit war es, den adäquaten Zugriff auf multimediale, XML-gesteuerte Dokumentstrukturen in objekt-relationalen DBMS zu untersuchen und diesbezüglich einen geeigneten, anwendungsorientierten Ansatz zu entwerfen und zu implementieren.

Trotz der inzwischen komfortablen Möglichkeiten bzgl. der Speicherung, Verwaltung und Recherche multimedialer Datentypen in ORDBMS stellt die anwendungsspezifische Darstellung und Repräsentation multimedialer Dokumentstrukturen (z.B. für mobile Umgebungen) eine Herausforderung dar. Benötigt wird demnach eine Schnittstelle zum jeweiligen DBMS, die eine Nutzung der objekt-relationalen Multimedia-Fähigkeiten zuläßt und die Ableitung der jeweiligen anwendungsabhängigen Präsentation ermöglicht. Hier bietet nun gerade XML als universelles Datenaustauschformat durch die Trennung von Struktur und Repräsentation neue Lösungsmöglichkeiten.

Im Rahmen dieser Diplomarbeit wurden zur Lösung der obigen Problemstellung zunächst bestehende Standardisierungsbestrebungen untersucht (u.a. XML, XSL, XQL, WML) und anhand eines ausgewählten, konkreten Anwendungsszenarios wurde anschließend ganzheitlich die Lösungsstrategie validiert und dabei folgende Ergebnisse erzielt:

- Ein wesentlicher Aspekt im Rahmen des Lösungsansatzes war das Daten-Preprocessing bzw. die serverseitige Multimedia-Content-Transformation. Dabei wurden insbesondere die Möglichkeiten objekt-relationaler Datenbanksysteme (z.B. Informix, Oracle8i) genutzt und geeignet adaptiert. Objekt-relationale Datenbanksysteme bieten neben der Integration von objektorientierten Konzepten auch Möglichkeiten der Erweiterung des Datenbanksystems. D.h. die Semantik multimedialer Daten kann mittels benutzerdefinierter Datentypen (User Defined Types, UDTs) erfaßt werden sowie die dazugehörige Funktionalität über nutzerdefinierte Funktionen (User Defined Functions, UDFs) in Form von Modulen (Informix DataBlades, Oracle Cartridges) definierte werden. Insbesondere durch die Nutzung der medienspezifischen Manipulationsfunktionen (z.B. Konvertierung, Komprimierung, Skalierung von Bildern), die durch die Multimedia-Erweiterungen der DBMS bereitgestellt werden, ergeben sich eine Vielzahl von Möglichkeiten eine Vorverarbeitung der verschiedenen Medien-Typen vorzunehmen. Dies ist notwendig um die anwendungs- und gerätespezifische Transformation nicht nur auf

struktureller Ebene sondern auf den Daten selbst zu steuern und auf die Server-Seite zu verlagern.

- Zur Steuerung der Request-Bearbeitung und Anfragetransformation wurde eine XML-Framework entwickelt, das aus einer Datenbank mit einem umfangreichen Metadatenkatalog besteht sowie über XML-Schnittstellen für den Datenaustausch verfügt. Katalogisiert werden alle Informationen, die die Datenquellen und ihren Inhalt betreffen. Zusätzlich müssen in einer mobilen Umgebung die Eigenschaften und Fähigkeiten der Endgeräte (**Device Identification**) beschrieben werden, um optimale Ergebnisse bei der Erschließung großer Datenbestände auch auf mobilen Geräten zu gewährleisten. Ziel ist es, mit Hilfe des Content-Repositories die kontextabhängige Steuerung (Endgeräte- bzw. anwendungsspezifisch) der Datentransformation (Content Transformation) auf der Bakkend-Seite zu ermöglichen. D.h. daß die Datenpreprocessing-Funktionalität, die z.B. durch die Multimedia-Erweiterungen der ORDBMS bereitgestellt wird, kontextabhängig genutzt werden kann. Im Falle von Datenbank-Informationsquellen können die SQL-Anweisungen in Abhängigkeit vom Client-Request und dem Endgeräte-Kontext entsprechend generiert werden. So wird eine intelligente Daten-Projektion auf genau diejenigen Daten-Elemente möglich, die entsprechend der Endgeräte-Identifikation auf dem mobilen Gerät visualisiert werden können. Die Generierung der entsprechenden XML-Strukturen für die Weiterverarbeitung (z.B. Anwendung von XSLT-Stylesheets) kann direkt auf diese vom Datenbanksystem zurückgelieferten SQL-Ergebnisse aufsetzen. Gleichzeitig dient das Content-Repository zur übergreifenden Suche über alle Datenbestände. Durch die Auswertung der Metadaten kann jederzeit ermittelt werden, wo sich bestimmte Daten in welcher Form befinden. Im Rahmen dieser Arbeit wurde als Metadatenformat die **R**esouce **D**escription **F**ramework-Spezifikation **(RDF)** des W3-Consortiums verwendet.
- XSL stellt einen allgemeinen Mechanismus für die Transformation von XML-Bäumen bereit und wird momentan als Standard vom W3C-Consortium entwickelt. XSL hat zwei Bestandteile. Eine allgemeine Sprache zur Transformation von Bäumen (XSLT) und eine Sprache zur Notation von in Flächen dargestellten Bäumen, die aus Formated 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 sol-

ches 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 unterschiedliche Medien und Endgeräte. Unter Nutzung dieses Mechanismus wurden im Rahmen dieser Diplomarbeit vordefinierte XSLT-Stylesheets (Templates) zur Verfügung gestellt, die die Publikation auf unterschiedliche Endgeräte und Ausgabeformate im beschriebenen Anwendungskontext ermöglichen. Eine Zielstellung für weitergehende Arbeiten wäre darüber hinaus intelligente Funktionen zu implementieren, um die entsprechende Endgeräte-Visualisierung teilweise generieren zu können und die zugehörigen Benutzerfunktionen effektiv zu unterstützen. Für die Anwendung in mobilen Umgebungen ist der Leistungsumfang und das Konzept von XML/XSLT jedoch nicht ausreichend. Mobile Geräte sind hinsichtlich der Größe der Dokumente und der darstellbaren Datentypen eingeschränkt. Eine Selektion der darstellbaren Datentypen aus einem XML-Dokument reicht zur Präsentation des Inhaltes oft nicht aus. Darum war es notwendig, das System so intelligent zu gestalten, daß neben der Selektion der Inhalte auch eine automatische Transformation auf den entsprechenden Datentypen erfolgen konnte.

Ein wesentliches Kernstück des vorgestellten Systemlösung ist das Database-to-XML-Servlet. Hier wurde ein leistungsfähiger Funktionsbaustein entwickelt, mit dem die relationalen Datenstrukturen neuer bzw. bestehender Datenbanken effektiv in XML-Dokumentstrukturen umgewandelt werden können. Der Ansatz, der vorgestellt wurde, ermöglicht es Ergebnisse von SQL-Anfragen direkt an den XML-Applikations-Server (Retrieval-Server) weiterzuleiten. Das Database-to-XML-Servlet setzt weitgehend auf offene Standards auf wie z.B. JDBC, ODBC für den Datenbankzugriff bzw. DOM (Document Object Model) für den Zugriff auf generierten XML-Dokumente. Durch den Verzicht auf die Nutzung verfügbarer kommerzieller und proprietärer XML-Erweiterungen, die durch die Datenbanksystem-Hersteller (z.B. Oracle XML Developer's Kit, XDK) angeboten werden, bleibt die Offenheit und die Systemunabhängigkeit des Ansatzes erhalten.

Als eine praktische, anwendungsorientierte Weiterentwicklung dieser Arbeit wäre die Einbeziehung von Daten-Integrations- und Föderations-Fragestellungen in die Systemlösung denkbar. In einer zusätzlichen Schicht der Content-Integration könnten die benötigten Informationen für das jeweilige Anwendungsszenario aus verschiedensten Datenquellen ermittelt (unter Einbeziehung des Content Repositories) und in eine definierte XML-Struktur

transformiert werden. Dies gilt insbesondere für Inhalte, die sich nur in definierten Zeitintervallen ändern. Die persistente Speicherung der Daten in eine zentrale XML-Datenbank könnte zugleich einen Vorteil für die Performance mit sich bringen, da die Daten zum Anfragezeitpunkt bereits in vorverarbeiteter Form verfügbar wären. Durch die einheitliche Repräsentationsform der gesamten Daten in XML erspart man sich so in den Folgeschritten die aufwendige Verarbeitung verschiedener Datenformate. Am Ende der Content Integration ist somit der Zugriff auf die benötigte Information in einem einheitlichen Datenformat möglich. Dazu können entsprechende Abfragesprachen (SQL, XQL, OQL etc.) verwendet werden.

Der Forderung nach Integration heterogener Endgeräte in die Bedienprozesse von Informationssystemen wird zur Zeit durch spezialisierte Entwicklungen pro Endgeräteklasse entsprochen. Die Entwicklung solcher Systeme ist entsprechend aufwendig und unwirtschaftlich. Aufsetzend auf die im Rahmen dieser Diplomarbeit erarbeiteten Daten-Zugriffskomponenten könnten neuartige Werkzeuge zur endgeräteunabhängigen Modellierung der Dialogkomponenten für Informationssysteme mit anschließender regelbasierter Abbildung auf die speziellen Möglichkeiten verschiedener Endgeräteklassen dieses Problem lösen. Diese Werkzeuge könnten sowohl beim Neuentwurf als auch im Reverse Engineering bestehender Systeme eingesetzt werden und in Form von Prototypen der Dialogkomponenten mit angemessenem Aufwand implementierbare Spezifikationen konkreter Benutzeroberflächen liefern.

Anhang A Metadatenstruktur

A.1 RDF-Graph für den Application Context

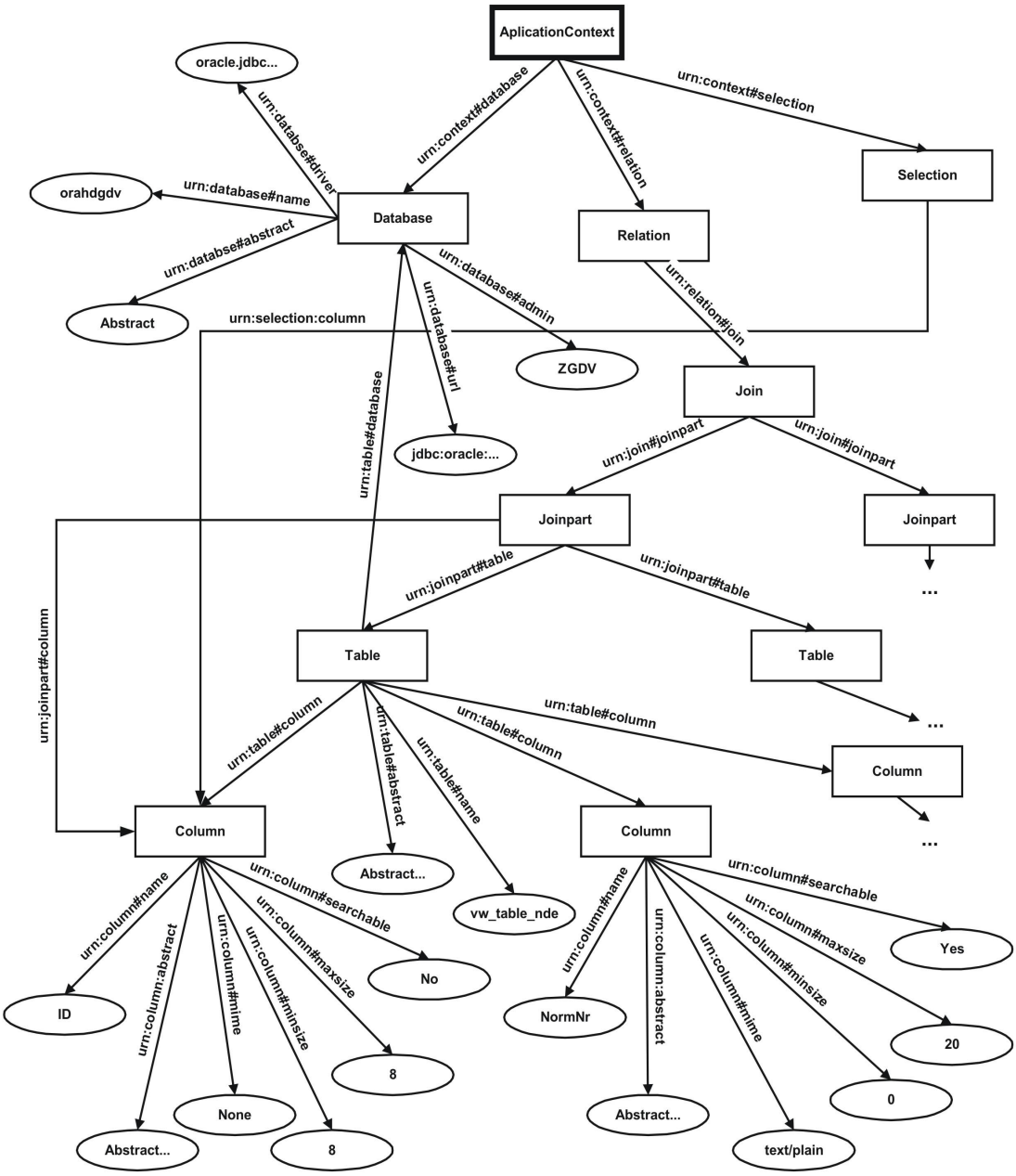

Abbildung A.1 RDF-Graph (Auszug)

Zusätzliche Namespaces (Auszug)

urn:context

Property	Beschreibung
#relation	Resource, die eine Relation zwischen Tabelle und Daten beschreibt
#selection	Resource, die die zu selektierenden Attribute beschreibt
#database	Resource, die Angaben über die Datenbank enthält
#name	Eindeutiger Name des Kontextes

urn:relation

Property	Beschreibung
#join	Join oder Table Resource

urn:join

Property	Beschreibung
#joinpart	Resourcen, die die Joinpartner beschreiben

urn:joinpart

Property	Beschreibung
#table	Resource, mit der gejoint wird (Join oder Table)
#column	Joinspalte

urn:table

Property	Beschreibung
#name	Name der tabelle
#abstract	Beschreibung des Inhaltes
#column	Spalten der Tabelle

urn:column

Property	Beschreibung
#name	Name der Spalte
#table	Tabelle, in der die Spalte definiert ist
#abstract	Beschreibung der Daten
#type	Datentyp der Spalte
#mime	Mime-Type der Attributwerte
#minsize	minimale auftretende Größe eines Attributewertes (regelm. Update)
#maxsize	maximale auftretende Größe eines Attributewertes (regelm. Update)
#searchable	gibt an, ob dieses Attribut durchsucht werden soll (YES/NO)

urn:database

Property	Beschreibung
#name	Name der Datenbank
#abstract	Beschreibung
#admin	Administrator der Datenbank
#url	JDBC URL der Datenbank
#driver	JDBC Treiber der Datenbank

Anhang B Content-Mediator

B.1 Application Context-DTD

```
<!-- Jeweils Daten zu einem Anwendungskontext  -->
<!-- Tabellenbeziehungen und Ergebnisattribute -->
<!ELEMENT ApplicationContext (Relation, Selection,Database)>

<!-- Ergebnistabelle ist entweder eine Datenbanktabelle -->
<!-- oder ein Join aus mehreren Datenbanktabellen       -->
<!ELEMENT Relation ((Table,ValueJoin?)|Join)>

<!-- Ein Join erfolgt zwischwen zwei Tabellen,      -->
<!-- die auch Ergebnis eines vorhergehenden         -->
<!-- Joins sein können                       -->

<!ELEMENT Join (JoinPart, JoinPart)>
<!-- Ein Joinpartner ist eine Tabelle oder ein Join -->
<!ELEMENT JoinPart ((Table|Join), Column, ValueJoin?)>
<!ELEMENT ValueJoin (Column)>
<!ATTLIST ValueJoin
       Value CDATA #REQUIRED>

<!-- Die zu selektierenden Attribute -->
<!ELEMENT Selection (Column+)>

<!-- ---------------------- -->
<!-- Metadaten einer Tabelle -->
<!-- ---------------------- -->
<!ELEMENT Table (Column+)>
<!ATTLIST Table
       Name     CDATA #REQUIRED
       Abstract CDATA #REQUIRED
       Id       ID    #REQUIRED>
<!-- Attribute einer Tabelle -->
<!ELEMENT Column EMPTY>

<!ATTLIST Column
       Name       CDATA #REQUIRED
       TableId    IDREF #REQUIRED
       Abstract   CDATA #REQUIRED
       Type       CDATA #REQUIRED
       MimeType   CDATA #REQUIRED
       MinSize    CDATA #REQUIRED
       MaxSize    CDATA #REQUIRED
       Searchable CDATA #REQUIRED>
```

```
<!-- Die Datenbank -->
<!ELEMENT Database EMPTY>
<!ATTLIST Database
       Name     CDATA #REQUIRED
       Abstract CDATA #REQUIRED
       Admin    CDATA #REQUIRED
       URL      CDATA #REQUIRED
       Driver   CDATA #REQUIRED>
```

B.2 SQL Anweisung-DTD

```
<!ELEMENT ResultSQL (Select, From, Where)>
<!ELEMENT Select (Column+)>
<!ELEMENT From (Table+)>
<!ELEMENT Where (Equal|Less|More|Like|And|Or)>
<!ELEMENT And ((Equal|Less|More|Like|And|Or), (Equal|Less|More|Like|And|Or)+)>
<!ELEMENT Or ((Equal|Less|More|Like|And|Or), (Equal|Less|More|Like|And|Or)+)>
<!ELEMENT Equal (Column, (Column|Value))>
<!ELEMENT Less (Column, (Column|Value))>
<!ELEMENT More (Column, (Column|Value))>
<!ELEMENT Like (Column, Value)>
<!ELEMENT Value #PCDATA>

<!-- ---------------------- -->
<!-- Metadaten einer Tabelle -->
<!-- ---------------------- -->
<!ELEMENT Table (Column+)>
<!ATTLIST Table
       Name     CDATA #REQUIRED
       Abstract CDATA #REQUIRED
       Id       ID    #REQUIRED
       NewName  CDATA #IMPLIED>

<!-- Attribute einer Tabelle -->
<!ELEMENT Column EMPTY>
<!ATTLIST Column
       Name       CDATA #REQUIRED
       TableId    IDREF #REQUIRED
       Abstract   CDATA #REQUIRED
       Type       CDATA #REQUIRED
       MimeType   CDATA #REQUIRED
       MinSize    CDATA #REQUIRED
       MaxSize    CDATA #REQUIRED
       Searchable CDATA #REQUIRED>
```

B.3 Client Request (Testimplementierung)

```
<!ENTITY % devices "(palm | handy | pc)">
<!ENTITY % SrvCommands "(shutDown | cacheInfo )">

<!ENTITY % Rubrics "(all | PV | TL | VW | PDF | SAE | AMS )">
<!ENTITY % Languages "(german | english)">
```

```
<!ENTITY % MaxResults "(all | 10 | 20 | 30 | 40 | 60 | 100 )">

<!ELEMENT REQUEST (SRVCOMMAND | DBQUERY)>
<!ATTLIST REQUEST
          DEVICE %devices; #REQUIRED>

<!ELEMENT SRVCOMMAND EMPTY>
<!ATTLIST SRVCOMMAND
          COMMAND %SrvCommands; #REQUIRED>

<!ELEMENT DBQUERY (SELECT | FETCH?)>
<!ATTLIST DBQUERY
           User CDATA #REQUIRED
           Password CDATA #REQUIRED>

<!ELEMENT SELECT EMPTY>
<!ATTLIST SELECT
          RUBRIC %Rubrics; "all"
          LANGUAGE %Languages; #IMPLIED
          RESULTS %MaxResults; "all">

<!ELEMENT FETCH EMPTY>
<!ATTLIST FETCH
          RESOURCE CDATA #REQUIRED
          ID CDATA #REQUIRED>
```

Abbildungsverzeichnis

Abbildung 2.1 Die geschichtliche Entwicklung von XML *11*
Abbildung 2.2 Baumstruktur *19*
Abbildung 2.3 Blockschema *19*
Abbildung 3.1 Wireless Application Environment Model *28*
Abbildung 3.2 Das WAP-Schichtenmodell *29*
Abbildung 4.1 XML Extender Überblick *33*
Abbildung 4.2 Informix DataBlade-Technologie *34*
Abbildung 4.3 Aufbau und Funktionsweise des Web-DataBlades *35*
Abbildung 4.4 Datenimport über XSL/ XSLT *37*
Abbildung 5.1 Prototyp für „DigTeD“ *44*
Abbildung 5.2 Java-Pakete und Ihre Zielmärkte (Quelle: SUN) *45*
Abbildung 5.3 im Rahmen von iViP entwickelte Client-Server Lösung *46*
Abbildung 5.4 XML Framework Architektur *47*
Abbildung 5.5 Anfragebearbeitung und Transformation *49*
Abbildung 5.6 Requestaufbau auf dem Client *49*
Abbildung 5.7 Anwendungs-Kontext *50*
Abbildung 5.8 Palm-Lösung *52*
Abbildung 5.9 WAP-Lösung *53*
Abbildung 7.1 RDF-Graph (Auszug) *58*

Literaturverzeichnis

[AbBS00] Abiteboul, S.; Bunemann, P., Suciu, D. – data on the Web – *From Relations to Semistructered Data and XML*, Morgan-Kaufmann, 2000

[ABH97] P.M.G. Apers,H.M. Blanken, M.A.W. Houtsma (Eds.). Multimedia Databases in Perspective. Springer , 1997

[AbKl97] K. Aberer, W. Klass. Multimedia and its Impact on Database System Architectures. In *Apers, P.M.G, Blanken, H.M., Houtsma, M.A.W (Eds.). Multimedia Databases in Perspective*. Springer , 1997

[ANSI95] American National Standards Institute, "*Information Retrieval Application Service Definition and Protocol Specification for Open Systems Interconnection*", ANSI Z39.50--1995, 1995, http://lcweb.loc.gov/z3950/agency.

[AQM+97] S. Abiteboul, D. Quass, J. McHugh, Widom, J., Wiener, J.L."*The Lorel Query Language for semistructured data*", International Journal on Digital Libraries, 1(1): 68--88 (1997).

[ATN98] K. Aberer, H. Thimm,E.J. Neuhold. Multimedia Database Systems. In: *Handbook of Multimedia Computing*, Borko Furth (Editor), CRC Press, 1998

[BaRi99] R.A. Baeza-Yates, B. Ribiero-Neto, Modern Information Retrieval. *Addison Wesley*, Harlow, England, 1999

[BDH+96] P. Buneman, S. Davidson, G. Hildebrand, D. Suciu. A query language and optimization techniques for unstructured data. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Tucson, AZ, USA, June 4--6, (1996). Published as ACM SIGMOD Record 25(2): 505--516 (1996).

[BeMi98] H. Behme, S. Mintert. XML in der Praxis – Professionelles Web-Publishing mit der Extensible Markup Language. Addison-Wesley, 1998

[BoKL96] S. Boll, W. Klas, M. Löhr. Integrated Database Services for Multimedia Presentations. In S.M. Chung (editor), Multimedia Information Storage and Management. Kluwer Academic Publishers, Dordrecht 1996

[BöLu97] J. Bönigk, U. v. Lukas. Interactive Exchange of structured multimedia Data with mobile hosts. In Global Communication Interactive, published by Hanson Cooke Ltd, 1997

[BöFl98] J. Bönigk, G. Flach. System Architecture and Strategies for the Exchange of structured multimedia Data in the Context of mobile Visualization. In *Proc. 5th Intl. Workshop on Mobile Multimedia Communication (MoMuc)*, October 1998

[Brad00] N. Bradley. The XML companion, 2nd Edition. Addison-Wesley, 2000

[CBS94] R.H.L. Chiang, T. M. Barron, V.C. Storey. Reverse Engeneering of Relational Databases: Extraction of an EER model from a Relational Database, *Data & Knowledge Engeneering* 12 (1994), 107-142

[CGF00] T. Courvoisier, N. Günther, G. Flach. Business-MV - Internet-based Services for Database Federation, Communication and Cooperation, *Computer Graphics topics*, 2000

[ChHa98] D. Chang, D. Harkey. Client slash server data access with Java and XML. John Wiley and Sons, New York, NY, USA; London, UK; Sydney, Australia, 1998.

[Chia95] R.H.L. Chiang. A knowledge based system for performing reverse engeneering of relational databases, *Decision Support Syst*., 1995

[CJS99] S. Cluet, S. Jacmin, J. Simeon. *The New YATL: Design and Specification*, Technical Report, INRIA, 1999.

[CoFl00] T. Courvoisier, G. Flach. DaS – Database to XML Servlet In *Tagungsband 12. GI-Workshop "Grundlagen von Datenbanken*", Plön, 2000

[Cour00] T. Courvoisier. Nutzung relationaler Datenquellen in der XML-Technologie – Überblick über existierende Ansätze und Tools, In *Tagungsband „3. Darmstädter XML-Kongress*“, Darmstadt, 2000

[Ditt00] Dittberner, Christoph. Konzeption und Umsetzung eines interaktiven, multimedialen Produktinformationssystems. Studienarbeit, Universität Rostock FB Informatik, 2000

[DFF+99] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, D. Suciu,. XML-QL: a query language for XML. In Proceedings of the International World Wide Web Conference (WWW), 1999.

[FHK+98] G. Flach, S. Haßinger, B. Kehrer, U. v.Lukas. Aspects of iViP – Integrated Virtual Product Realization, In. *COMPUTER GRAPHIK topics*, 6/98 Vol. 98

[Flac00] G. Flach. Objekt-Relationale Datenbank-Technologie: Einsatz und Nutzen für multimediale Information-Retrieval-Anwendungssysteme - Use Case: Datenbankbasierte Volltextsuche (Digitales Archiv); In Tagungsband: 12. GI-Workshop Grundlagen von Datenbanken, Ploen, 2000

[Fla99a] G. Flach. Die Nutzung eines ORDBMS für medienspezifische Anfragen im Rahmen einer mobilen Infrastruktur In Tagungsband: 11. GI-Workshop "Grundlagen von Datenbanken", Luisenthal, 1999

[Fla99b] G. Flach. Objekt-relationale Datenbank-Technologie: Einsatz und Nutzen für multimediale Information-Retrieval-Anwendungssysteme, In *Tagungsband: 2. IuK-Tage Mecklenburg-Vorpommern*, Rostock, 1999

[FlGü00] G. Flach, N. Günther. Architecture for the Interaction and Access on Multimedia Database Systems in the Context of Mobile Environments, In Proc. of the 4th Int. Database Engineering and Application Symposium (IDEAS), Yokohama, Japan, 2000

[FlLo01] M. Flehming, H. Loeser. *Ansätze der Nutzung von Erweiterbarkeitsmechanismen zur Anwendungsintegration in ORDBMS – Eine qualitative und quantitative Evaluierung*, , Proc. of Datenbanksysteme in Büro, technik und Wissenschaft (BTW), Oldenburg, 2001

[FloKo99] D. Florescu, D. Kossmann. Storing and Querying XML Data using an RDBMS, *Bulletin of the Technical Commitee on Data Engineering*, 22(3):27-34, 1999

[GBS91] G.H. Gonnet, R.A. Baeza-Yates, T. Snider.. *Lexicological indices for text: Inverted files vs. PAT trees*, Technical Report OED-91-01, University of Waterloo Centre for the New Oxford English Dictionary and Text Research, 1991.

[Gibb98] W.W. Gibbs, Cyber view: The Web learns to read. Scientific American, 278(6):40-40, June 1998. Discusses HTML, MathML, MusicML, ChemML, and XML.

[GoPr98] C.F. Goldfarb, P. Prescod. The XML Handbook. Prentice-Hall, 1998.

[HaMe01] E. R. Harold, W. S. Means. XML in a Nutshell. O'Reilly & Associates, 2001

[HeLu96] A. Heuer, A. Lubinski. Mobile Information Access - Challenges and possible Solutions. *Workshop "Information Visualization and Mobile Computing" (IMC '96)*, Rostock, February 1996

[HePr99] A. Heuer, D. Priebe. IRQL- yet another language for querying semistructured data? *Preprint* CS-01-99, Universität Rostock, Fachbereich Informatik, 1999

[HKK+95] A. Heuer, B. Kehrer, T. Kirste, H. Schumann, B. Urban. Concepts for Mobile Information Visualization - The MOVI-Project. *Proc. Eurographics Workshop on Scientific Visualization 1995*, May 1995

[IIFF01] Informix Internet Foundation Framework. Product Announcements. Online Dokument, Informix Corp., http://www.informix.com/informix/products/pa/iif2000.htm

[IIFF02] Informix Internet Foundation Framework. Informix and XML: Smart Data, Product Roadmap. Online Dokument, Informix Corp., http://www.informix.com/informix/products/xml/roadmap.htm

[IIFF03] Informix Internet Foundation Framework. Informix and XML: Smart Data, What is XML. Online Dokument, Informix Corp., http://www.informix.com/informix/products/xml/xml.htm

[IIFF04] Informix Internet Foundation Framework. Informix and XML: Smart Data, Informix XML Strategy, An Overview. Online Dokument, Informix Corp., http://www.informix.com/xml/strategy.htm

[ISO96] International Organization for Standardization, "*Information technology---Processing languages---Document Style Semantics and Specification Language (DSSSL)*". ISO/IEC 10179:1996, 1996.

[KlMe00] M. Klettke, H. Meyer. Managing XML documents in object-relational databases. *Submitted*, Universität Rostock, Fachbereich Informatik, 1999

[Lund99] L. Lunde. CJKV Information Processing: Chinese, Japanese, Korean & Vietnamese Computing. O'Reilly & Associates, 1999.

[LBB+00] W. Lindner, H. Berthold, F. Binkowski, A. Heuer, K. Meyer-Wegener. Enabling hypermedia videos in multimedia database systems coupled with realtime media servers. In: Proc. IDEAS‘2000, Yokohama, Sept. 2000

[Nico98] G.T. Nicol. Multilingual, and parallel aligned texts: Putting XML and XLL to work. In Unicode Consortium, editor, Asia, Software + the Internet: Going Global with Unicode: The Twelfth International Unicode slash ISO10646 Conference (IUC12) April 8-10, 1998, Tokyo, Japan, 1998

[NMZ98] M.L.Nelson, K. Malym, M. Zubiar. Interoperable heterogeneous digital libraries. Technical report, Old Dominion University, TR_98_07, September 1998.

[Ora801] Oracle8i interMedia: *Platform Service for Internet Media and Document Content*, http://technet.oracle.com/products/intermedia

[Orac00] Oracle Corp., *XML Developers Kit*, http://www.oracle.com/xml/

[Orac99] Oracle8i *inter* Media Text Release 8.1.5 Reference, Februar 1999

[Pors00] B. Porst. Untersuchungen zu Datentyperweiterungen für XML-Dokumente und ihre Anfragemethoden am Beispiel von DB2 und Informix. Diplomarbeit, Universität Rostock, Fachbereich Informatik, 1999

[RCF] J. Robie, D. Chamberlin, D. Florescu. Quilt: an XML Query Language. (Unpublished.) http://www.almaden.ibm.com/cs/people/chamberlin/quilt.html

[RLS98] J. Robie, J. Lapp, D. Schach. XML Query Language (XQL). In QL'98---The Query Languages Workshop, Boston, MA, USA, 1998. http://www.w3.org/TandS/QL/QL98/

[SDT+97] R. Sacks-Davis, T. Dao, J. Thom, J. Zobel. Indexing Documents for Queries on Structure, Content and Attributes. International Conference on Digital Media Information Bases (DMIB), Nara, Japan, (1997), 236--245.

[SIM99a] SIM 3.0 Administrator Manual Series, Content Manager User's Guide, *RMIT Multimedia Database Systems*, RMIT University, GPO Box 2476V, Melbourne, Victoria, Australia (1999), http://www.simdb.com

[SIM99b] SIM 3.0 Administrator Manual Series, *Content Manager Reference Manual*, RMIT Multimedia Database Systems, RMIT University, Melbourne, Victoria, Australia (1999), http://www.simdb.com

[SIM99c] SIM 3.0 Administrator Manual Series, ACE Programmer's Guide, *RMIT Multimedia Database Systems*, RMIT University, Melbourne, Victoria, Australia (1999), http://www.simdb.com

[SiTW01] S. Sizov, A. Theobald, G. Weikum. Ähnlichkeitssuche auf XML-Daten. Proc. of Datenbanksysteme in Büro, Technik und Wissenschaft (BTW), Oldenburg, 2001

[StDe01] U. Störl, U. Deppisch. XML-basierte Content-Analyse: Vision und Realität , Proc. of Datenbanksysteme in Büro, Technik und Wissenschaft (BTW), Oldenburg, 2001

[StBr99] M. Stonebraker, P. Brown. Object-Relational DBMSs- Tracking the Next Great Wave, The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann Publishers, San Francisco, 1999

[STH+99] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, J. Naughton. Relational Databases for Querying XML Documents: Limitations and Opportunities. Proc. of the Very Large Data Base (VLDB), 1999

[Ston96] M. Stonebraker. Object-Relational DBMSs – The Next Great Wave, Morgan Kaufmann, 1996

[Subr98] V.S. Subrahmanian. Principles of Multimedia Database Systems, Morgan Kaufmann Publishers Inc., 1998

[TBS+99] R. Thibadeau, J. Balderas, A. Snyder, J. Nestor. E-commerce catalog construction: An experiment with programmable XML for dynamic documents. Technical Report february99-thibadeau, D-Lib Magazine, February 1999.

[Timm99] J. Timm. Speicherung von XML-Dateien in Objekt-relationalen Datenbanken, *Diplomarbeit*, Universität Rostock, Fachbereich Informatik, 1999

[Tura99] V. Turau. Making Legacy Data Accessible for XML Applications, FH Wiesbaden, Paper ID: 331, 1999

[Wate01] W. Waterfeld. Realisierungsaspekte eines XML Datenbanksystems, Proc. of Datenbanksysteme in Büro, technik und Wissenschaft (BTW), Oldenburg, 2001

[Wor98a] World Wide Web Consortium. Extensible Markup Language (XML), Version 1.0, *1998,* http://www.w3.org/TR/1998/REC-xml-19980210

[Wor98b] World Wide Web Consortium. XML-QL: A Query Language for XML. http://www.w3.org/TR/NOTE-xml-ql/

[Wor98c] World Wide Web Consortium. XML Query Language (XQL), http://www.w3.org/TandS/QL/QL98/pp/xql.html

[Wor99a] World Wide Web Consortium. XML Schema Part 1: Structures / XML Schema Part 2: Datatypes, http://www.w3.org/TR/xmlschema-2/

[Wor99b] World Wide Web Consortium. XSL Transformations (XSLT) Specification, http://www.w3.org/TR/1999/WD-xslt-19990421.html

[Wor99c] World Wide Web Consortium . Document Object Model, http://www.w3.org/DOM, Online article.

[Wor99d] W3C. *Platform for internet content selection.* http://www.w3.org/PICS/, June 1999. Online article.

[Wor99e] World Wide Web Consortium, Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommandation, 1999, http://www.w3.org/TR/REC-rdf-syntax . Online article.

[Worl97] World Wide Web Consortium. XML representation of a relational databases (by B. Bos), http://www.w3.org/XML/RDB.html

Selbständigkeitserklärung

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig und nur unter Benutzung der angegebenen Quellen angefertigt habe.

Christoph Dittberner

Rostock, 19.3. 2001

Thesen

I. Die entwickelte Lösung zur Anfragebearbeitung und Daten-Transformation basiert auf einem zweistufigen Ansatz bei der die Daten-Transformation (Content-Mediation) sowohl auf struktureller Ebene (XML/XSLT) als auch auf den Datenelementen selbst stattfindet. Die Funktionsbausteine werden dabei direkt durch die Device- und Kontext-Informationen des RDF-Content-Repositories gesteuert. Dieser neuartige Ansatz hat sich als grundsätzlich tragfähig erwiesen.

II. Existierende objekt-relationale Datenbanksysteme (Oracle8i, Informix) bieten sehr flexible Möglichkeiten medienspezifisch Preprocessing-Funktionalität direkt als serverseitige Manipulationsfunktionalität im DBMS ablaufen zu lassen. In den verwendeten ORDBMS wird diese Funktionalität in Form vorgefertigter Module (Informix DataBlades, Oracle Cartridges) integriert. Dabei handelt es sich um eine Sammlung von neuen Datentypen, zusammen mit den dazugehörigen Funktionen, Integritätsbedingungen, Indexmechanismen etc.. Die interne Struktur der neuen Datentypen bleibt nach außen völlig verborgen, der Zugriff ist nur über die entsprechenden Funktionen möglich. Es wird also das objektorientierte Prinzip der Kapselung ausgenutzt, indem eine Schnittstelle zur Verfügung gestellt wird, welche ausschließlich auf benutzerdefinierten Funktionen basiert.

III. Als problematisch hat sich allerdings die Umsetzung eigener Multimedia-Erweiterungen bei dem ORDBMS Oracle8i aufgrund nur eingeschränkt vorhandener API-Schnittstelle erwiesen. Durch die offenere Systemarchitektur des Informix-Servers (DataBlade-API) stehen hier wesentlich umfangreichere Realisierungsmöglichkeiten zur Verfügung.

IV. Im Falle von Datenbank-Informationsquellen können die SQL-Anweisungen in Abhängigkeit vom Client-Request und dem Endgeräte-Kontext entsprechend der aufgezeigten und evaluierten Lösung generiert werden. So wird eine intelligente Daten-Projektion auf genau diejenigen Daten-Elemente möglich, die entsprechend der Endgeräte-Identifikation auf dem mobilen Gerät visualisiert werden können. Die Generierung der XML-Strukturen für die Weiterverarbeitung (z.B. Anwendung von XSLT-Stylesheets) kann so direkt auf diese vom Datenbanksystem zurückgelieferten SQL-Ergebnisse aufsetzen.

V. Die unterschiedlichen Fähigkeiten und Zweckmäßigkeiten solcher Medien erfordert eine zusätzliche Transformation von Inhalten der Informationen. Das kann nur durch die Erfassung semantischer Zusammenhänge in den Datenmengen geschehen was den Einsatz geeigneter Metadatensysteme erfordert.

VI. Das Resource Description Framework bietet ein Instrument zur maschinenlesbaren Formulierung von Metadaten, die dank des offenen W3C-Standards geeignet sind, semantische Informationen über unstrukturierte und semistrukturierte Daten zu speichern. Diese Informationen können nicht nur für die Verwaltung sondern auch zur Verarbeitung der Information genutzt werden.

VII. Der offene RDF-Standard ermöglicht den Austausch von Metadaten zwischen unterschiedlichen Systemen, was zusammen mit der metadatenbasierten Klassifizierung und Indizierung unstrukturierter und semistrukturierter Daten, das Auffinden von Informationen in komplexen heterogenen Systemen ermöglicht.

VIII. Die Anbindung relationaler Datenquellen an XML-Infrastrukturen schafft die notwendige Flexibilität in der Datenverarbeitung unter Nutzung von RDBMS als persistente Ebene. Die DaS-Komponente des vorgestellten Systems eignet sich, durch die Nutzung offener Standards, JDBC-kompatible Datenbanken in java-basierte XML-Applikationen einzubinden. Die Nutzung der standardisierten DOM-Schnittstelle ermöglicht die Verarbeitung von Anfrageergebnissen durch beliebige XML-Parser.

IX. Die Portabilität der Client-Anwendung ist durch die Verwendung von Java als Programmiersprache nur bedingt gewährleistet. Viele Probleme traten vor allen Dingen durch die Inkompatibilität bzw. das stark eingeschränkte Klassenmodell der KVM von SUN bezüglich des JDK 2.0 bei der Implementierung des Clients auf einem Palm III-Gerät auf..