Institut für Informatik
der Technischen Universität München

Integriertes Management großer Web-Sites auf der Basis datenbankbasierter Modellierungskonzepte

Ulrike Sommer

Institut für Informatik
der Technischen Universität München

Integriertes Management großer Web-Sites auf der Basis datenbankbasierter Modellierungskonzepte

Ulrike Sommer

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. B. Radig

Prüfer der Dissertation:

1. Univ.-Prof. R. Bayer, Ph.D.

2. Univ.-Prof. Dr. A. Brüggemann-Klein

Die Dissertation wurde am 21.01.2000 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 28.08.2000 angenommen.

Vorwort

Die vorliegende Arbeit ist während meiner Tätigkeit als wissenschaftliche Angestellte am Bayerischen Forschungszentrum für Wissensbasierte Systeme (FORWISS) in München entstanden. Vielfältige Anregungen zu der Arbeit kamen dabei aus den FORWISS-Projekten M-Line, TTM-Line und abayfor-online, an denen ich mitgewirkt habe. Bei allen Personen, die zum Erfolg meiner Arbeit beigetragen haben, möchte ich mich an dieser Stelle herzlich bedanken.

Als erstes gilt mein Dank meinem Doktorvater, Herrn Prof. Rudolf Bayer, Ph.D., für die umfassende wissenschaftliche Betreuung und die konstruktiven Diskussionen. Frau Prof. Dr. Anne Brüggemann-Klein danke ich für die Übernahme des Zweitgutachtens.

Viele Kolleginnen und Kollegen haben durch fachliche Diskussionen zum Gelingen dieser Arbeit beigetragen. Insbesondere danken möchte ich Herrn Dr. H. Haddouti, Herrn Prof. G. Specht, Herrn P. Zoller und Herrn Dr. N. Widmann für ihre wertvolle Unterstützung.

Mein großer Dank gilt meinem Mann Milan, der mir über den langen Zeitraum immer wieder Kraft zum Durchhalten gegeben hat. Besonders danken möchte ich ebenfalls meinen Eltern, die mich in meinem beruflichen Werdegang immer bestätigt haben und in jeder Weise förderten.

München, im September 2000

Ulrike Sommer

Kurzfassung

Der Internet-Mehrwertdienst World Wide Web hat in den letzten Jahren zunehmend an Bedeutung gewonnen. Die Einfachheit des zugrunde liegenden Hypertext-Modells und die mangelnde Trennung zwischen logischer und physischer Ebene im Web führen bei der Verwaltung komplexer Anwendungen und großer Informationsbestände zu verschiedenen Problemen. Um die riesigen Informationsmengen im Web effizienter verwalten zu können, werden deshalb zunehmend Datenbanken eingesetzt.

In dieser Arbeit werden Verbesserungen zu Entwurf, Implementierung und Verwaltung großer Web-Anwendungen auf der Basis der entwickelten Modellierungsmethode HDBM (Hypermedia Database Methodology) beschrieben. Hierzu wird ein Meta-Schema vorgestellt, das die Verwaltung von Struktur, Layout und Navigation der Web-Anwendung in einer relationalen Datenbank gestattet. Der Einsatz von Meta-Daten gewährleistet die einfache Erstellung, Wartung und Skalierbarkeit von dynamisch generierten Web-Sites auf der Basis einer Datenbank. Realisiert wurden die Konzepte im Rahmen einer Implementierung mit dem relationalen Datenbanksystem Oracle.

Inhaltsverzeichnis

Kapitel 1

Einführung

Die rasante Entwicklung des World Wide Web gewährleistet durch den Einsatz plattformunabhängiger Standards wie HTML, HTTP oder TCP/IP zwar die weite Verbreitung von Informationen, unterstützt aber nicht die sichere und zuverlässige Verwaltung großer Datenmengen. Anders als im Datenbankbereich existieren keine anerkannten konzeptuellen und logischen Datenmodelle, die den strukturierten Entwurf und die Verwaltung großer Web-Sites unterstützen. Vor allem die mangelnde Trennung von Inhalt und Präsentation führt in der Praxis zu schwerer Wartbarkeit und schlechter Skalierbarkeit. Gut strukturierten und qualitativ hochwertigen Hypertext aufzubauen, spielt gerade bei dem zunehmenden Umfang von Web-Sites eine immer wichtigere Rolle und kann mit Hilfe der Datenbanktechnologie verbessert werden.

Dieses Kapitel führt kurz in die Thematik von Internet-Datenbanken ein. Im folgenden wird die Problemstellung diskutiert, die aktuelle Situation beschrieben sowie die Zielsetzung vorgestellt. Daneben werden Inhalt und Aufbau der Arbeit erläutert.

1.1 Motivation

Die Geschwindigkeit, mit der sich das Internet als neue Kommunikationstechnik durchgesetzt hat, übertrifft alle Erwartungen. Aus wirtschaftlicher Perspektive erschließt das Internet neue Märkte und eröffnet neue Chancen für Produktivität und Flexibilität, neue Servicequalität, unmittelbare Kundenbeziehung und Globalisierung. Der Online-Umsatz in Europa wird für das Jahr 2002 auf 2786,9 Milliarden Dollar geschätzt[1]. Laut dem NUA Internet-Service gibt es in Europa 47,15 Millionen und weltweit 201 Millionen Internet-Nutzer[2]. Um die riesigen Informationsmengen, die über das WWW angeboten werden, zu verwalten, werden zunehmend Datenbanken eingesetzt. Auf der einen Seite lassen sich die Verbreitung und die Bedienung vieler immer wieder anders konzipierter Datenbank-Insellösungen durch die Internet-Technologie optimieren. Auf der anderen Seite gewährleisten Datenbanksysteme den effizienten Zugriff auf große strukturierte Da-

[1] `http://www.dmmv.de/multi/zahlen.html`, Frost & Sullivan-Studie, Nov. 1998

[2] NUA Internet-Services `http://nua.ie/surveys/how_many_online/index.html`

tenbestände, mächtige Suchmöglichkeiten auf der Basis eines Datenmodells, Concurrency Control und die konsistente Verwaltung der Daten. Dabei bildet vor allem die durch das Datenbanksystem gewährleistete physische und logische Datenunabhängigkeit eine ideale Basis für Client/Server-Umgebungen, wie sie das Internet darstellt.

Aus technischer Sicht existiert eine Vielzahl unterschiedlicher Architekturvarianten, die eine performante Anbindung von Datenbanksystemen an das World Wide Web ermöglichen. Bei der gewählten Systemarchitektur sind Kriterien wie Portabilität, Plattformunabhängigkeit, Transaktionskontrolle und Sicherheit ausschlaggebend. Aus logischer Sicht läßt sich die Modellierung einer datenbankbasierten Hypertext-Anwendung in die Modellierung der Datenbank und die Modellierung der zugehörigen Hypertext-Anwendung untergliedern. Für den ersten Bereich gibt es bekannte semantische und logische Datenmodelle, wie beispielsweise das ER-Modell und das relationale Modell. Das logische Datenmodell einer Datenbank gewährleistet dabei die Unabhängigkeit von der physischen Repräsentation der Daten. Für Daten im Web gibt es eine derartige Trennung zwischen der logischen und physischen Ebene dagegen nicht. Für die Modellierung von datenbankbasierten Web-Sites bedeutet dies, daß nur die Modellierung der Anwendungsdomäne auf der Basis eines logischen Datenmodells erfolgt. Die Abbildung auf die zugehörige Hypertext-Anwendung wird dagegen üblicherweise durch plattformabhängige Programme, Skripten oder HTML-Templates außerhalb des Datenbanksystems implementiert. Dies führt in der Praxis zu verschiedenen Problemen:

- **Mangelnde Konsistenz zwischen Datenbank-Schema und Hypertext-Anwendung**
  Die häufig mit Hilfe von Skripten oder HTML-Templates implementierte Hypertext-Anwendung ist abhängig von der Datenbankstruktur. Ändert man das Schema der Datenbank, entstehen Inkonsistenzen in der Hypertext-Oberfläche, die schwer zu lokalisieren sind. Mangelnde Unterstützung von dynamischer Schemaevolution erschwert so die Wartung des Informationssystems.

- **Eingeschränkte Suchmöglichkeiten**
  Bei datenbankbasierten Informationssystemen im Web lassen sich verschiedene Techniken zur Formulierung von Datenbankanfragen unterscheiden. Während bei *vordefinierten Anfragen* Struktur und Layout des Anfrageergebnisses bereits im Vorfeld festgelegt sind, erfolgt die Ergebnispräsentation von *Ad-hoc-Anfragen* üblicherweise mit Hilfe einfacher HTML-Listen oder -Tabellen. Die Problematik bei Ad-hoc-Anfragen liegt darin, daß die Anfragen im Vorfeld nicht bekannt sind und damit die Struktur und Darstellung des Anfrageergebnisses erst zur Laufzeit erzeugt werden können.

- **Schwere Wartbarkeit und Skalierbarkeit**
  Bei der Programmierung von GUIs entfällt ca. 60 % des Programm-Codes auf die Organisation von Daten, wie z.B. Cursor-Koordinaten, Vordergrund, Hintergrund, Fonts, Fenster-Hierarchien. Die gleiche Tendenz zeichnet sich im Web ab. Durch

den steigenden Bedarf von Interaktion, Animation und Layout-Gestaltung werden immer mehr Meta-Daten (Tags) in eine HTML-Seite integriert. Die klare Trennung zwischen Daten, Layout-Informationen und Navigationsstruktur geht so verloren. Dies führt zu schwerer Wartbarkeit und Skalierbarkeit einer erstellten Hypertext-Anwendung.

Neue Anforderungen an Hypertext-Applikationen, wie beispielsweise Einfachheit, Konsistenz und Reichhaltigkeit, einfache Lesbarkeit und Wartbarkeit stellen heute neue Herausforderungen an Entwurf und logische Modellierung datenbankbasierter Hypertext-Anwendungen.

1.2 Zielsetzungen

Um die oben angeführten Probleme zu beheben, wurde ein neues Konzept zur integrierten Verwaltung großer datenbankbasierter Informationssysteme im World Wide Web im Rahmen des abayfor-online-Projektes entwickelt. Dabei sollen neben den Daten des Anwendungsbereichs auch Informationen zu Hypertext-Struktur, -Layout und -Navigation modelliert und innerhalb einer Datenbank in Form von Meta-Daten abgelegt werden. Dies erfordert einen ganzheitlichen Entwurfsprozeß und die einheitliche Umsetzung des Datenbankschemas mit den darauf abgestimmten, benutzerspezifischen Hypertext-Views. An das zu entwickelnde System werden insbesondere folgende Anforderungen gestellt:

- **Vermeidung von Update-Anomalien:** Änderungen der Datenbank-Struktur sollten sich automatisch auf die Hypertext-Anwendung übertragen. Inkonsistenzen zwischen Datenbank-Schema und Hypertext-Anwendung werden so vermieden. Wird beispielsweise eine Relation in der Datenbank gelöscht, so sollen automatisch auch die hierzu gehörenden HTML-Vorlageseiten zur dynamischen Generierung gelöscht werden.

- **Schnelles Prototyping:** Die Erstellung eines Prototypen sollte einfach und schnell möglich sein. Weiterhin sollte die Hypertext-Anwendung ohne Programmierkenntnisse mit Hilfe einer graphischen Entwicklungsumgebung auf der Basis des Datenbank-Schemas vom Anwender erstellt und gewartet werden können.

- **Ad-hoc-Anfragen an die Datenbank:** Bei der Formulierung von Ad-hoc-Anfragen an die Datenbank sollte die Hypertext-Struktur und Navigation der Web-Site bei der Ausgabe der Datenbank-Ergebnisse berücksichtigt werden.

- **Verschiedene Benutzersichten:** Die verfügbaren Daten der Hypertextanwendung sollten aus der Sicht verschiedener Anwenderklassen im Web modelliert werden können. Dies ermöglicht verschiedene Hypertext-Views auf ein und denselben Datenbestand.

- **online- und offline-Seitengenerierung:** Die Generierung von HTML-Seiten aus einer Datenbank sollte sowohl dynamisch zur Laufzeit (online) als auch im Vorfeld (offline) erfolgen können. Eine Problematik bei materialisisierten HTML-Seiten liegt in der schweren Wartung und den hierdurch bedingten Konsistenzverletzungen. Das System sollte die inkrementelle Wartung der HTML-Seiten auf der Basis des Datenbankschemas unterstützen.

- **Verwendung eines Hypertext-Modells:** Die logische Modellierung einer Web-Site sollte von der physischen Implementierung getrennt sein. Der Einsatz einer durchgängigen Modellierungsmethode, die die dynamische Generierung komplexer HTML-Seiten auf der Basis eines logischen Hypertext-Modells unterstützt, ermöglicht die einfache Skalierbarkeit, Wiederverwendbarkeit und Wartbarkeit einer Anwendung.

In dieser Arbeit werden einige Aspekte von Web-Anwendungen nicht berücksichtigt, da sie Inhalt einer anderen am FORWISS (Bayerisches Forschungszentrum für Wissensbasierte Systeme) durchgeführten Dissertation sind. Hierzu gehört die Modellierung von HTML-Formularen zur Bearbeitung vordefinierter Anfragen und zur Wartung von datenbankbasierten Web-Anwendungen auf der Basis eines erweiterten Meta-Schemas. Ebenfalls nicht Inhalt dieser Arbeit ist die Befüllung des vorgestellten Meta-Schemas mit Hilfe von CASE-Tools.

1.3 Aufbau der Arbeit

Die genannten Zielsetzungen führten zur Entwicklung der Hypermedia Database Methodology (HDBM), die im Rahmen dieser Arbeit vorgestellt wird. Das konzeptuelle Design ist unterteilt in konzeptuelles Datenbank-Design und konzeptuelles Hypermedia-Design. Als semantische Datenmodelle kommen das ER-Modell (Entity-Relationship-Modell) [Che76] und das erweiterte RMDM (Relationship Management Data Model) [IKK97] zum Einsatz. Den Kern der HDBM bildet ein Meta-Schema, das auf der logischen Strukturierung von Hypertext-Struktur, Layout und Navigation in einem relationalen Datenmodell basiert. Die Generierung der dynamischen HTML-Seiten erfolgt mit Hilfe eines im Rahmen dieser Arbeit entwickelten Tools, das die Meta-Daten zur Laufzeit auswertet und HTML-Seiten generiert. Auf der Basis des Meta-Schemas werden Ansätze zur Ad-hoc-Anfragebearbeitung und Wartung von materialisierten HTML-Seiten vorgestellt. Als praktisches Beispiel für die gesamte Arbeit dient das im Rahmen des abayfor-Projektes entworfene Datenbank- und Hypertextschema. Die vorgestellten Ansätze wurden prototypisch auf der Basis des Oracle-Datenbanksystems implementiert.

Der Schwerpunkt der vorliegenden Arbeit liegt auf der Entwicklung einer geeigneten Methodik zum Entwurf datenbankbasierter Hypertext-Anwendungen auf der Basis des HDBM-Meta-Modells. Als Grundlage dieser Arbeit werden insbesondere folgende Fragestellungen genauer untersucht und diskutiert:

- Welche Architektur-Modelle eignen sich zum Anbinden von Datenbanken an das World Wide Web?
- Welche Entwurfsmethoden eignen sich zur Modellierung datenbankbasierter Web-Sites?
- Inwieweit sind traditionelle Datenbankmodelle auch für die Strukturierung und Speicherung von Hypertext-Applikationen geeignet?
- Welche Hypertext-Datenmodelle wurden entwickelt und wie können diese auf traditionelle Datenbank-Modelle umgesetzt werden?

In Kapitel 2 werden verschiedene Ansätze zur Kopplung von Datenbank-Systemen an das World Wide Web besprochen und beide Technologieformen gegenübergestellt. Neben den gängigen Architekturformen werden aktuelle Entwicklungstendenzen und Probleme, die sich aus der Kopplung ergeben, aufgezeigt und Lösungsansätze diskutiert. Kapitel 3 beschreibt das Forschungsfeld Hypermedia. Der Schwerpunkt liegt hierbei auf der Darstellung strukturierter Hypermedia-Entwurfsmethoden und Modelle. Anhand ausgewählter Projekte, insbesondere aus den Bereichen Information Retrieval, Web-Site-Management und Integration heterogener Datenquellen, werden aktuelle Entwicklungsaktivitäten im Hypermedia-Gebiet aufgezeigt. In Kapitel 4 wird das abayfor-online-Projekt vorgestellt. Anforderungen und Hintergrund des Projektes dienten als Motivation für die Entwicklung des Meta-Schemas und bilden die Grundlage für die prototypische Implementierung. Das durchgehend in der Arbeit verwendete Beispiel entspricht einem Ausschnitt des abayfor-online-Datenmodells. Kapitel 5 beschreibt die HDBM-Entwurfsmethode und die Entwicklung des zugehörigen Meta-Schemas zur Ablage einer Hypertext-Anwendung innerhalb einer Datenbank. Kapitel 6 zeigt die Systemarchitektur der prototypischen Implementierung auf. Verschiedene Anwendungsmöglichkeiten des Meta-Schemas insbesondere zur verbesserten Ad-hoc-Anfragebearbeitung und Materialisierung von Hypertext-Views werden in Kapitel 7 beschrieben. Das letzte Kapitel schließt die Arbeit mit einer Zusammenfassung und einem Ausblick ab.

Kapitel 2

Datenbanken und World Wide Web

Datenbanksysteme gewinnen für das World Wide Web zunehmend an Bedeutung da sie die effiziente Speicherung sowie die Integrität und Konsistenz großer strukturierter Datenbestände gewährleisten. Das Web bietet demgegenüber eine globale Infrastruktur und Standards für den weltweiten Austausch von Informationen. Die effiziente Verwaltung von komplex strukturierten Daten, eine abgestufte Benutzerautorisierung, die flexible Recherche und ein zuverlässiges Transaktionsmanagement im Mehrbenutzerbetrieb übernimmt das Datenbanksystem während das Web die einfache Bereitstellung und den intuitiven Zugang durch universelle Web-Clients gewährleistet.

Im Rahmen dieses Kapitels erfolgt zunächst eine Gegenüberstellung beider Technologien. Auf dieser Basis werden verschiedene Architekturformen für die Kopplung von Datenbanken an das WWW, deren Motivation und Forschungsschwerpunkte aus technischer Sicht betrachtet.

2.1 Ein Vergleich

Die Eigenschaften von Datenbanksystemen und World Wide Web ergänzen sich in vielerlei Hinsicht. Bei der Kopplung ergeben sich aufgrund der divergierenden Technologien jedoch auch Schwierigkeiten, wie beispielsweise die Behandlung von Transaktionen und die Bewältigung der Sicherheitsproblematik. Dieses Kapitel soll Gemeinsamkeiten und Unterschiede der beiden Technologieformen aufzeigen.

2.1.1 Hypertext und World Wide Web

Die Verknüpfung von Informationen aus verschiedenen Dokumenten mit Hilfe von Referenzen (Links) ermöglicht die nicht-lineare Repräsentation von Wissen. Nach Smith und Weiss [SW88] kann *Hypertext* in Kurzform beschrieben werden als:

> *an approach to information management in which data is stored in a network of nodes connected by links.*

Hypertext bildet ein Netzwerk aus Knoten (Textpassagen) und Kanten (Links) über die sich der Benutzer verschiedene Wege durch ein Informationssystem suchen kann. Das Konzept für transtextuelle Verknüpfungen wurde 1945 von Vannevar Bush [Bus45] mit dem Informationssystem Memex vorgestellt. Die Bezeichnungen 'Hypertext' und 'Hypermedia' wurden von Ted Nelson geprägt, dessen Arbeiten das Fundament heutiger Hypertextforschung bilden. Seine Idee zu dem Informationsbereitstellungssystem XANADU läßt sich bis in das Jahr 1965 zurückverfolgen. Der Durchbruch des Hypertext-Konzepts gelang mit dem kartei-orientierten Hypertext-Programm 'HyperCard' von Apple im Jahr 1987.

Die Kombination von Hypertext und Multimedia bezeichnet man als *Hypermedia*. Knoten des Hypermedia-Netzwerkes können nun neben Text auch beliebige andere Medien beinhalten, wie beispielsweise Bild und Audio.

Das *World Wide Web* basiert auf dem Hypertext-Konzept auch wenn es im Vergleich zu anderen existierenden zum Teil rein experimentellen Hypertext-Systemen [Mau96, Ass96, Spe98] in vielerlei Hinsicht sehr rudimentär ist. Gerade seine Einfachheit ist jedoch der Schlüssel für den weltweiten Erfolg und das exponentielle Wachstum. Die auf dem Client/Server-Prinzip basierende Systemarchitektur erlaubt das Verknüpfen von Dokumenten, die an beliebiger Stelle im weltweiten Internet eingebettet sind. Das WWW gestattet so die translokale Verknüpfung von Dokumenten und ist nach [Lem95]:

> *ein globales, interaktives, dynamisches, Plattformen-übergreifendes, verteiltes, grafisches Hypertext-Informations-System, das im Internet betrieben wird.*

Die Grundidee für das World Wide Web entstand im Jahr 1989 am CERN (Europäisches Zentrum für Teilchenphysik, Genf). Berners-Lee schlägt ein Projekt zum effektiven Meinungs-, Ideen- und Dokumentenaustausch zwischen den Mitarbeitern von CERN vor und entwickelt so die Grundlagen für das WWW. Nach Berners-Lee [BLCL+94], kann das World Wide Web durch folgende Punkte charakterisiert werden:

- **Hypertext-Markup-Language (HTML):** Eine Auszeichnungssprache, die jeder WWW-Client versteht.
- **Uniform-Resource-Locator (URL):** Ein Adressierungssystem, das alle Ressourcen im Internet eindeutig identifiziert. Ein URL setzt sich im wesentlichen aus dem verwendeten Protokoll, dem Servernamen und dem Pfad des HTML-Dokuments auf dem Zielrechner zusammen.
- **Hypertext-Transfer-Protocol (HTTP):** Ein Netzwerkprotokoll, das die Kommunikation zwischen Web-Clients und Web-Servern festlegt.

Die *Hypertext Markup Language* (HTML) ist die Sprache, mit deren Hilfe Web-Seiten erstellt werden. HTML basiert auf der *Standard Generalized Markup Language* (SGML), ein von der International Standards Organization (ISO) festgelegter Standard zur Beschreibung von Dokumenten. Als einfache Auszeichnungssprache, durch die lediglich

markiert werden soll, wie bestimmte Textpassagen auszusehen haben, bringt HTML aber auch einige Nachteile mit sich:

- *Erweiterbarkeit*: HTML ermöglicht weder die Ergänzung eigener Tags noch die Spezifizierung individueller Attribute zur semantischen Auszeichnung von Daten.
- *Struktur*: In HTML können keine zusammenhängenden Datenstrukturen beschrieben werden.
- *Validierung*: HTML fehlen Sprachspezifikationen, die es Anwendungen ermöglichen würden, die strukturelle Validität der Daten zu überprüfen.

Hypertext-Knoten sind einfache HTML-Text-Files, die im Dateisystem des zuständigen Internet-Hosts liegen. Bei HTML Dokumenten handelt es sich um semistrukturierte Daten. Man bezeichnet Daten als semistrukturiert, wenn ihnen kein vorher festgelegtes Schema zugrunde liegt und die Struktur unvollständig oder irregulär sein kann.

Hyperlinks sind die Basiskonstrukte der Interaktion im WWW. Sie sind direkt in den Dokumentkörper eingebettet und referenzieren über einen URL das Zieldokument. Nachteil ist, daß das Zieldokument von dieser Abhängigkeit unberührt bleibt. Auf diese Weise können leicht Inkonsistenzen entstehen, die auch als *dangling links* bezeichnet werden. Im WWW sind Hyperlinks typfrei. Deshalb ist es nicht möglich, ihnen eine anwendungsbezogene Semantik zu verleihen.

Die Kommunikation zwischen dem Web-Browser und dem Web-Server erfolgt über das zustandslose *Hypertext Transfer Protokoll* (HTTP) [FGM+97], das auf TCP/IP aufsetzt. Sobald der Anwender im Web-Browser einen Link aktiviert, wird eine Verbindung zum Web-Server aufgebaut. Ist der Transfer abgeschlossen, wird die Verbindung wieder abgebrochen wodurch alle zugehörigen Sitzungsinformationen verloren gehen.

Das World Wide Web kann demnach vereinfacht durch folgende Definition formal beschrieben werden [Mih96]:

Definition 2.1.1.1
Das World Wide Web ist ein Quintupel $H = (V, E, \sum, r, \psi)$ *wobei gilt:*

- V *ist die Menge aller HTML-Dokumente auf allen Web-Servern;*
- E *ist die Menge aller Links, die in diesen Dokumenten definiert werden;*
- $\sum$ *ist eine ASCII-Zeichenmenge;*
- $\tau : V \to \sum^*$ *bildet jeden Knoten durch Eliminierung und Auswertung der HTML-Tags auf den damit assoziierten Text ab;*
- $\psi : E \to (V \times N^2) \times (V \times N)$ *assoziiert jeden Link mit einem Paar* $((v_1, i, j), (v_2, k))$ *wobei* v_1 *ein Knoten ist, dessen Dokument den Link beinhaltet,* i *und* j *sind Start- und Endposition des Ankers,* v_2 *ist der Zielknoten und* k *ist die Position innerhalb des Zielknotens (Null, falls nicht spezifiziert);*

2.1.2 Datenbanksysteme

In diesem Kapitel werden theoretische und methodische Grundlagen von Datenbanksystemen erläutert, soweit sie zum Verständnis der verschiedenen Systemarchitekturen von Web/DB-Kopplungen notwendig sind. Für weiterführende Informationen sei auf die umfangreiche Fachliteratur verwiesen: [EN94], [Dat95], [SS83], [Heu92], [Ull88], [Vos94], [KS91].

Datenbanktechnologie bezeichnet nach R. Dittrich [Dit98] die Gesamtheit der Konzepte, Methoden, Systeme und Werkzeuge für die Organisation und den Betrieb von Datenbanken. Eine *Datenbank* (DB) ist eine Menge zusammengehörender Daten, mit folgenden Eigenschaften:

- dauerhaft verfügbar
- potentiell groß
- integriert (über mehrere Anwendungen mit überlappendem Informationsbedarf)
- verwendbar unabhängig vom Erzeugungsprogramm
- parallel zugreifbar
- konsistent, sicher
- flexibel und effizient handhabbar

Die Software zur Verwaltung von Datenbanken bezeichnet man als *Datenbankverwaltungssysteme* oder *Database Management System* (DBMS). Diese Software realisiert zum einen ein *Datenmodell*, das den konzeptuellen Rahmen zur logischen Datenorganisation bildet und Mittel zur Strukturierung und Beschreibung der Daten (DDL) bereitstellt. Daneben bietet sie Mittel für den Datenzugriff und die Datenmanipulation, wie beispielsweise DB-Operatoren (DML) und Anfragesprache. Auf der physischen Ebene bieten Datenbanksysteme komplexe Speicherstrukturen und Indexstrukturen, die die effiziente Speicherung großer Datenmengen und das schnelle Auffinden von Daten gewährleisten.

Eine wichtige Aufgabe des DBMS ist die *Steuerung und Überwachung* der Daten. Dazu gehören das Transaktionsmodell, Konsistenzsicherung, Trigger, Zugriffsschutz, Datensicherung, Archivierung und Hintergrundspeicherverwaltung. Wenn mehrere Applikationen gleichzeitig auf die Datenbank zugreifen, müssen die Zugriffe synchronisiert werden. Diese Aufgabe übernimmt die *Concurrency Control Komponente* eines DBMS. Die Concurrency-Komponente verwaltet die Zugriffe verschiedener Transaktionen, um Konsistenzverletzungen der Daten zu vermeiden.

Die Funktionalität eines Datenbanksystems läßt sich nach dem ANSI-Referenzmodell [ANS] durch drei Abstraktionsebenen beschreiben:

- die *interne Ebene* beschreibt, wie die Daten der zu verwaltenden Datenbank im Sekundärspeicher abgelegt werden.

- die *konzeptuelle Ebene* beschreibt das der Datenbank zugrunde liegende Datenmodell. Das logische Schema basiert auf den Konzepten eines Datenmodells (z.B. relational, objektorientiert) und legt die Struktur der Datenbank fest.
- die *externe Ebene* beschreibt die verschiedenen Sichtweisen der Applikationen auf die zugrundeliegende Datenbank.

Die ANSI/SPARC-Architektur gilt als Referenzmodell für die Architektur von Datenbanksystemen und gewährleistet die Forderung nach *Datenneutralität* und *Datenunabhängigkeit*. Datenunabhängigkeit ermöglicht den Anwendungsprogrammen eine Benutzung der DB-Daten, ohne Details zur systemtechnischen Realisierung zu kennen. Dies läßt sich durch logische Datenmodelle und deklarative Anfragesprachen erzielen. Datenneutralität bedeutet, daß die Strukturierung der Daten innerhalb einer Datenbank auf keine spezielle Anwendung zugeschnitten ist, sondern unabhängig von der externen Sichtweise verschiedene Arten von Awendungen in gleicher Weise unterstützt.

2.1.3 Gegenüberstellung

Nachdem in den beiden vorangegangenen Abschnitten die Technologieformen World Wide Web und Datenbanksysteme kurz vorgestellt wurden, sollen nun die Unterschiede und Gemeinsamkeiten dieser beiden Technologien deutlich gemacht werden (siehe Tabelle 2.1).

Eigenschaften	WWW	DBS
Verbindung	• Netzprotokoll HTTP • Zustandslos • Keine Sitzungen	• Verbindung überdauert reine Datenübertragung • Sitzungen möglich
Steuerung und Überwachung	• Keine Transaktionen • Eingeschränkter Zugriffsschutz • Ablage im Dateisystem • Keine Gewährleistung der Datenintegrität	• Transaktionsmodell (ACID) • Zugriffsschutz mit Nutzerkonzept • Concurrency Control • Datensicherung/Archivierung • Hintergrundspeicherverwaltung • Recovery • Gewährleistung der Datenintegrität
Modellierung und Strukturierung	• Semistrukturierte Daten • Komplexe Typen • Kein logisches Datenmodell	• Strukturierte Daten • Logisches Datenmodell mit DDL • In RDB nur einfache Typen
Anfragesprache und Datenmanipulation	• Keine Anfragesprache • Eingeschränktes Retrieval • Navigierender Zugriff	• Mächtige Anfragesprache • Effizienter Zugriff auf Daten
Indexierung der Datenbestände	• Search Engines	• Verschiedene Indexstrukturen: B-Bäume, R-Bäume, usw.

Tabelle 2.1: Vergleich von Datenbanken und WWW

Die Tabelle zeigt, daß sich die Eigenschaften von WWW und Datenbanksystemen in vielen Punkten sehr gut ergänzen. Die Anbindung von Datenbanken an das Web eignet sich vor allem für die Verwaltung großer Datenmengen, die vorwiegend strukturiert vorliegen und sich über die Zeit ändern. Dazu gehören beispielsweise Produktkataloge, digitale Bibliotheken und Kiosk-Informationssysteme.

Die Vorteile der Kopplung ergeben sich auf der Ebene der Benutzeroberfläche durch den einheitlichen, intuitiven Zugang durch universelle Web-Clients und auf der Ebene der Datenhaltung durch die einfache Verwaltung großer Datenmengen. Von Web-Applikationen genutzte Daten werden häufig von verschiedenen Anwendern abgerufen und verändert. Die Verwaltung von verschiedenen Nutzerzugriffen auf einen gemeinsamen Datenpool ist aber eine typische Aufgabe eines DBMS (concurrency control). Weiterhin gewährleistet die referenzielle Integrität des Datenbanksystems die Konsistenz der zum Anfragezeitpunkt generierten Hyperlinks. Zur Laufzeit berechnete Hyperlinks können außerdem abhängig vom jeweiligen Benutzerprofil angezeigt oder ausgeblendet werden.

Ein weiterer Vorteil der Kopplung ergibt sich aus den erweiterten Suchmöglichkeiten von Datenbanken. Im Web hat der Anwender nur die Möglichkeit, mit Hilfe sogenannter Suchmaschinen (Search Engines) nach Informationen zu suchen. Eine strukturierte Anfragesprache existiert dabei nicht. Allen Suchmaschinen liegt eine ähnliche Arbeitsweise zugrunde: Programme erfassen verschiedenste WWW-Server, katalogisieren und analysieren die dort bereitgestellten Informationen und speichern diese in Datenbanken. Ein Anwender kann den so erstellten Index nutzen und nach einzelnen Schlagwörtern suchen oder diese gegebenenfalls durch UND, ODER oder andere einfache Kombinationen verknüpfen. Als Ergebnis erhält der Anwender eine Liste von URLs. Es gibt unterschiedliche Prinzipien der Aufbereitung von Informationen durch eine Suchmaschine. Im wesentlichen werden zwei Ansätze unterschieden:

- *Volltext-Indexierung:* Hier wird der Text der Web-Seiten automatisch indexiert. Art und Umfang der Indexierung unterscheiden sich stark (Schlüsselworte, Stoppworte). Die Schlagworte werden durch Datenbanken verwaltet (Infoseek[1], Altavista[2], Webcrawler[3]). Einige Search-Engines erfassen nur Meta-Tags und den Titel von HTML-Dokumenten (ALIWEB)).

- *Intellektuelle Gliederung:* Die zu erschließenden Seiten werden intellektuell betreut und Oberkategorien bzw. tiefergehenden Unterkategorien zugewiesen, über die der Anwender suchen kann (Dino[4], Yahoo[5]).

Zur Erfassung der Informationen gibt es verschiedene Navigationsstrategien. Man unterscheidet in erster Linie Breiten- von Tiefensuche [Bra95], abhängig davon in welcher

[1] `http://www.infoseek.de` und `http://www.infoseek.com`
[2] `http://www.altavista.de` und `http://www.altavista.digital.com`
[3] `http://www.webcrawler.com`
[4] `http://www.dino-online.de`
[5] `http://www.yahoo.de` und `http://www.yahoo.com`

Reihenfolge die URLs im Web verfolgt werden. Dabei bleibt festzuhalten, daß kein von einer Suchmaschine aufgebauter Index eine vollständige und korrekte Abbildung aller WWW-Informationen ist. Ein von einem DBMS aufgebauter Index wird im Gegensatz hierzu immer aktuell gehalten. Zudem ist durch das Datenbankschema eine Semantik vorgegeben, die bei einer Verschlagwortung durch Indexserver nicht erreicht werden kann.

Weiterhin fällt auf, daß dem World Wide Web kein logisches Datenmodell zugrunde liegt, das ähnlich dem Datenbankbereich als Basis für die Definition von Anfragen und Views dienen könnte. Datenbanken gewährleisten demgegenüber durch die klare Trennung der verschiedenen Architektur-Ebenen und die damit einhergehende Datenunabhängigkeit die einfache Skalierbarkeit und Wartbarkeit einer Anwendung.

Während Datenbanken verschiedenste Sicherheitsmechanismen durch Authentifizierung, unterschiedliche Sichten auf Daten und Rollen mit bestimmten Rechten bieten, sind im WWW nur ein einfaches Hostname-Filtering, Basic-Authentification und Digest-Authentification möglich. Als generelles Verschlüsselungsverfahren für den Schutz der Datenübertragung zwischen WWW-Server und Client dienen das Secure HTTP (S-HTTP) und die Secure Socket Layer (SSL).

Probleme bei der Kopplung von Datenbanksystemen an das Web resultieren vorwiegend durch das zustandslose Umfeld des HTTP-Protokolls und die dadurch erschwerte Transaktionsverabeitung und Benutzerverwaltung (siehe Kapitel 2.3). Ein weiteres Problem ist die Einfachheit von HTML, wodurch die Benutzerinteraktion und Modellierung von Semantik innerhalb der HTML-Seiten stark beeinträchtigt werden. In den folgenden Kapiteln werden verschiedene Architekturformen und Lösungsansätze für die genannten Problemstellungen diskutiert.

2.2 Systemarchitekturen

Es gibt vielfältige Ansätze, Datenbanken und Web-Server zu kombinieren [BG98], [Loe97], [NS96], die sich im wesentlichen durch die unterschiedliche Umsetzung der im vergangenen Kapitel dargestellten Eigenschaften beider Technologien auszeichnen. Dies betrifft insbesondere verwendetes Verbindungsprotokoll, Transaktionsverarbeitung, Anwendungslogik, Präsentationslogik und Sicherheitsmechanismen. Welcher Ansatz für eine konkrete Problemstellung ausreichend ist, hängt demnach von der jeweiligen Anwendung ab. Die Klassifikation von Anwendungen ist unter anderem durch Zugriffsart (schreibend, lesend), Datensensitivität, Benutzeridentifikation, Sicherheitsbedarf, Sitzungslänge und Aktualität der Anwendung möglich [Loe98].

2.2.1 Anforderungen

Durch die Kombination der gegensätzlichen Eigenschaften beider Technologieformen – Datenbanken und World Wide Web – entstehen neue auf das WWW ausgerichtete Anwendungen, die folgenden Anforderungen genügen sollten:

- **Skalierbarkeit:**
  Das Internet hat in den vergangenen Jahren einen rasanten Zuwachs erfahren. Die Architektur datenbankbasierter Informationssysteme sollte deshalb leicht skalierbar sein. Dies betrifft insbesondere die flexible Verteilung verschiedener Komponenten (Web-Server, Middleware und Datenbank-Server) auf unterschiedliche Rechner.

- **Flexibilität und Offenheit:**
  Die technologischen Grundlagen im World Wide Web ändern sich ständig. Interoperabilität, Portabilität und Anpassungsfähigkeit an unterschiedliche Browser, Plattformen usw. sind ausschlaggebend für die Langlebigkeit eines Produktes.

- **Transaktionskontrolle:**
  Das WWW ist ein zustandsloses Umfeld. Im Datenbankbereich ist sitzungsorientiertes Arbeiten vor allem im Bereich der Transaktionsverarbeitung Voraussetzung. Mechanismen zur Speicherung und Verwaltung von Zustandsinformationen sind erforderlich.

- **Performance:**
  Die Gesamt-Performance von WWW-Applikationen mit Datenbankanbindung setzt sich aus verschiedenen Parametern zusammen. Dabei ist zu beachten, daß nicht nur die einzelnen Parameter Einfluß auf die Gesamt-Performance haben, sondern auch deren Zusammenspiel diese beeinflußt. Insbesondere wirken sich hier die verwendete Systemarchitektur und die Verteilung der einzelnen Komponenten auf verschiedene Rechner aus. Daneben ist die Zeit des Verbindungsaufbaus und der Verbindungsdauer für WWW-Applikationen ein wichtiger Parameter.

- **Sicherheit:**
  Datenbank-Server und Web-Server bieten unterschiedliche Mechanismen vor unbefugtem Zugriff und Systemfehlern. Der Zugriff auf statische HTML-Seiten und CGI-Programme wird auf der Ebene des Web-Servers durch Hostname-Filtering, Basic- und Digest Authentification geschützt. Auf Datenbank-Ebene erfolgt der Zugriffsschutz durch Nutzername und Paßwort. Um den sicheren Zugriff auf die Datenbank über das Web zu gewährleisten, muß deshalb die Anbindungssoftware die Authentifizierung der Benutzer auf Datenbank-Ebene ermöglichen. Neben der Authentifizierung ist auch die sichere Übertragung der Daten wichtig. Hierfür werden beispielsweise SHTTP (secure HTTP) und SSL (Secure Socket Layer) unterschieden.

Die unterschiedlichen Techniken zur Kopplung von Datenbanksystemen an das World Wide Web lassen sich in zwei grundlegende Architekturformen einordnen. Ein zentraler Unterschied liegt dabei in der Trennung und Verteilung der einzelnen Architekturschichten (Bedienoberfläche, Anwendungslogik, Datenhaltung) auf physische Einheiten (Client, Server). Während bei der *Web-Server-seitigen Datenbankanbindung* nur die Oberfläche mit der Bedienung im Web-Browser üblicherweise mittels HTML realisiert ist, wird bei

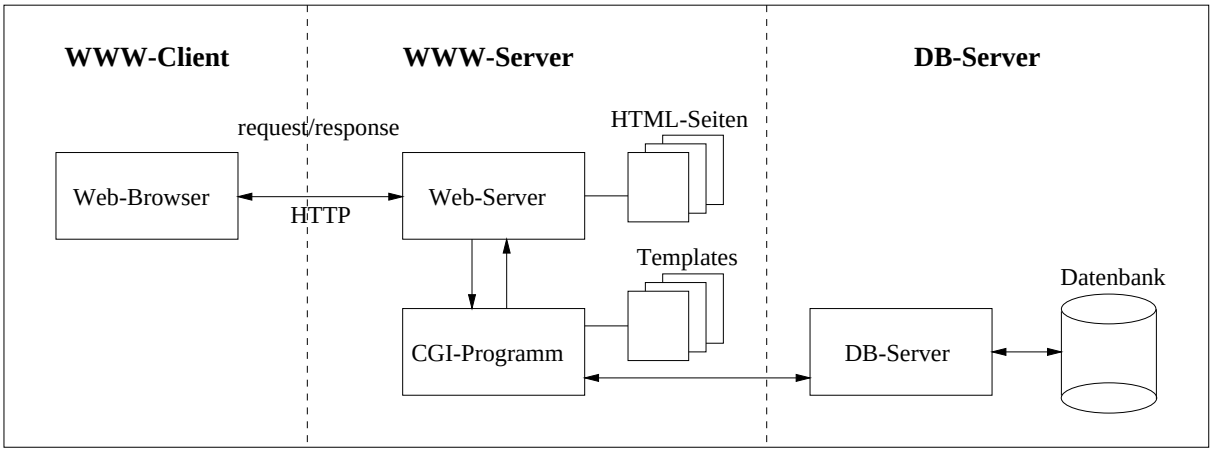


Abbildung 2.1: CGI-Anbindung

der *Client-seitigen Datenbankanbindung* die Anwendungslogik z.B. in Form eines Applets zur Ausführung in den Web-Browser geladen.

2.2.2 Web-Server-seitige Anbindung

Bei der Web-Server-seitigen Datenbankanbindung erfolgt die Kommunikation zwischen dem Web-Browser und dem Datenbank-Server ausschließlich über den Web-Server. Der Benutzer setzt im Browser, beispielsweise mit Hilfe von HTML-Eingabeformularen, Datenbank-Anfragen ab. Der Web-Server nimmt die Anfrage des Clients entgegen und reicht sie über eine Middleware an den Datenbank-Server weiter. Die Kopplung zwischen Web-Server und Datenbanksystem wird durch einfache *Web-Gateways* oder *Application-Server* realisiert, die im wesentlichen die Extraktion und Auswertung der Benutzereingabe, den Zugriff auf die gewünschte Datenbank und die Generierung der HTML-Ergebnisseiten steuern.

Bei der Web-Server-seitigen Datenbankanbindung werden vor allem die Anbindung über das Common Gateway Interface (CGI) [Gun96] oder eine proprietäre Web-Server-API (Application Programming Interface) unterschieden.

Frühe Implementierungen von Web-Server-basierten Applikationen basierten fast ausschließlich auf dem Common Gateway Interface. Dieser Standard legt fest, wie Programme vom Web-Server aufgerufen und Daten zwischen Web-Server und dem CGI-Programm transferiert werden. Der Anschluß zum Web-Server erfolgt über Kanäle der Standardeingabe und -ausgabe (Pipes). Für die Implementierung eines CGI-Programms kann eine beliebige Programmiersprache verwendet werden (z.B. C, C++, Perl). Das vom Web-Server gestartete CGI-Programm stellt eine Verbindung zur Datenbank her, leitet die Anfrage weiter und generiert nach der Auswertung eine HTML-Seite mit den gewünschten Ergebnissen (siehe Abbildung 2.1). Zur Definition der Datenbank-Anfragen und der Ergebnispräsentation dienen häufig HTML-Templates (siehe Kapitel 5.4.1).

Bei dem CGI-basierten Ansatz involviert jede Datenbankanfrage den Start eines neuen

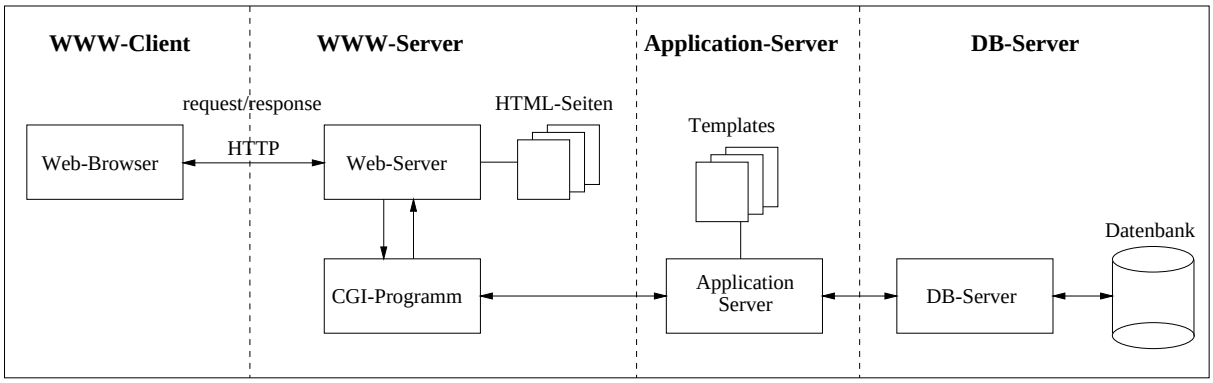


Abbildung 2.2: CGI-Anbindung mit Application-Server

Prozesses. Dieses einfache *Ein-Prozeß-pro-Anfrage-Prinzip* von CGI beeinträchtigt aber neben dem hohen Ressourcenverbrauch vor allem die Performance bei der Anfragebearbeitung. Daneben ist kein sitzungsorientiertes Arbeiten möglich. Um den heutigen Anforderungen gerecht zu werden, greifen CGI-basierte Datenbankanwendungen deshalb häufig auf permanent laufende Application-Server zurück. Das CGI-Programm wird minimiert und dient nur noch zum Durchreichen der Daten. Der Application-Server verwaltet permanente Datenbankverbindungen, speichert Zustandsinformationen, ermöglicht komplexe Transaktionen und optimiert die Anfragebearbeitung (siehe Abbildung 2.2). Weiterhin ist durch die zweistufige Lösung eine bessere Lastverteilung möglich.

FastCGI [Ope96] ist eine Erweiterung des CGI-Standards und bietet verschiedene Vorteile gegenüber herkömmlichen CGI-Programmen und Web-Server APIs. FastCGI Prozesse bleiben nach einer Anfrage persistent, haben durch caching-Strategien eine erheblich bessere Performance und können auf entfernten Rechnern ausgeführt werden. Daneben bleiben die Vorteile des CGI-Ansatzes, wie beispielsweise die Unabhängigkeit von einer Programmiersprache und Prozess-Isolation bestehen. Es bleibt abzuwarten, wie sich diese Schnittstelle durchsetzen wird. Alle aktuellen Microsoft und Netscape-Server-Produkte unterstützen FastCGI.

Alternativ zu CGI können dynamische Applikationen über eine proprietäre Schnittstelle des Web-Servers hinzu gebunden werden. Die zwei bekanntesten Schnittstellen auf diesem Gebiet sind die *Netscape Server API* (NSAPI) [Net] und die *Internet Server API* (ISAPI) [Mic97] von Microsoft. Im Unterschied zu CGI-Programmen läuft die Server-Applikation im Prozeßraum des Web-Servers und teilt damit dessen Daten- und Kommunikationsressourcen (siehe Abbildung 2.3). Neben der Performanzsteigerung und der Verringerung des Ressourcenverbrauchs, bietet dies die Möglichkeit, Einfluß auf Server-Funktionen, wie z.B. die Fehlerbehandlung und Benutzerauthentifizierung zu nehmen. Server-Applikationen, die eine proprietäre Schnittstelle des Web-Servers verwenden, laufen üblicherweise in einer multithreading-Umgebung (abhängig von der Web-Server-Architektur) ab und müssen im Gegensatz zu CGI-Programmen, die isoliert ablaufen, multithreading-sicher sein. Fehlerhafte Server-Applikationen führen durch den gemeinsam verwendeten Prozeßraum mit dem Web-Server zu Server-Abstürzen. Der zum Web-

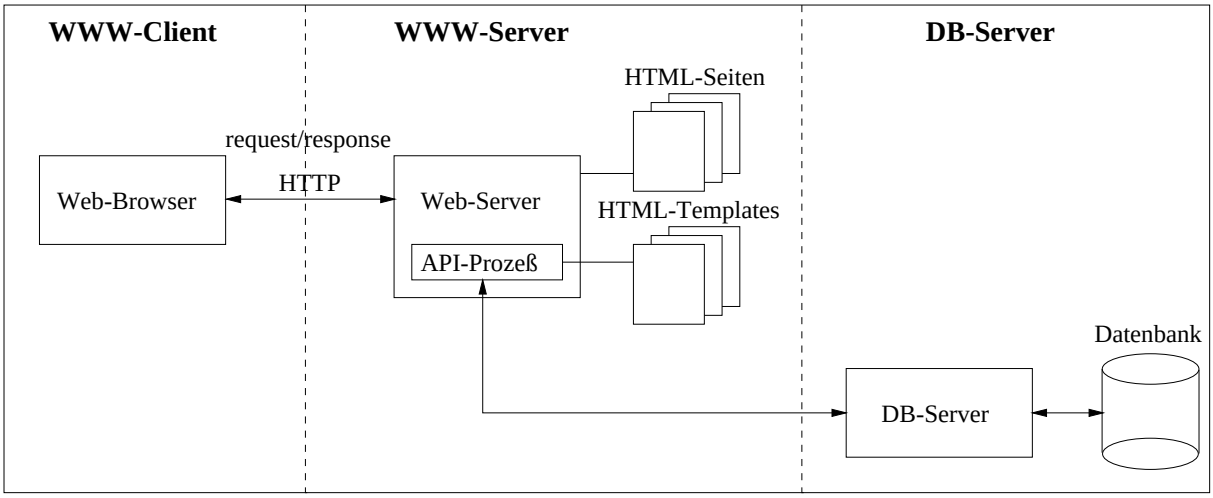


Abbildung 2.3: Erweiterbare Server-API

Server hinzu gebundene Applikationscode wird deshalb möglichst klein gehalten. Ein oder mehrere Application-Server, die als Vermittler zwischen dem Web-Server und der Datenbank agieren, übernehmen die Anfrageübermittlung und Generierung der dynamischen HTML-Seiten. Mangelnde Standards erschweren die Anbindung über eine proprietäre Web-Server API.

2.2.3 Client-seitige Anbindung

Vor allem der Einsatz des zustandslosen HTTP und die starke Belastung des Web-Servers (Flaschenhalsproblematik) führen bei der Server-seitigen Datenbankanbindung zu Problemen. Die mangelnde Informationsverarbeitung auf der Client-Seite, wie beispielsweise eine Syntaxüberprüfung der Eingabe im Web-Browser, erhöht zusätzlich die Anfrage- und Bearbeitungszeiten. Um diese Nachteile zu umgehen, kann bei der Client-seitigen Datenbankanbindung mit Hilfe von Browser-Erweiterungen mit Java oder ActiveX eine direkte Kommunikation zwischen dem Web-Client und der Datenbank erfolgen.

Die Web-Anbindung von Datenbanken über die von Sun Microsystems entwickelte objektorientierte Programmiersprache *Java* [Dic97], [Jep97] hat in der Vergangenheit große Erwartungen geweckt. Diese Programmiersprache zeichnet sich insbesondere durch ihre Plattformunabhängigkeit aus. Ein *Java-Compiler* (Teil der Java Entwicklungsumgebung JDK) übersetzt Java-Quellcode in plattformunabhängigen Bytecode, der dann von einem Java-Interpreter (Java Virtual Machine JVM) auf der jeweiligen Plattform ausgeführt wird. Die gewünschte Anwendung kann so bei Bedarf in Form eines Java-Applets auf den Rechner des Anwenders übertragen und dort in einem Java-fähigen Browser ausgeführt werden. Damit ermöglicht Java auch auf Client-Seite die Ausführung komplexer Anwendungslogik, wie z.B. die Überprüfung von Benutzereingaben und eine flexible Oberflächenprogrammierung. Außerdem sind Java-Applets nicht auf die Verwendung des zustandslosen HTTP angewiesen, sondern können DB-Verbindungen während Benutzerin-

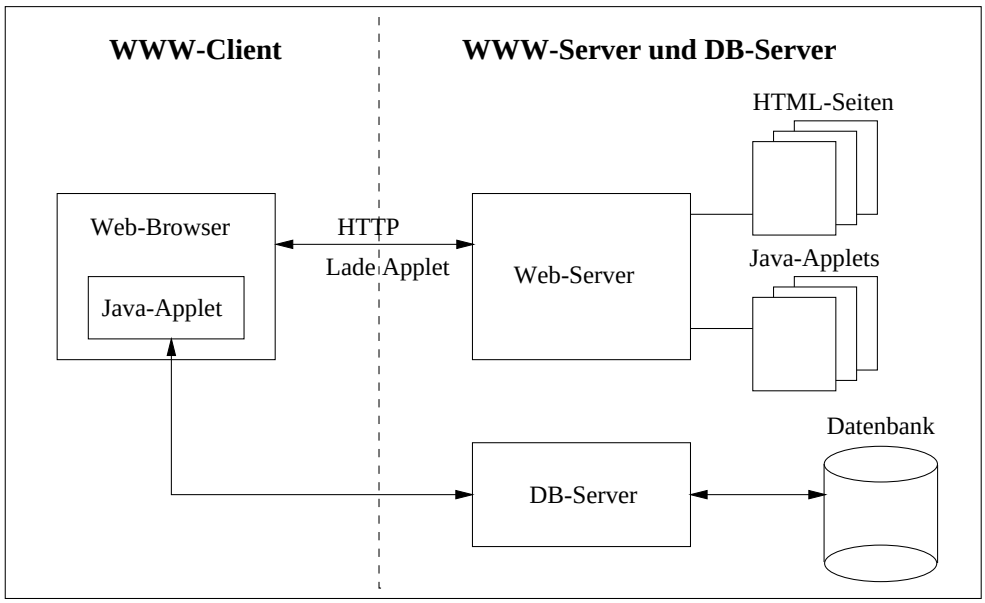


Abbildung 2.4: Two-Tier-Architektur

teraktionen offen halten. Eine Java-basierte Datenbankanbindung kann deshalb die volle Funktionalität des darunter liegenden Datenbanksystems nutzen und auf performantere Kommunikationsprotokolle zurückgreifen.

Dem gegenüber stehen die potentiell erhöhten Ladezeiten durch die Übertragung des Java-Applets und die geltenden Sicherheitsrestriktionen für untrusted Java-Applets[6]. Ein weiterer Nachteil ist die vergleichsweise aufwendige Oberflächenprogrammierung. HTML dient nur zur Einbettung der Java-Anwendung. Alle Eingabeformulare, Ergebnistabellen und Grafiken, die für eine Datenbankrecherche nötig sind, werden in Java programmiert. Durch die langen Ladezeiten im Internet eignet sich eine Client-seitige Datenbankanbindung mit Java vor allem für Intranet-Anwendungen. Ein weiteres Anwendungsfeld sind interaktive Anwendungen mit komplexer Anwendungslogik, die mit HTML nicht realisiert werden können (z.B. Informationssysteme zu Geo-Datenbanken).

Bei der Anbindung von Datenbanken über Java-Applets kann zwischen einer *zweistufigen (Two-Tier)* und einer mehrstufigen, üblicherweise der *dreistufigen (Three-Tier)* Prozeßarchitektur unterschieden werden. Das Two-Tier-Modell entspricht einer typischen Client-Server-Architektur bei der ein heruntergeladenes Java-Applet mittels eines datenbankspezifischen Protokolls direkt mit dem gewünschten Datenbank-Server kommuniziert (siehe Abbildung 2.4). Untrusted Java-Applets ist es nur erlaubt, eine Netzwerkverbindung zu dem Rechner herzustellen, von dem sie geladen wurden. Bei einer Two-Tier-Architektur müssen deshalb Web-Server und Datenbank-Server auf dem gleichen Rechner laufen.

[6]In Java unterscheidet man zwischen trusted und untrusted Applets. Trusted (vertrauenswürdig) ist ein Applet dann, wenn es signiert ist oder von einer als vertrauenswürdig deklarierten Stelle, beispielsweise der lokalen Platte, stammt. Alle sonstigen Applets sind unstrusted und unterliegen verschiedenen Sicherheitsrestriktionen.

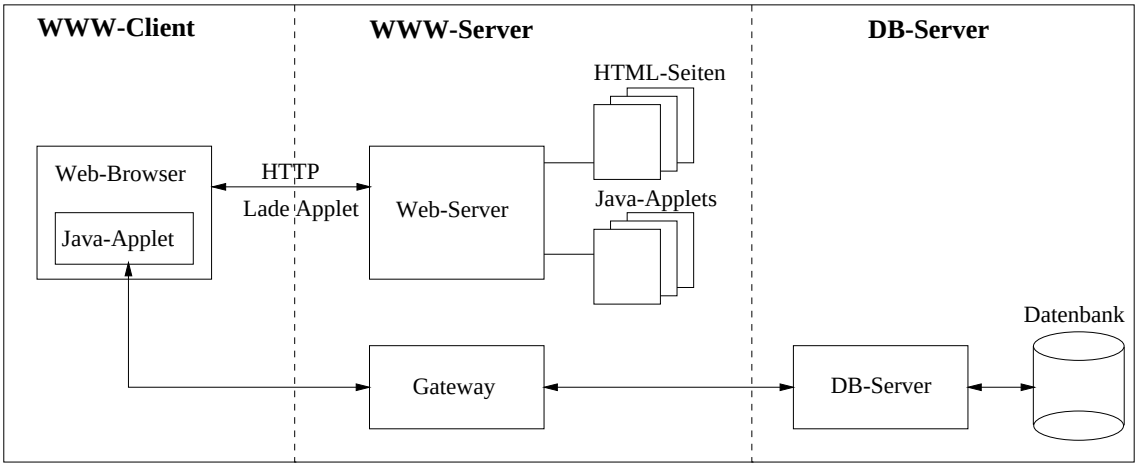


Abbildung 2.5: Three-Tier-Architektur

Bei der Three-Tier-Architektur wird ein Gateway oder Application-Server als Vermittler zwischen den Web-Browser und die Datenbank geschaltet. Das Applet kommuniziert mit dem Gateway, das wiederum die Anfragen an die Datenbank weiterleitet. Die Datenbank führt die Anfragen aus und schickt das Ergebnis über das Gateway zurück zum Browser. In Abbildung 2.5 wird eine Three-Tier-Architektur gezeigt.

Um eine datenbankunabhängige Programmierung von Java-Anwendungen zu gewährleisten, entwickelte JavaSoft zusammen mit führenden Datenbankanbietern JDBC (Java Database Connectivity) [Sun99], eine einheitliche Schnittstelle für den Zugriff auf relationale SQL Datenbanksysteme. JDBC basiert wie Microsofts ODBC-Schnittstelle auf dem Industriestandard X/Open SQL CLI (Call Level Interface). Die Gesamtarchitektur von JDBC umfaßt mehrere Schichten, wie in Abbildung 2.6 dargestellt. Von zentraler Bedeutung ist dabei der JDBC Driver Manager, der die unterschiedlichen Treiber verwaltet. Diese wiederum können auf verschiedene Datenbanksysteme zugreifen. Die Treiber sind in Java (pure Java) oder einer anderen Programmiersprache (native) codiert. Für SQL-Datenbanksysteme müssen sie mindestens den Sprachumfang ANSI-SQL-92 Entry Level erfüllen, um als JDBC Compliant bezeichnet zu werden.

Je nach Art des Treibers werden vier unterschiedliche Kategorien unterschieden: JDBC-ODBC-Bridge, Native API Partial Java-Treiber, Native Protocol All-Java-Treiber und Net Protocol All-Java-Treiber. Die JDBC-ODBC-Bridge konvertiert die JDBC-Methodenaufrufe in entsprechende ODBC-Funktionsaufrufe. Damit kann dieser Treiber alle ODBC-fähigen Datenbanken an das WWW anbinden. Native API Partial Java-Treiber konvertieren JDBC-Aufrufe direkt in Kommandos datenbankspezifischer Middleware Produkte. Sie basieren damit auf proprietärer und plattformabhängiger Datenbanksoftware und können nicht als Applet geladen werden. Ein Pure-Java-Treiber mit datenbankspezifischem Protokoll ist vollständig in Java programmiert und kann über das Netz in den Web-Client geladen werden. Diese Treiber kommunizieren direkt mit dem DB-Server. Der Pure-Java-Treiber mit DB-unabhängigem Protokoll besteht auch aus reinem Java-Code, JDBC-Kommandos werden jedoch in ein datenbankunabhängiges Protokoll übersetzt.

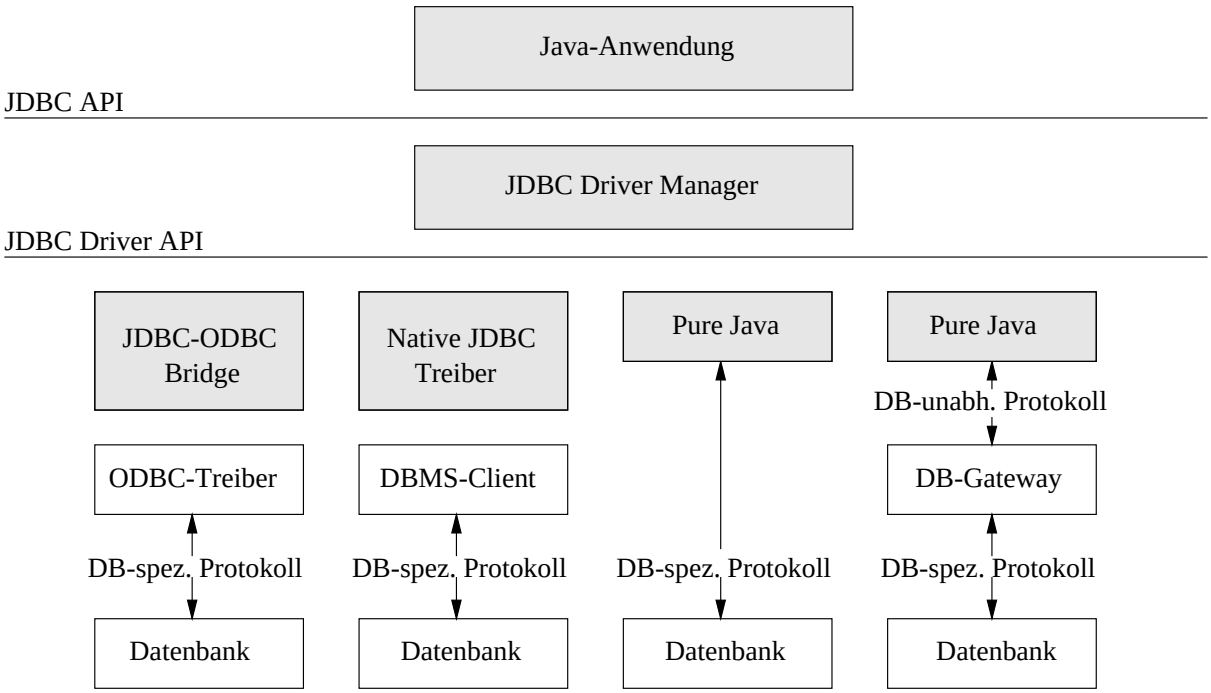


Abbildung 2.6: JDBC-Treiber-Typen

Der JDBC-Treiber kommuniziert mit einem Datenbank-Gateway auf dem Server, das die Anfragen in das jeweilige datenbankspezifische Netzwerkprotokoll übersetzt.

2.3 Technische Problemstellungen

Im folgenden Kapitel werden technische Probleme aufgezeigt, die bei der Kopplung von Datenbanken an das Web auftreten.

2.3.1 Verwaltung von WWW-Dialogen und Transaktionen

Ein Dialog bezeichnet die Aneinanderreihung einzelner Dialogschritte und damit den zeitlich gestaffelten Besuch von HTML-Seiten. Aufeinander folgende auf- und abgebaute Verbindungen sind durch das zustandslose Protokoll HTTP 1.0 völlig unabhängig voneinander und können nicht auf Informationen der letzten Verbindungen zurückgreifen. Weiterhin unterstützt HTTP nicht die Identifikation von Web-Clients und ermöglicht keine Sitzungen. Web-Server bieten demnach für die Abarbeitung von Transaktionen standardmäßig keine Unterstützung [AHR+98].

Eines der zentralen Konzepte von Datenbanksystemen ist die Gewährleistung des Transaktionenkonzepts. Eine Transaktion besteht aus mehreren Aktionen, die nach dem *Alles-oder-Nichts-Prinzip* durchgeführt werden. Die Eigenschaften einer Transaktion werden durch das ACID-Prinzip (atomicity, consistency, isolation, durability) beschrieben. Um im Web dennoch sitzungsorientiertes Arbeiten über mehrere HTML-Seiten zu ermög-

lichen, muß ein Client einer bestimmten Applikation, beispielsweise durch einen Transaktionsidentifikator, eindeutig zugeordnet werden können.

Transaktionen im WWW werden in *single-page-Transaktionen* und *multiple-page-Transaktionen* unterschieden. Bei einfachen single-page-Transaktionen ist die Transaktion auf eine HTML-Seite bzw. auf einen vom Web-Client angeforderten URL beschränkt. Wird dieser URL, beispielsweise in Form eines CGI-Programms, vom Client aufgerufen, so wird eine Verbindung geöffnet, die Transaktion ausgeführt, die generierte HTML-Seite dem Benutzer angezeigt und die Verbindung wieder abgebaut. Im Gegensatz hierzu ist die Verwaltung von multiple-page-Transaktionen über mehrere URLs hinweg deutlich komplexer. Ein Beispiel ist die Erstellung eines Einkaufskorbes. Erst auf der letzten HTML-Seite soll die Bestellung bestätigt und dauerhaft durchgeführt werden.

Für die Realisierung von multiple-page-Transaktionen muß die Middleware alle benötigten Sitzungsinformationen (z.B. Sitzungserkennung, Transaktionsidentifikator, DB-Login ...) über mehrere Web-Seiten verwalten und Mechanismen für deren Übertragung und Speicherung festlegen. Damit übernimmt sie die Aufgaben eines TP-Monitors (Transaction-Processing-Monitor).

Unter anderem lassen sich folgende Varianten zur Dialogverwaltung unterscheiden:

- **Dialogsteuerung über URL-Erweiterung:** Informationen werden mit Hilfe von URL-Anhängseln zwischen Client und Server transferiert.
- **Cookies:** Cookies wurden von Netscape eingeführt und dienen vorwiegend zur Sitzungsverwaltung. HTTP-Cookies werden mit der Meta-Information einer HTML-Seite vom Server zum Browser übertragen und dort temporär gespeichert. Bei jeder Dokumentanforderung an den Web-Server wird das Cookie zum Web-Server übertragen.
- **Formularvariablen:** Zur Identifikation eines Benutzers wird die Sitzungs-Id als versteckte Eingabevariable in HTML-Formularen übertragen.
- **Dialogprogrammierung auf dem Client:** Mit Hilfe von Java, ActiveX und JavaScript kann die Dialogverwaltung durch den Web-Client erfolgen. Die Verbindung zwischen Web-Client und Datenbank-Server erfolgt nicht über das zustandslose HTTP-Protokoll.

Neben der Gewährleistung von sitzungsorientiertem Arbeiten ist die geeignete Verwaltung der multiple-page-Transaktionen erforderlich. Beim Oracle Application Server 3.0 [HB98a] (siehe auch Kapitel 2.4.1 und 6.2) werden Transaktionen in *konfigurierte Transaktionen* und *programmierte Transaktionen* unterschieden. Bei konfigurierten Transaktionen legt der Anwendungsentwickler den Transaktionsnamen und den Präfix aller URLs fest, die zu dieser Transaktion gehören. Programmierte Transaktionen bieten die Möglichkeit, innerhalb von PL/SQL-Prozeduren den Beginn und das Ende einer Transaktion zu definieren.

2.3.2 Authentifizierung und Sicherheit

Authentifikation bezeichnet den Vorgang des beweisens der eigenen Identität. Die Zusicherung der Identität wird als Identifikation bezeichnet. Die Ableitung von Rechten aus dieser Identität bezeichnet man als Autorisierung [CZ95]. Um sowohl die sichere Übertragung von Daten als auch den sicheren Zugriff auf Daten im World Wide Web zu gewährleisten, bieten gängige WWW-Server verschiedende Authentifizierungsmöglichkeiten und Verschlüsselungsalgorithmen an. Bei der Authentifizierung werden vor allem das *Hostname-Filtering*, die *Basic-Authentification* und die *Digest-Authentification* unterschieden. Im Bereich der Datenverschlüsselung wird das *Secure-Socket-Layer* Protokoll (SSL) oder *Secure-HTTP* (SHTTP) eingesetzt. Durch seine Praktikabilität hat sich heute SSL weitgehend durchgesetzt.

Zum autorisierten Zugriff auf Hypertext-Dokumente können im WWW-Server Schreib- und Lesesperren auf einzelne Verzeichnisse oder Datei-Gruppen angelegt werden. Diese Art der Benutzerautorisierung ist für den Schutz von statischen HTML-Seiten ausreichend, für den Datenbankzugriff über ein CGI-Programm oder eine Server-API dagegen weitgehend ungeeignet. Um verschiedene Benutzersichten auf eine Datenbank zu ermöglichen, bieten DBMSe sehr fein granulare Autorisierungsmöglichkeiten. Einzelne Benutzer oder Benutzergruppen haben verschiedene Sichten auf das Datenbankschema, was durch unterschiedliche Schreib- und Leserechte auf einzelne Relationen oder Klassen gewährleistet wird. Mittels einer Abbildung des Web-Benutzers auf eine lokale ID kann der HTTP-Authentifizierungsmechanismus genutzt werden, um eine DB-Verbindung unter der Kennung des Benutzers aufzubauen [Loe98]. Um die Autorisierungskonzepte von Datenbanken im WWW zu nutzen, muß demnach die Software zur Datenbankanbindung an das WWW eine entsprechende Benutzerverwaltung unterstützen.

2.3.3 Interaktionsformen

HTML-Formulare sind insbesondere für Datenbankanwendungen von großer Bedeutung, weil mit ihrer Hilfe DB-Applikationen in das WWW transportiert werden können. Es lassen sich hier Texteditoren, Eingabefelder, Radio-Buttons, Check-Boxes, Menü-Buttons, Selection-In-Lists und Action-Buttons unterscheiden. Allerdings sind diese Formular-Widgets im Rahmen von HTML-basierten Anwendungen verschiedenen Einschränkungen unterworfen, die durch die mangelnde Ausdrucksmächtigkeit von HTML bedingt sind. Beispielsweise können während der Bearbeitung eines Formulars keine Widgets hinzugefügt oder weggenommen werden. Daneben ist es nicht möglich, den Zustand einzelner Widgets in Abhängigkeit zu anderen Bedienelementen gezielt zu beeinflussen. Zur Lösung dieser Probleme muß die gesamte HTML-Seite neu übertragen werden. Dies ist eine relativ kostenaufwendige Lösung. Eine andere Möglichkeit ist der Einsatz von Client-seitigen Scripts (z.B. JavaScript).

2.4 Neue Entwicklungstendenzen

Wurden früher Datenbanksysteme zur reinen Datenhaltung und -speicherung eingesetzt, verwischen heute die Grenzen zwischen Applikation und Datenhaltung zusehends. Das folgende Kapitel soll neue Entwicklungen im Bereich Internet-Datenbanken aufzeigen.

2.4.1 Application-Server mit Komponentenschnittstelle

Der entscheidende Vorteil dieses Ansatzes ist neben der hohen Modularität vor allem die Wiederverwendbarkeit vorhandener Komponenten über standardisierte Schnittstellen. Beispiele hierfür sind der Microsoft Internet Information Server und der Oracle Web Application Server.

Der Microsoft Internet Information Server arbeitet mit einer Komponentenschnittstelle für OLE/COM beziehungsweise ActiveX. Microsoft Active Server Pages (ASP) sind eine ISAPI Server-Erweiterung des Microsoft Internet Information Servers (IIS) 2.0, die serverseitige Web-Programmierung ermöglicht und eine Vielzahl von Objekten beziehungsweise Komponenten für die Entwicklung von Web-Anwendungen zur Verfügung stellt. Die Erweiterung des Web-Servers besteht aus einer Laufzeitumgebung (ASP.DLL) und aus einer für Web-Programmierung nützlichen Komponentenbibliothek. Die Server-Erweiterung ASP.DLL wird nur einmal geladen und läuft im gleichen Prozeß-Adreßraum des Web-Servers. ASP stellt als WWW-Anwendungs-Framework viele Objekte zur Verfügung, wodurch die Funktionalität von ASP beliebig erweitert werden und bestimmten Anwendungsgebieten angepaßt werden kann. Eine eigene ASP-Programmiersprache existiert nicht. Es kann jede beliebige Skriptsprache verwendet werden, falls eine entsprechende Scripting-Engine vorhanden ist.

Der Oracle Application Server 4.0 (OAS) (siehe Abbildung 2.7) arbeitet mit einem CORBA-compliant Web-Request-Broker.

Der OAS besteht aus drei Teilen, die über verschiedene Plattformen verteilt sein können: dem HTTP Server (Web-Server) einschließlich Dispatcher, Oracle Application Server und verschiedenen Application Cartridges. Dabei unterstützt der OAS verschiedene Programmiermodelle (CORBA, Web, Enterprise Java Beans) und Programmiersprachen (Java, Perl, C, PL/SQL, COBOL und LiveHTML). Der Kern des OAS ist der als CORBA-kompatibler Object Request Broker implementierte Web Request Broker (WRB). Die Application Cartridges sind Programme, deren Schnittstellen mit CORBA definiert sind. Neben der Möglichkeit, eigene Programme in Form von Cartridges einzubinden, stehen mit dem OAS 4.0 bereits verschiedende Cartridges zur Verfügung. Dazu gehören Java, C, Perl, PL/SQL, COBOL und LiveHTML. Eine genaue Erläuterung des Oracle Web Application Servers erfolgt in Kapitel 6.2.

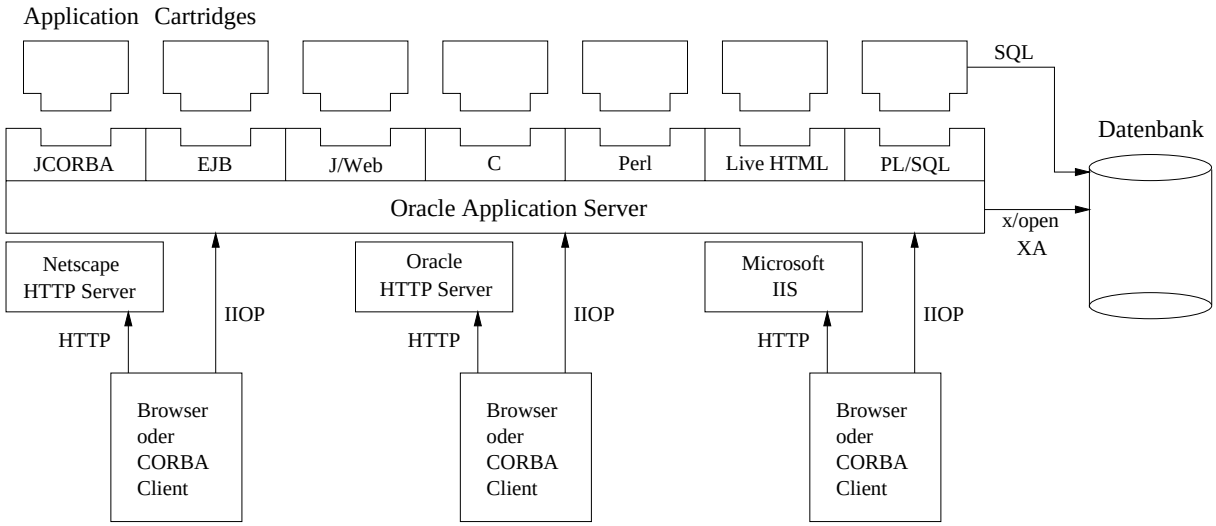


Abbildung 2.7: Architektur des Oracle Application Servers 4.0

2.4.2 Java-Servlets

Java-Servlets sind mit Java-Applets vergleichbar, werden jedoch im Unterschied zu Applets nicht im Web-Browser sondern im Web-Server ausgeführt. Servlets bieten viele Vorteile gegenüber proprietären Server-seitigen-Lösungen wie Active Server Pages (ASP) und server-seitigem JavaScript, da sie mit einer standardisierten Java-Servlet-API entwickelt werden. Das Java Server Toolkit enthält ein Java Server Framework, das mit Java Developers Kit 1.2 (JDK) voll kompatibel ist und die Java Servlet APIs unterstützt. Damit sind Java-Servlets nicht von einem speziellen Web-Server abhängig, sondern können in einem beliebigen Java enabled Server ausgeführt werden. Im Unterschied zu CGI, werden Servlets bei der ersten Anfrage in den Web-Server geladen und initialisiert. Damit sind sie ähnlich zu API-Anwendungen deutlich performanter. Da Servlets auf Java basieren, zeichnen sie sich weiterhin durch Plattformunabhängigkeit aus. Damit ist es möglich, plattform- und herstellerunabhängige Erweiterungen von Web-Servern zu entwickeln. Diese Vorteile führten in jüngster Zeit zu verschiedenen Produkten zur Anbindung von Datenbanken an das WWW, die Java-Servlets unterstützen.

2.4.3 XML

Für die Zukunft planen viele Hersteller von Middleware-Produkten die Einbindung von XML/XSL-Werkzeugen. XML (Extensible Markup Language) [BPSM98], [Gra99], [Bra98] ist eine Markup-Sprache zur Beschreibung von semistrukturierten Dokumenten. Um die Nachteile von SGML und HTML zu umgehen, wurde von einer 1996 gegründeten Arbeitsgruppe des World Wide Web Consortiums (W3C) die Extensible Markup Language, kurz XML, in einem Standardisierungsprozeß entwickelt. Dabei bildet XML eine Untermenge von SGML.

Durch den kommerziellen Erfolg des WWW und die wachsenden Möglichkeiten und

Anforderungen an den Datenaustausch über das Internet wurde deutlich, daß sich HTML für anspruchsvollere Anwendungen nicht eignet. Einschränkungen von HTML sind insbesondere die mangelnde Erweiterbarkeit durch eine fest vorgegebene Anzahl von HTML-Tags. Durch HTML können keine zusammenhängenden Datenstukturen beschrieben werden. Weiterhin fehlen Sprachspezifikationen, die es Anwendungen ermöglichen würden, die strukturelle Validität der Daten zu überprüfen. Durch die Verwendung von HTML geht die reiche, innere Struktur der Daten verloren. Auf Client-Seite können die HTML-Dokumente beispielsweise nicht mehr hinsichtlich ihrer Struktur weiterverarbeitet werden. XML ist deshalb insbesondere in Verbindung mit Datenbanken und World Wide Web interessant.

Mit XML-Dokumenten-Beschreibungssprachen lassen sich Meta- und Zusatzinformationen in Dokumente integrieren. Die innerhalb einer Datenbank abgelegten Relationen und Attribute können so zur logischen und physischen Strukturierung von XML-Dokumenten dienen [MAM+98]. Um die erlaubten *Elemente* eines bestimmten Dokumenttyps, insbesondere die Regelgrammatik einer Formalen Sprache, beschreiben zu können, wurde in XML das Konzept der *Dokumenttypdefinition* (DTD) eingeführt. Eine DTD beschreibt die logische (syntaktische) Struktur eines XML-Dokuments und entspricht einer kontextfreien Grammatik [ABS00]. Jedes Dokument kann rein formal anhand der Regeln der zugehörigen DTD auf Validität überprüft werden. Die logische Struktur von XML-Dokumenten ist durch *Elemente* und deren *Attribute* definiert. In der Zwischenzeit bieten viele Datenbankhersteller bereits den Export von Daten in XML und das Speichern von XML-Dokumenten innerhalb der Datenbank an. Ein Beispiel hierfür ist das XML Repository von Poet.

XML ist noch sehr jung; deshalb gibt es zur Zeit vorwiegend prototypische Implementierungen. Die Anwendungsfelder von XML werden aber vor allem in folgenden Bereichen gesehen:

- standardisierter Datenaustausch (z.B. XML/EDI)
- Verlagerung der Rechenleistung vom Server zum Client (Electronic Banking, Electronic Commerce)
- Variable Darstellung von Informationen durch Style Sheets
- Intelligentes Suchen nach Informationen in XML-Datenbeständen
- Meta-Daten für Bibliotheks- und Kataloganwendungen

Das RDF (Resource Description Framework) [LS99] soll eine Infrastruktur zur Nutzung von Meta-Daten zwischen verschiedenen Web-Anwendungen bieten und somit die Interoperabilität verbessern. RDF vereinigt verschiedene webbasierte Metadaten-Initiativen und nutzt XML als Austausch-Syntax.

In jüngster Zeit gewinnen vor allem Anfragesprachen und Views für XML an Bedeutung [Abi99], [AAA+99], [LPVV99].

2.5 Zusammenfassung

In diesem Kapitel wurden Anwendungsgebiete, Konzepte und Techniken für die Anbindung von Datenbanken an das World Wide Web vorgestellt und die jeweiligen Stärken und Schwächen analysiert. Die verschiedenen Architekturen wurden zunächst in Web-Server-seitige und Client-seitige Verfahren unterschieden. HTTP-basierte Lösungen auf Basis von CGI-Programmen oder Server-Erweiterungen bieten viele Vorteile. Durch die Wahl der geeigneten Prozeß-Architektur (einstufig, zweistufig) und Benutzeridentifikation (z.B. Cookies) lassen sich leistungsfähige Systeme entwickeln. Für Anwendungen mit besonderen Anforderungen, wie beispielsweise grafischer Interaktion oder erhöhter Sicherheit, eignen sich Java-basierte Lösungen besser. Sie bieten vielfältige Möglichkeiten zur Benutzerinteraktion und Kommunikation mit der Datenbank durch die client-seitige Datenaufbereitung.

Geht man von dem klassischen Begriff einer Datenbankanwendung aus, so steht die reine Datenspeicherung und Verwaltung im Vordergrund. Softwaresysteme mit Datenbankanwendungen bestanden lange Zeit aus separaten Komponenten zur Anwendungsentwicklung einerseits und zur Datenbankentwicklung andererseits. Die Integration dieser Komponenten wurde über proprietäre Schnittstellen realisiert. Betrachtet man die verschiedenen Architekturen und Interaktionsmechanismen zur Internet-Anbindung von Datenbanken heute, ist das eine eher veraltete Sichtweise. Heute vermischen sich die klassischen Datenbankprodukte mit integrierten Anwendungsentwicklungsumgebungen; die Trennlinie zwischen Datenbank und Anwendungsteil wird immer schwächer. Durch die Integration von Datenspeicherungskomponenten und Anwendungsentwicklungsumgebungen sollen zukünftige Internet-Lösungen nicht nur den Zugriff auf Daten sondern auch das Anwendungssystem miteinbeziehen. Die folgenden Kapitel sollen neue Forschungsansätze zu dieser Problematik aufzeigen.

Kapitel 3

Das Forschungsfeld Hypermedia

3.1 Strukturierter Hypermedia-Entwurf

Softwaretechnik wird heute als eigenständiges Fachgebiet anerkannt und geschätzt. Die Umsetzung des theoretischen Fachwissens in die praktische Anwendung ermöglicht den Entwurf hochwertiger Software, die verschiedenen Qualitätsanforderungen, wie Korrektheit, Robustheit, Erweiterbarkeit und Wiederverwendbarkeit genügt. Für den strukturierten Entwurf von Multimedia-Anwendungen im World Wide Web sind neben den Modellen des traditionellen Software-Engineering auch die Begriffe und Modelle aus dem Bereich Hypermedia relevant. Während sich der Einsatz von verschiedenen objektorientierten Entwurfsmethoden, wie z.B. OMT (Object Modeling Technique) [RBP+91], OOD (Object Oriented Design) [Boo94] und UML (Unified Modeling Language) [BRJ96] zum Entwurf von Multimedia-Anwendungen bereits bewährt hat, ist das Hypermedia-Design noch eine relativ junge Disziplin. Hypermedia-Applikationen im World Wide Web werden deshalb üblicherweise ad hoc entworfen. Häufige Folgen sind die schlechte Skalierbarkeit, Inkonsistenzen, Dangling Links und mangelhafte Strukturierung des Hypertextes (lost in hyperspace).

Um die genannten Probleme zu umgehen, wurden in der Vergangenheit verschiedene strukturierte Methoden entwickelt, die durch eine definierte Technik und Notation den Entwurf von Hypermedia-Anwendungen unterstützen. Die meisten Hypermedia-Entwurfsmethoden zeichnen sich dabei durch folgende Eigenschaften aus:

- Methodischer Design-Ansatz
- Schrittweise Vorgehensweise mit genauen Instruktionen
- Präsentation der Entwurfsschritte durch verschiedene semantische Modelle
- Kombination der einzelnen Komponenten bildet Gesamt-Applikation
- Unterstützung des Design-Prozesses durch verschiedene Tools

Die nützliche Abstraktion von Teilbereichen der realen Welt und die Modellierung der Semantik großer Informationsbestände mit Hilfe von semantischen Datenmodellen führt ähnlich dem DB-Design zu verschiedenen Vorteilen, wie z.B. der besseren Skalierbarkeit und Wiederverwendbarkeit.

In der Vergangenheit wurde eine Anzahl unterschiedlicher Forschungsarbeiten auf diesem Gebiet durchgeführt. Zu den bekanntesten Hypermedia-Entwurfsmethoden gehören HDM (Hypermedia Design Model) [GPS93], OOHDM (Object Oriented Hypermedia Design Method) [SR95], RMM (Relationship Management Methodology) [IRB95], W3DT (World Wide Web Design Technique) [BN96] und WSDM (Web Site Design Method) [Tro98]. Im folgenden sollen die Anwendungsgebiete, Grundlagen des Entwurfsprozesses und einige ausgewählte Entwurfsmethoden zur Modellierung von Hypermedia-Anwendungen vorgestellt werden.

3.1.1 Anwendungsfelder

Die meisten Methoden zum Entwurf von Hypermedia-Anwendungen wurden zunächst nicht speziell für das Web entwickelt, haben sich jedoch an diese Problematik angepaßt und werden häufig für die Entwicklung von Web-Anwendungen eingesetzt. Im Verlauf dieser Arbeit werden deshalb vornehmlich Beispiele aus dem World Wide Web, als eines der bekanntesten Hypertext-Systeme, angeführt. Der Entwurf von Hypertext-Applikationen gliedert sich in zwei Aufgabenbereiche: die Modellierung von Struktur, Navigation und Layout, und die Instantiierung der Hypertext-Knoten mit Inhalten, die z.B. aus einer Datenbank generiert werden.

Der Design-Vorgang ist dabei sehr eng verwandt mit dem Datenbank-Design und basiert zum Teil auf den gleichen Datenmodellen. Semantische Datenmodelle bilden ähnlich zum Datenbankentwurf die Schnittstelle zwischen dem Anwendungsentwickler und dem Spezialisten des Problembereichs. Ein logisches Datenmodell, das von der physischen Struktur getrennt verwaltet wird, existiert bei Hypertext-Entwurfsmethoden dagegen nicht.

> *The need for a development methodology is present, whether the system to be established is a Home page window into the World Wide Web or an internal information system operating on a Local Area Network.* [IRB95]

Obgleich die Notwendigkeit für eine Hypermedia-Entwurfsmethode außer Frage steht, eignet sich ein strukturierter Entwurfsprozeß nicht für alle Hypermedia-Applikationen gleichermaßen. Vor allem Applikationen mit großen, strukturierten Datenmengen, die sich über die Zeit schnell ändern sind typische Anwendungen für eine Hypermedia-Design-Methode und bringen deren Vorteile langfristig zur Geltung.

3.1.2 Grundlagen des Entwurfsprozesses

Während man im traditionellen Software-Lebenszyklus eine relativ starre Abfolge der Entwurfsschritte vorsieht (Benington 1956), zeichnet sich der Hypermedia-Designprozeß vor allem durch seinen iterativen und inkrementellen Charakter aus. Der verstärkte Einsatz von multimedialen Elementen, die unterschiedlichen Fachkenntnisse der Mitarbeiter innerhalb des Projektteams und die starke Gewichtung des Oberflächendesigns machen ein frühes Prototyping bei Anwendungen im Web sehr wichtig. Bei dem Entwurf von Hypermedia-Anwendungen ist weiterhin, ähnlich dem objektorientierten Entwurf, die Modellierung verschiedener Benutzergruppen und Sichten, Zustände und Ereignisse und schließlich der verteilten Client-Server-Architektur wichtig.

Die Anzahl, Abfolge und die Inhalte der einzelnen Entwurfsschritte des Hypermedia-Designs differieren in der Literatur stark. Um die Grundlagen des Hypermedia-Entwurfsprozesses dennoch einheitlich darzustellen, wurden die Phasen des traditionellen SW-Entwicklungszyklus als Basis verwendet und hinsichtlich des Hypermedia-Designs erweitert. Eine detaillierte Erläuterung der unterschiedlichen Entwurfsmodelle und -prozesse erfolgt im nächsten Kapitel.

Analysephase

Im Verlauf der Analysephase wird eine Hypertext-Applikation aus verschiedenen Perspektiven betrachtet und beschrieben [GMP95]:

- **Inhalte:** Genaue Analyse der Anwendungsdomäne mit Spezialisten aus den jeweiligen Fachgebieten. Die in der Anwendung beinhalteten Informationen können sich aus statischen und dynamischen Teilen zusammensetzen.
- **Struktur:** Organisation des Inhaltes in HTML-Seiten.
- **Präsentation:** Darstellung der Inhalte; Modellierung der graphischen Benutzeroberfläche.
- **Dynamik:** Beschreibung der Art und Weise, wie der Anwender mit Teilen von Informationen interagieren und zwischen unterschiedlichen Medien wechseln kann.
- **Interaktion:** Grad der Benutzerinteraktion: statische (Text, Bilder) oder aktive (Videoclips, Klang) Informationsteile, wie z.B. Videoclips, Klang usw.

Weiterhin erfolgt im Verlauf der Analysephase auch die Klassifikation und Modellierung der einzelnen Benutzergruppen [Tro98].

Konzeptuelles Design

Im Verlauf des konzeptuellen Designs erfolgt die Modellierung der Anwendungsdomäne, auf deren Basis die Modellierung der Hypertext-Struktur und Navigation durchgeführt

wird. HTML-Dokumente haben sowohl eine Intra-Dokument-Struktur (Anordnung der Informationen auf einer HTML-Seite) als auch eine Inter-Dokument-Struktur (Navigation zwischen Dokumenten via Hyperlinks). Herkömmliche semantische Datenmodelle, wie beispielsweise das ER-Modell [Che76], reichen für die Modellierung dieser Informationen nicht mehr aus. Deshalb werden für das Design der Struktur von Hypertext und der Navigation zwischen Hypertext-Dokumenten spezifische semantische Modelle zur Verfügung gestellt. Das konzeptuelle Hypermedia-Design gliedert sich in folgende drei Phasen:

- **Modellierung des Anwendungsbereichs auf der Basis bekannter Datenmodelle:** Modellierung der Anwendungsdomäne als Teilbereich der realen Welt und Abbildung auf ein semantisches Datenmodell (z.B. ER-Modell, OMT).
- **Festlegen der Hypertext-Struktur:** Klare und sinnvolle Strukturierung der Entities und Attribute in Hypertext-Knoten.
- **Navigational Design:** Entwicklung einer konsistenten Navigationsstruktur durch die Hypertext-Knoten (Vor- und Zurückblättern, intuitive Bedienung). Die wichtigsten Zugriffsstrukturen sind hierbei: (Conditional) Guided Tour, (Conditional) Index und (Conditional) Indexed Guided Tour.

Bei dem Zugriff auf eine Menge gleich strukturierter HTML-Seiten werden im Web verschiedene Zugriffsstrukturen unterschieden. Bei einem *Index* greift der Anwender über eine Liste von Hyperlinks auf die einzelnen HTML-Seiten zu. Eine *Guided Tour* leitet den Anwender durch eine verkettete Liste von HTML-Seite zu HTML-Seite. Eine *Indexed Guided Tour* verbindet beide Zugriffsstrukturen (siehe Abbildung 3.1).

Implementierung

Im Verlauf der Implementierungsphase erfolgt das Design des Layouts und die Implementierung des konzeptuellen Schemas. Da es auf diesem Gebiet umfangreiche Literatur gibt, soll an dieser Stelle nur auf einige der wichtigsten Kriterien für die Erstellung von Web-Sites eingegangen werden:

- Übersichtliches Design der Benutzeroberfläche (Bedienelemente, Farben, Schrift)
- Klare und sinnvolle Anordung der Informationen innerhalb einer HTML-Seite
- Intuitive Bedienbarkeit durch einheitliche Navigationshilfen

Die Implementierung erfolgt entweder manuell duch die Umsetzung des semantischen Modells auf HTML-Seiten oder mit Hilfe von CASE-Werkzeugen, die die einfache Generierung und Wartung der HTML-Seiten ermöglichen. Die Ablage der Informationen kann dabei entweder auf der Basis von HTML-Dateien innerhalb des File-Systems oder auf der Basis einer Datenbank mit Hilfe von HTML-Templates erfolgen.

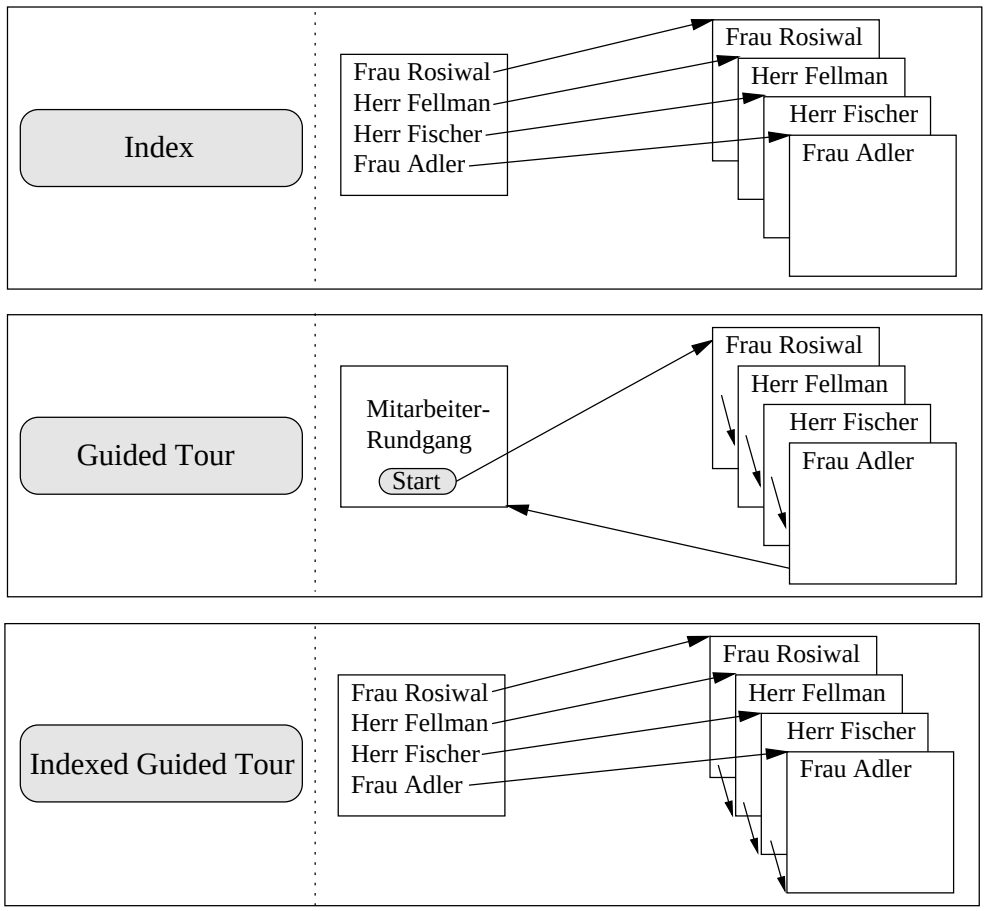


Abbildung 3.1: Zugriffsstrukturen im Web

Evaluierung

Am Ende des Software-Entwurfs steht die Evaluierungsphase. Auch hier sind deutliche Unterschiede zum herkömmlichen SW-Entwurf erkennbar. Ganz andere Kriterien spielen eine wichtige Rolle. Interessante Fragen sind beispielsweise, wie man ein adäquates Design mit verträglichen Ladezeiten im Internet ermöglicht. Garzotto, Mainetti und Paolini [GMP95] legten für die Evaluierungsphase im Hypermedia-Entwurf vor allem folgende Kriterien fest:

- **Reichhaltigkeit:** Die Fülle von Informationen und die Möglichkeiten, sie zu erreichen.
- **Einfachheit:** Einfache Bedienbarkeit (Durchführung von Operationen, Zugriff auf Informationen).
- **Konsistenz:** Konzeptuell gleiche Elemente sollten gleich, konzeptuell verschiedene Elemente sollten verschieden dargestellt werden.
- **Eindeutigkeit:** Intuitive Bedienung durch den Anwender. Bedeutung und Zweck der dargestellten Information sollte selbsterklärend sein.

- **Vorhersehbarkeit**: Die Resultate der verschiedenen Operationen sollten vorhersehbar sein.
- **Lesbarkeit:** Allgemeine Akzeptanz beim Anwender.
- **Wiederverwendbarkeit:** Klare Trennung der verschiedenen Ebenen, durch Konsistenz und Vorhersehbarkeit unterstützt.

3.1.3 Entwurfsmethoden und Hypermedia-Modelle

In der Vergangenheit wurden verschiedene Datenmodelle für *Hypertext-Systeme* entworfen, wie beispielsweise das Dexter Referenz Modell [HS94], HB1 [SLHS93] , HAM [CG88] und das Konstanzer Hypertext-System [Ass96]. Diese Hypertext-Systeme bilden die Umgebung von *Hypermedia-Applikationen.*

Für den Entwurf dieser Hypermedia-Anwendungen existieren verschiedene Design-Methoden und Modelle. Zu den bekanntesten Entwurfsmethoden gehören: HDM [GPS93], OOHDM [SR95], RMM [IKK97], W3DT [BN96] und WSDM [Tro98]. Eine Entwurfsmethode bietet neben einem definierten Entwurfsprozeß und geeigneten Datenmodellen auch Richtlinien zur genauen Vorgehensweise. Dazu gehören u.a. eine genaue Beschreibung der Inhalte einzelner Phasen, Mappingregeln und Design-Hinweise. Nach [CSP98] sollen Hypermedia Application Development and Management Systeme folgendes bieten:

- ein abstraktes Hypermedia Modell, das die Organisation und Strukturierung des Informations-Materials erlaubt und im Hinblick auf zukünftige 'Hypertext Features' leicht erweiterbar ist.
- die Möglichkeit die Informationsinhalte des Modells in statische oder dynamische Hypertext-Applikationen zu konvertieren.
- die einfache Integration anderer Datenbestände und die Wiederverwendbarkeit von existierenden Applikationen.
- die Möglichkeit, die gleichen Informationsinhalte für Hypertext-Applikationen in verschiedenen Hypertext-Systemen (wie beispielsweise WWW, Hyperwave) zu generieren.

Im folgenden sollen drei der bekanntesten Hypertext-Enturfsmethoden und -Modelle vorgestellt werden.

HDM

Als erstes sei das von Garzotto, Paolini und Schwabe entwickelte Hypermedia Design Model (HDM) [GPS93] vorgestellt. Viele bekannten Entwurfsmethoden, wie die später genauer erläuterte RMM und OOHDM stützen sich auf HDM ab. Die wichtigsten Konzepte von HDM sind:

- **Entities und Entity-Typen:** Entsprechen in ihren Grundzügen den Entities, die aus dem ER-Modell bekannt sind. Entities im HDM unterscheiden sich allerdings durch eine komplexe innere Struktur, beinhaltete Browsing-Semantik und Application-Links (anstelle von Relationships) zu anderen Entities.
- **Component:** Ein HDM-Entity besteht aus einer Menge von sogenannten Components, die hierarchisch (baumartig) angeordnet sind. Components können nur als Bestandteil eines Entities existieren. Jedes Component ist eine Abstraktion für eine Menge von sogenannten ´Units´, die die eigentlichen Informationen beinhalten.
- **Perspektiven:** Perspektiven bieten die Möglichkeit den gleichen Inhalt eines Entities auf unterschiedliche Art und Weise zu präsentieren. Ein Beispiel hierfür wären multilinguale Web-Sites: der Inhalt eines Entities wird in verschiedenen Sprachen dargestellt.
- **Units:** Jede Unit gehört zu einem Component. Jede Unit besteht aus einem Namen und einem Body, der die eigentlichen Informationen beinhaltet. Eine Unit entspricht im Wesentlichen einem Hypertext-Knoten.
- **Links:** Links werden in strukturelle Links (zwischen Komponenten) und Application-Links (zwischen Entities) unterschieden.

Anhand eines Beispiels soll die Unterscheidung zwischen Entities, Components und Units deutlich gemacht werden. Ein mögliches Entity eines Entity-Typs ´Oper´ wäre beispielsweise ´La Traviata´. Zugehörige Components wären beispielsweise 'Beschreibung', ´Ouvertüre´ und 'Akt'. Mögliche Perspektiven für die Beschreibung einer Oper wären ´Sound´ und ´Partitur´. Die Units der Component ´Ouvertüre´ wären in diesem Fall ´Ouvertüre.Partitur´ und ´Ouvertüre.Sound´ (siehe Abbildung 3.2).

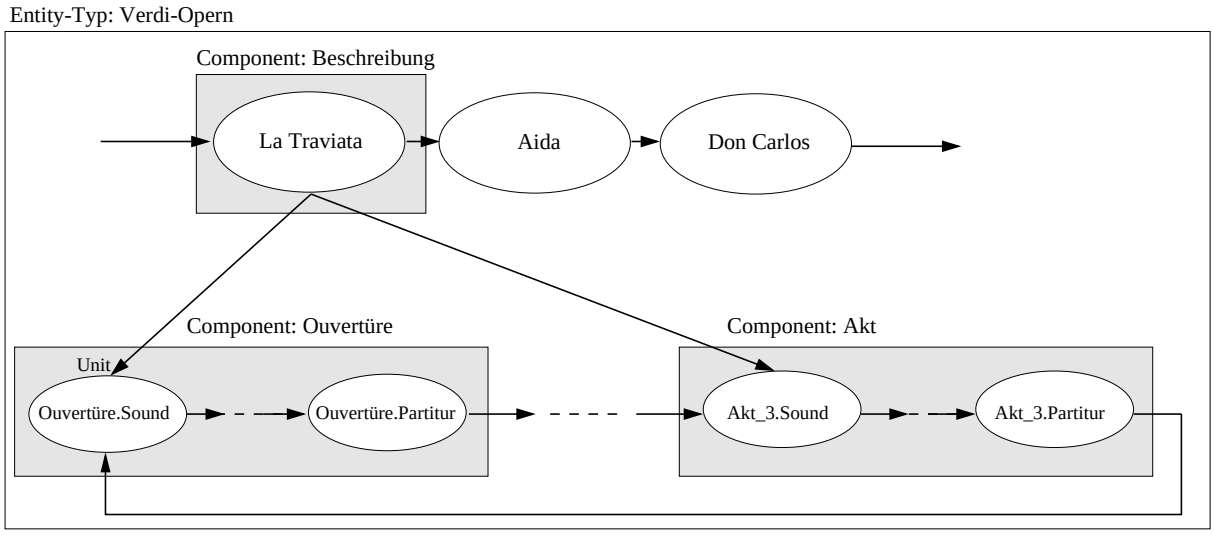


Abbildung 3.2: Beispiel zu den Modellierungskonzepten der HDM

RMM

Zu einer der bekanntesten Entwurfsmethoden gehört die von Isakowitz et. al. entwickelte strukturierte Hypermedia-Entwurfsmethode RMM (Relationship Management Methodology) [IRB95], die auf dem HDM-Datenmodell und dem ER-Modell basiert. Die Basis-RMM-Technologie ist in [IKK97] um das Konzept der M-Slices erweitert worden, die eine Gruppierung von Informationen innerhalb von Entitäten und die rekursive Schachtelung von Informationseinheiten ermöglichen. Da die RMM als Grundlage für die im Rahmen dieser Arbeit vorgestellte HDBM dient, erfolgt in Kapitel 5.3 eine ausführliche Beschreibung der Modellierungskonzepte der RMM. An dieser Stelle sollen nur die Basiskonzepte erläutert werden. Das RMM-Datenmodell (RMDM) besteht aus folgenden drei Arten von Elementen:

- **ER Domain Primitives:** Modellierung der Anwendungsdomäne auf der Basis eines vereinfachten ER-Modells unter Verwendung von Entities, Attributen, 1:1-Beziehungen und 1:N -Beziehungen zwischen den Entities. M:N-Beziehungen werden auf zwei 1:N-Beziehungen abgebildet.
- **RMDM Domain Primitives:** Gruppierung von Attributen aus Entities in sogenannte M-Slices. Slices bilden das wichtigste Modellierungskonzept der RMM und entsprechen einer Gruppierung von logisch zusammengehörigen Attributen eines Entities. Jeder Slice kann rekursiv aus anderen Slices aufgebaut sein. Slices, die als HTML-Seiten angezeigt werden, bezeichnet man als Presentation Units.
- **Access Primitives:** Navigation zwischen Slices und Entities durch verschiedene Zugriffsarten: Uni-Directional Link, Bi-Directional Link, Conditional Index, Conditional Guided Tour, Conditional Indexed Guided Tour.

Neben dem Modell zeichnet sich die RMM-Methode durch eine strukturierte Vorgehensweise aus, die in verschiedene Entwurfsschritte gegliedert ist.

1. **ER-Design:** Modellierung der Anwendungsdomäne mit Entities und Beziehungen zwischen diesen.
2. **Slice-Design:** Modellierung der Slices.
3. **Navigational Design:** Modellierung der Navigationsstruktur mit Startpunkten, Rundgängen und anderen Zugriffsstrukturen.
4. **Conversion Protocol Design:** Aufteilung der verschiedenen Komponenten auf dem Web-Server (Datenbanken, Browser, HTML-Formate).
5. **User Interface Design:** Modellierung des Layouts der HTML-Seiten.
6. **Runtime Behavior Design:** Programme, die ausgeführt werden sollen, werden in die Seiten eingebunden.

7. **Construction and Testing:** Erzeugung der Seiten und Testphase.

Neben den genauen Inhalten der einzelnen Phasen werden auch Richtlinien und Mapping-Regeln in der RMM beschrieben. Die RMM bietet ebenfalls das RMCase Werkzeug [DI95] zur Entwicklung und Generierung von Hypermedia-Applikationen.

OOHDM

Die dritte hier behandelte Entwurfsmethode ist die von Schwabe, Rossi und Barbosa entwickelte Object Oriented Hypermedia Design Method (OOHDM)[SR95]. Im Unterschied zu den bereits besprochenen Methoden verfolgt die OOHDM einen objektorientierten Ansatz und zeichnet sich durch folgende Entwurfsphasen aus:

- **Konzeptuelles Design:** Modellierung der Anwendungsdomäne unter Verwendung objektorientierter Modellierungsprinzipien auf der Basis der OMT.
- **Navigational Design:** Modellierung einer Hypertext-Sicht über dem konzeptuellen Schema. Das Navigational Design wird durch zwei Schemata ausgedrückt: das *navigational class schema* und das *navigational context schema*. Das navigational class schema modelliert Hypertext-Knoten, Links und Zugriffsstrukturen. Im Verlauf des navigational context designs werden die Hypertext-Knoten in Abhängigkeit von dem jeweiligen Kontext, in dem sie angezeigt werden sollen, modelliert.
- **Abstract Interface Design:** Modellierung der Benutzeroberfläche mit Hilfe des Abstract Data View Designs (ADV).
- **Implementierung:** Abbildung von Navigations- und Interface-Objekten auf physische Objekte.

Im Verlauf des *Navigational Designs* gibt es drei Typen von Klassen: Knoten, Links und Zugriffsstrukturen. Knoten bilden das Basiskonstrukt bei Hypermedia-Applikationen. Sie beinhalten einwertige (single typed) Attribute und Link-Anker und können atomar oder geschachtelt sein. Link-Klassen werden definiert durch Link-Attribute, Verhalten, Quelle, Ziel und Kardinalität. Zugriffsstrukturen sind beispielsweise Menü, Index und Guided Tours. Zugriffsstrukturen werden durch eine Menge von Selektoren, eine Menge von Ziel-Objekten und ein Prädikat auf den Zielobjekten definiert. Selektoren stehen für einige Attribute des Zielobjekts, die entsprechend einer vordefinierten Datenstruktur (z.B. einer geordneten Liste) organisiert sind.

Abhängig von dem jeweiligen Kontext ist es möglich, Hypertext-Knoten unterschiedlich anzuzeigen. Während des *navigational context designs* werden die Klassen um zusätzliche Informationen ergänzt, die Aufschluß darüber geben, welche Information bzw. welcher Anker in welchem Kontext wie angezeigt werden soll.

Ein Vorteil von der OOHDM ist vor allem die klare Trennung zwischen dem Interface- und dem Navigational Design. Das Interface ist so vollkommen unabhängig von anderen Ebenen wartbar.

3.2 Ausgewählte Projekte

Das starke Wachstum des Internets und die zunehmende Nutzung für Business-Anwendungen hat die Anforderungen an den Zugriff auf Web-Daten und die Verwaltung von Web-Daten nachhaltig verändert. Die Einschränkungen von aktuellen Suchmaschinen und die mangelnde Strukturierung und Inkonsistenz von Daten im Web führten zu der Entwicklung neuer logischer Web-Datenmodelle, Anfragesprachen und Web-Site-Management-Systemen. Hierbei spielen Konzepte aus dem Datenbankbereich eine immer wichtigere Rolle. Die Forschungsarbeiten zu den unterschiedlichen Bereichen sollen im folgenden kurz erläutert werden.

3.2.1 Datenbankbasierte Verwaltung von Web-Sites

Die im vergangenen Kapitel vorgestellten Hypermedia-Entwurfsmethoden bilden die Basis für verschiedene Forschungsprojekte zum Entwurf und zur Verwaltung von Web-Sites auf der Basis von Datenbanken. Zwei Projekte in diesem Bereich sind ARANEUS und AUTOWEB.

ARANEUS

Im Rahmen des ARANEUS-Projekts [AMM97b] wurden ein Datenmodell und darauf basierend verschiedene Sprachen zur Modellierung, Verwaltung und Restrukturierung von Daten aus dem World Wide Web entwickelt. Die ARANEUS-Design-Methode bietet neben dem konzeptuellen Hypertext-Design, das auf der RMM basiert, auch ein logisches Hypertextmodell. Das seitenorientierte ARANEUS-Datenmodell (ADM) basiert auf dem ODMG-Objektmodell (Object Database Management Group). Jede HTML-Seite wird als ein Objekt modelliert, das einen eindeutigen Identifikator (URL) besitzt und durch eine Menge von Attributen beschrieben wird. Die Struktur von HTML-Seiten wird durch Seiten-Schemata modelliert, deren Instanzen HTML-Seiten sind. Attribute können einfache oder komplexe Web-Typen haben. Einfache Typen sind einwertig und beispielsweise Bilder, Texte oder Links zu anderen Seiten. Komplexe Typen modellieren Kollektionen von Objekten und entsprechen Listen von Tupeln, die möglicherweise geschachtelt sind. Neben Tupeln und Listen werden keine Mengen unterschieden. Folgende Typen sind Web-Typen:

- jeder Basis-Typ (z.B. Integer, Char) ist ein Web-Typ;
- Link to P ist ein einwertiger Web-Typ, für jedes Seitenschema P;
- List of$(A_1 : T_1, A_2 : T_2, \ldots A_n : T_n)$ ist ein mehrwertiger Web-Typ, wenn $A_1, A_2, \ldots A_n$ Attribute und $T_1, T_2, \ldots T_n$ Web-Typen sind;

Ein Seitenschema hat die Form $P(URL, A_1 : T_1, A_2 : T_2, \ldots A_n : T_n)$, wobei P der Name der Seite, A_i ein Attribut, T_i ein Web-Typ und URL der Uniform Resource Locator von P

und gleichzeitig der Schlüssel von P ist. Basierend auf dem ADM-Modell unterstützen die zwei Sprachen ULIXES und PENELOPE die Restrukturierung von Web-Sites. ULIXES ermöglicht die Extraktion von Daten aus dem Web in relationale Views. Die Grundlage hierfür bildet das Wissen über die mit Hilfe der ADM modellierten Struktur der Web-Site. PENELOPE ermöglicht in die Rückrichtung die Generierung von Hypertext aus einer Datenquelle und verwendet ebenfalls ADM, um die Struktur der generierten Web-Site zu beschreiben.

Der Ansatz von ARANEUS eignet sich, um bestehende Web-Sites mit Hilfe eines Hypermedia-Modells zu strukturieren, die Daten anschließend aus dem Web zu extrahieren und in eine Datenbank einzuspoolen. Innerhalb der Datenbank können die HTML-Seiten restrukturiert und anschließend wieder als HTML-Seiten generiert werden. ARANEUS läßt sich allerdings nur dann sinnvoll einsetzen, wenn der jeweiligen Web-Site eine klare Struktur zugrunde liegt.

AUTOWEB

Das Autoweb-System [FP98] unterstützt den Entwurf und die Generierung großer Web-Sites auf der Basis eines relationalen Datenbanksystems. Als konzeptuelles Hypermedia-Modell wird HDM-lite, das auf HDM aufsetzt, verwendet. HDM-lite unterstützt die Modellierung von Struktur, Navigation und Präsentation einer Applikation. Dabei werden drei Schemata, das *hyperbase schema*, *access schema* und *presentation schema* unterschieden. Mit Hilfe des CASE-Tools Visual HDM erfolgt der Entwurf der Web-Site und die Abbildung auf ein relationales Schema. Dabei werden die konzeptuellen Schemata in eine logische Präsentation überführt. Der Autoweb Page Generator erzeugt auf der Basis der Datenbank dynamisch generierte HTML-Seiten. Der Autoweb-Ansatz ist sowohl für die Entwicklung neuer Anwendungen als auch für 'reverse engineering' existierender Anwendungen geeignet, die auf einer relationalen Präsentation der Daten beruhen.

3.2.2 Integration heterogener Datenquellen

Eine Problematik im Internet ist die Integration verschiedener heterogener Datenquellen. Für die Integration werden teilweise wissensbasierte Ansätze und Konzepte aus der Datenbanktechnologie eingesetzt. Dabei wird üblicherweise eine zweistufige Architektur eingesetzt: ein Mediator übernimmt die Verteilung von Anfragen auf mehrere Quellen bzw. die Kombination verschiedener Antworten. Er verwendet hierfür Wrapper, die einzelne Datenquellen kapseln und für den Mediator transparent machen. Zu den bekanntesten Ansätzen in diesem Bereich gehören STRUDEL, TSIMMIS, Information Manifold und SIMS.

STRUDEL

STRUDEL [FFK+97] ermöglicht die einheitliche Modellierung heterogener Datenquellen auf der Basis eines semistrukturierten Datenmodells (markierter gerichteter Graph) und gewährleistet die Trennung der logischen Struktur einer Web-Site von der physischen Ablage der Daten. Insgesamt werden in STRUDEL die drei Ebenen Graphical Representation, Structure Management und Data Management unterschieden (siehe Abbildung 3.3). In der Data Management Ebene werden verschiedene heterogene Datenquellen in ein einziges semistrukturiertes Daten Repository integriert. Als Basis dient ein Graph-Modell für semistrukturierte Daten. Die Datenintegration erfolgt in STRUDEL standardmäßig mit Hilfe von Mediatoren und Wrappern, die die Daten von dem logischen Modell der Datenquelle in das logische STRUDEL-Modell übersetzen. In der zweiten Ebene wird die Struktur der Web-Site durch einen Site-Graphen beschrieben. Knoten entsprechen HTML-Dokumenten und Kanten entsprechen Hyperlinks. Die Konstruktion des Site-Graphen erfolgt mit Hilfe der deklarativen Anfragesprache StruQL. Die dritte Ebene definiert die graphische Repräsentation mit Hilfe von HTML-Templates. STRUDEL kann nicht für die Manipulation von Daten verwendet werden sondern ist für read-only Quellen geeignet.

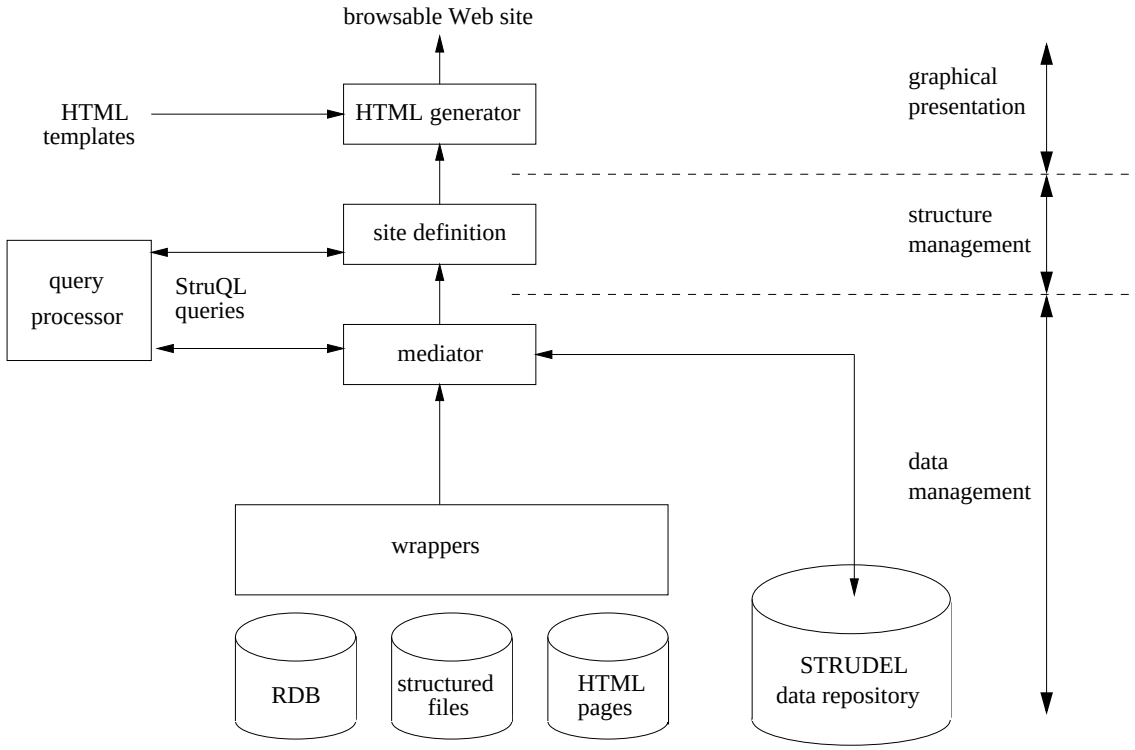


Abbildung 3.3: Architektur von STRUDEL nach [ABS00]

TSIMMIS

Ziel des an der Stanford University entwickelten TSIMMIS-Systems (**St**anford **I**BM **M**anager of **M**ultiple **I**nformation **S**ources) [CGMH+94] ist die Integration verschiedener he-

terogener Datenquellen. Dabei können die Datenquellen sowohl strukturierte als auch unstrukturierte Daten beinhalten. Die Systemarchitektur von TSIMMIS basiert auf Mediatoren, Wrappern und Übersetzern. Zur Beschreibung der heterogenen Datenquellen wird als einheitliches Datenmodell das im TSIMMIS Projekt entwickelte OEM (Object Exchange Model) eingesetzt. Jedes OEM-Objekt wird durch einen Object Identificator `OID`, `type`, `value` und einen `label` beschrieben. Verwendete Sprachen in TSIMMIS sind die Lightweight Object Repository Language (LOREL), die Wrapper Specification Language (WSL) und die Mediator Specification Language (MSL).

Infomaster System

Das Infomaster System [DG97] wurde an der Stanford University entwickelt und dient zur Integration verschiedener heterogener Datenbanken vorwiegend im Internet. Dabei werden verschiedene Abstraktionshierarchien unterschieden, die durch Relationen beschrieben werden. Die *Interface-Relationen* ermöglichen verschiedene Sichtweisen auf das Informationssystem und bilden die Schnittstelle zum Endanwender. Anfragen an Interface-Relationen werden auf *Base-Relationen* abgebildet. Diese wiederum werden mit Hilfe von Constraints und Information Source Descriptions auf die eigentlichen *Site-Relationen* der einzelnen Datenquellen abgebildet. Die heterogenen Informationsquellen werden dabei durch das Knowledge Interchange Format (KIF) beschrieben.

SIMS

Der Information-Mediator SIMS [ACHK93] wurde an der University of Southern California entwickelt und dient zur Übersetzung von Anfragen an heterogene Datenquellen. Die technologische Infrastruktur des SIMS-Systems setzt sich dabei aus LOOM, LIM und einem Planer Prodigy zusammen. LOOM bezeichnet die Wissensbasis zur Beschreibung und Integration der unterschiedlichen Modelle und Inhalte der heterogenen Datenbanken; LIM (LOOM Interface Module) bildet die Schnittstelle zwischen LOOM und den eigentlichen Datenquellen. LOOM-Anfragen werden durch LIM in native Anfragen an die betreffende Datenquelle übersetzt. Prodigy wählt die geeigneten Informationsquellen aus und legt die Reihenfolge der Anfragen fest. Weiterhin bietet SIMS einen Query-Plan-Optimierer und eine graphische Benutzeroberfläche.

3.2.3 Web-Anfragesprachen

Für das große und schnell wachsende Informationsangebot im WWW müssen Werkzeuge und Techniken zur Verfügung gestellt werden, die eine effiziente und zielgerichtete Suche über den riesigen Datenbestand ermöglichen. Halasz [Hal88] schrieb 1988 hierzu:

> *Navigational access itself is not sufficient. Effective Access to Information stored in a hypermedia network requires query-based access to complement navigation.*

Zur Informationsauffindung im WWW werden heute meist Suchmaschinen (Search Engines) verwendet (Altavista, Yahoo, Webcrawler). Daneben wurden verschiedene Techniken des Formulierens und Verarbeitens von Anfragen, wie man sie von Datenbanken kennt, auf das WWW angewendet. Das Ziel ist, jeweils bekannte, meist deklarative Anfragesprachen, um WWW-spezifische Kriterien zu erweitern und auf das Web als große verteilte Datenbank anzuwenden. Der Anwender kann mit Hilfe einer Anfragesprache sowohl nach der Struktur einzelner Dokumente als auch nach der Lokalisierung eines Dokuments innerhalb des Netzwerkes suchen. Man unterscheidet in diesem Zusammenhang *content queries*, die über den Inhalt einzelner HTML-Knoten abgesetzt werden, von *structure-specifying queries*, in denen spezielle Navigationsstrukturen gesucht werden. Als Grundlage werden die Netzwerk-Topologie und einige wenige Eigenschaften der Dokumente üblicherweise mit Hilfe eines virtuellen Graphen modelliert. Mit der Anfragesprache können reguläre Pfadausdrücke über den Graphen formuliert werden. Eine genaue Modellierung der Eigenschaften einzelner Dokumente ist aufgrund der fehlenden Struktur nicht möglich.

W3QL

W3QS (World Wide Web Query System) und die zugehörige Anfragesprache W3QL (World Wide Web Query Language) wurden von Konopnicki und Shmueli [KS95] entwickelt. W3QL ermöglicht sowohl Anfragen über den Inhalt einzelner Hypertext-Knoten (content queries) als auch über die Struktur des Hypertextes (structure-specifying queries). Das Web wird dabei als gerichteter Graph betrachtet, in dem die einzelnen HTML-Seiten die Knoten und die Hyperlinks die Kanten darstellen. Durch eine deklarative, SQL-ähnliche Anfrage wird ein Teilgraph beschrieben, der durch die Suche gefunden werden soll. Die Strukturelemente (z.B. HTML-Tags) der einzelnen meist semistrukturierten Dateien werden als Suchkriterien innerhalb einer Seite verwendet. In der SELECT-Klausel können beliebige Textmarkierungen und -eigenschaften abgefragt werden. In der From-Klausel wird der Graph selbst beschrieben. In der Where-Klausel werden die zu erfüllenden Eigenschaften der Knoten festgelegt. Als Ergebnis erhält der Anwender eine URL-Liste.

WebSQL

In dem an der University of Toronto entwickelten WebSQL [MMM96] wird dem Web ein relationales Datenbankschema mit zwei virtuellen Relationen zugrunde gelegt:

- Document(url, title, text, length, modif) zur Beschreibung der HTML-Knoten
- Anchor(base, label, ref) zur Beschreibung der Navigation zwischen Knoten, wobei 'base' und 'ref' die URLs von Quell- und Zieldokument sind.

Mit einer SQL-ähnlichen Sprache können an diese beiden Relationen Anfragen formuliert werden. Die inhaltliche Suche nach Dokumenten ist mit Hilfe der Attribute der

Relation *Document* möglich. Mit Hilfe eines Start-URL und der Angabe einer Pfadnotation können *structure-specifying queries* abgesetzt werden.

WebLog

Eine weitere Web-Anfragesprache ist die von Lakshmanan et al. [LSS96] entwickelte logikbasierte Sprache WebLog. Auch hier wird die Struktur der HTML-Seiten als Suchkriterium genutzt. Dabei werden die HTML-Tags verwendet, um das Dokument in sinntragende Einheiten zu untergliedern. Ausgehend von einem Start-Dokument wird bestimmt, welche Eigenschaften die erreichbaren HTML-Dokumente bzw. wegführenden Links haben sollen. Zur Extraktion von Informationen aus einzelnen HTML-Seiten werden Build-In Prädikate und atomare Formeln zur Verfügung gestellt. Das Ergebnis wird anders als bei W3QL nicht als URL-Liste, sondern als HTML-Seite mit den relevanten Informationen bereitgestellt.

3.3 Zusammenfassung

Das Spektrum zur Modellierung von Hypermedia-Informationen ist weit. Das enorme Wachstum des Internet hat die Art und Weise, wie Informationen modelliert und verwaltet werden nachhaltig beeinflußt. In der Vergangenheit wurden vielfältige Entwurfsmethoden entwickelt, die den strukturierten Entwurf von Web-Sites und Web-Applikationen unterstützen. Die Entwurfsmethoden basieren dabei auf bekannten semantischen Modellen, vorwiegend aus dem Datenbankbereich, die um spezifische Hypertext-Konzepte erweitert wurden. Die Wahl der richtigen Entwicklungsumgebung fällt schwer da es keine formale Methode gibt, die über die Qualität einer Modellierungstechnik Aufschluß geben würde [NN95]:

> *...to evaluate an application's structure no valid theory or technique currently exists to assess formally whether a conceptual model of a hypertext is clear to readers.*

Die im vergangenen Kapitel vorgestellten konzeptuellen Datenmodelle dienen häufig als Grundlage für die Entwicklung von Hypermedia Anwendungsentwicklungs- und Management- Systemen, die nach [CSP98] entlang folgender drei Kategorien evaluiert werden können

- **Methodik:** eingesetzte Hypermedia-Entwurfsmethode, Modelle und Abfolge der Entwurfsphasen.
- **Entwicklungsumgebung:** Entwicklungsumgebung, die den Entwurfsprozeß und insbesondere die Phasen Implementierung, Testen und Wartung durch entsprechende Tools unterstützt.

- **Plattform:** Plattformen auf denen die Hypermedia-Management-Systeme verfügbar sind.

Im vergangenen Kapitel wurden einige ausgewählte Projekte vorgestellt, in denen verschiedene Methoden zur Modellierung, Anfragebearbeitung und Verwaltung von Hypermedia-Daten zum Einsatz kommen.

Kapitel 4

Das abayfor-online-Projekt

Im Rahmen des abayfor-online-Projektes[1] wurde am FORWISS ein datenbankbasiertes Informationssystem entwickelt, das via WWW Informationen über die Arbeitsgemeinschaft der bayerischen Forschungsverbünde (abayfor) bereitstellt. Anforderungen und Hintergrund des abayfor-Projektes dienten als Motivation und Grundlage für die Entwicklung der innerhalb dieser Arbeit vorgestellten HDBM (Hypermedia Database Methodology). Im folgenden Kapitel sollen deshalb neben den Anforderungen, der Systemarchitektur und der Modellierung der abayfor-Datenbank auch die Limitierungen des verwendeten Ansatzes vorgestellt werden.

4.1 Zielsetzung und Anforderungen

Abayfor ist eine eigenständige Institution zur Verwirklichung der Ziele und für die Bewältigung gemeinsamer Aufgaben der bayerischen Forschungsverbünde. Derzeit besteht abayfor aus 21 Forschungsverbünden, in denen Wissenschaftler zu verschiedenen Forschungsschwerpunkten aus mehreren Universitäten zusammenarbeiten. Forschung wird so durch Bündelung des Forschungspotentials über Universitätsgrenzen hinaus organisiert. Die Verbünde in abayfor nutzen gegenseitige Fachkompetenz und Erfahrung. Während der Existenzdauer der befristet angelegten Verbünde wird eine dauerhafte Verankerung ihres jeweiligen Themas in Forschung und Lehre an bayerischen Universitäten und in der Anwendung bei bayerischen Unternehmen angestrebt. Neben den erbrachten Eigenleistungen wird die Finanzierung der Forschungsverbünde zu etwa je einem Drittel von der bayerischen Wirtschaft, der Bayerischen Forschungsstiftung und der Bayerischen Staatsregierung getragen.

Bei der Realisierung des abayfor-online-Projektes bestand eine Zielsetzung darin, das Informationsangebot von abayfor einschließlich der angeschlossenen Verbünde in eine gemeinsame Informationsstruktur zu integrieren, sodaß dem Nutzer des Systems sämtliche Informationen in einheitlicher Weise und verbundübergreifend zur Verfügung stehen.

[1] URL: `http://www.abayfor.de`

Darüber hinaus sollten alle Forschungsinformationen in ein einheitliches Forschungsfeld strukturiert und mit Hilfe ausgewählter Stichwörter, Themenbereiche, Branchen und weiteren Deskriptoren klassifiziert werden. Als Basisarchitektur wurde deshalb ein relationales Datenbanksystem gewählt, das an das World Wide Web über ein CGI-Gateway gekoppelt wurde. Die Anforderungen an das zu entwickelnde Forschungsinformationssystem waren insbesondere:

- **Erstellen eines Prototyps:** Zur Entwicklung des Informationssystems sollten die Vertreter der einzelnen Forschungsverbünde mit einbezogen werden. Den Verbünden sollte frühzeitig ein paßwortgeschützter Prototyp zur Verfügung stehen, der die Grundfunktionalität des Gesamtsystems veranschaulicht.

- **Deklarative und navigierende Suche:** Der Anwender sollte sowohl über HTML-Anfrageformulare nach einzelnen Verbünden suchen können, als auch über einen HTML-Index auf die Homepage der einzelnen Verbünde navigieren können.

- **Ad-hoc-Anfragebearbeitung:** Die abayfor-Geschäftsstelle sollte über ein HTML-Formular die Möglichkeit haben, beliebige Ad-hoc-Anfragen in Form von SQL-Queries an die Datenbank abzusetzen. Dies ist insbesondere deshalb wichtig, da die abayfor-Geschäftsstelle als zentraler Kontakt- und Informationsvermittler agiert und das gespeicherte Wissenspotential für die Bearbeitung verschiedenster Anfragen nutzt.

- **Ansprechende Oberflächengestaltung:** Bei der Generierung der HTML-Seiten aus der Datenbank sollen Guidelines zur Präsentation von großen dynamischen Datenmengen in Hypermedia berücksichtigt werden. Beispielsweise sollten Listen von Referenzen über mehrere Seiten aufgeteilt und geeignete Zugriffsstrukturen gewählt werden. Die Darstellung der Verbünde sollte einheitlich erfolgen.

- **Einfache Wartbarkeit und Skalierbarkeit:** Die Hypertext-Oberfläche sollte einfach und ohne Programmieraufwand von den abayfor-Mitarbeitern gewartet und verändert werden können.

- **Online-Pflege der Daten:** Sämtliche Daten sollten Online über eine Pflegekomponente in die Datenbank eingegeben und gewartet werden können. Die Datenpflege sollte von den Forschungsverbünden mit Hilfe eines Web-Interfaces durchgeführt werden.

- **Abgestufte Benutzerautorisierung auf HTML-Seiten:** 21 Forschungsverbünde sollen selbst eigene Datenbestände in das Informationssystem eingeben, ändern und löschen können. Dabei darf kein Forschungsverbund schreibend auf die Daten eines anderen Forschungsverbundes zugreifen. Weiterhin sollten alle Daten, die durch die Mitarbeiter eines Forschungsverbundes in die Datenbank eingegeben werden, nochmal durch den Geschäftsführer verifiziert werden. Es existieren demnach vier verschiedene Benutzergruppen mit unterschiedlichen Rechten: Administratoren, Geschäftsführer, Mitarbeiter und Endanwender.

- **Überprüfung der Datenqualität:** Die abayfor-Geschäftsstelle sollte durch ein entsprechendes Werkzeug die Möglichkeit haben, die Datenqualität der abayfor-Datenbank zu überprüfen und fehlerhafte Einträge gegebenenfalls zu korrigieren. Hierzu gehört die Selektion der Verbünde, die länger als sechs Monate keine Daten mehr in das System eingegeben oder geändert haben und das Auffinden und Löschen von Doppeleinträgen.
- **Statistik:** Jeder Verbund sollte die Möglichkeit haben, die Zugriffe für die HTML-Seiten des eigenen Verbundes im Web einzusehen. Die abayfor-Geschäftsstelle erhält eine Gesamtstatistik aller Zugriffe auf das Informationssystem.

Auf der Basis dieser Anforderungen wurde das Gesamtsystem entworfen. In den folgenden Kapiteln werden Systemarchitektur, Modellierung der Datenbank, Implementierung der Hypertext-Anwendung und die sich daraus ergebenden Limitierungen vorgestellt.

4.2 Systemarchitektur

Der dezentrale Zugriff auf das System erfolgt über das Internet. Die Gesamtarchitektur beruht auf der Kopplung des relationalen Datenbanksystems TransBase [Tra96] an das World Wide Web. Das hierfür eingesetzte CGI-Programm (Web-Gateway) wurde am Datenbanklehrstuhl von Professor R. Bayer entwickelt [CVW95] und von FORWISS im Rahmen verschiedener Projekte erweitert [ZS98]. In HTML-Templates werden die Anfragen an die Datenbank und das Layout der dynamisch generierten HTML-Seiten fest definiert. Hierfür wurde eine eigene Syntax entwickelt, die es ermöglicht mit neu definierten Tags (`<!-- OMNIS ....>`) innerhalb von HTML-Kommentaren, Datenbankanfragen zu formulieren. Die Ergebnisse einer Datenbankanfrage werden in einer Schleife zurück gegeben und in der array-Variablen `table` zwischengespeichert. Daneben werden verschiedene Konstrukte der strukturierten Programmierung unterstützt, um zur Laufzeit auf Bedingungen reagieren zu können (if, then, else, foreach...). Weiterhin werden spezielle Tags für Beginn und Ende einer DB-Transaktion und den connect zu einer Datenbank zur Verfügung gestellt. Jede Transaktion, die in einem HTML-Template geöffnet wird muß auch in diesem wieder geschlossen werden. Transaktionen über mehrere HTML-Seiten hinweg sind demnach nicht möglich. Bei einer Datenbankanfrage analysiert das Web-Gateway ein HTML-Template, reicht die Anfragen an die Datenbank weiter und fügt die Ergebnisse zu einer HTML-Seite zusammen. Folgendes Beispiel soll die Syntax eines HTML-Templates verdeutlichen:

```
<TABLE CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
<TR>
<TD><IMG SRC="images/spacer.gif" WIDTH=10></TD>
<TD VALIGN=TOP>
<UL>
```

```
<!-- OMNIS SQL
     select DISTINCT count(DISTINCT a.num), a.Titel, a.Untertitel, a.Ende,
     a.Kurzbeschreibung, a.num
     from Projekte a, Projekt_Forschbereich b, Forschungsbereiche c
     where a.num = b.Projekt and b.Forschbereich = c.num and
     c.Verbund = &(num) order by 2 -->
   <!-- OMNIS SET titel = &(table[1]), utitel = &(table[2]),
                    ende = &(table[3]) -->
   <!-- OMNIS SET beschreibung = &(table[4]), schluessel = &(table[5]) -->

   <LI><A HREF="&(webcon_cgipath)/webcon/&(webcon_dbalias)/details/&(num)/
Projekte/num/&(schluessel)/1">&(titel)</A>
   <!-- OMNIS IFDEF utitel -->
      <FONT SIZE=-1>(&(utitel))</FONT>
   <!-- /OMNIS -->
   <P>
   <!-- OMNIS IF &(table[0]) < 4 -->
      <UL><FONT SIZE=-1><!-- OMNIS PRINTRAW beschreibung -->
          </FONT></UL><P>
   <!-- /OMNIS -->
<!-- /OMNIS -->
```

Der Name des HTML-Templates wird dem Web-Gateway als Parameter in der URL übergeben. Sämtliche Informationen, wie beispielsweise der genaue Pfad der HTML-Templates, der definierte Template- und Datenbank-Alias, der Datenbankname, die Kennung und das Paßwort zum Anmelden des Web-Gateways in die Datenbank werden in einer Konfigurationsdatei abgelegt. Da ein CGI-Prozeß nach jeder Client-Anfrage wieder beendet wird, muß für die Erstellung jeder HTML-Seite ein Datenbank-Connect durchgeführt werden. Die gesamte Systemarchitektur von abayfor-online wird durch Abbildung 4.1 veranschaulicht.

Mit Hilfe eines WWW-Browsers können Informationen in die abayfor-Datenbank eingegeben und anschließend recherchiert werden. Hierfür wurde eine umfangreiche Pflege- und Recherchekomponente entwickelt. Bei der Dateneingabe werden drei verschiedene Anwendergruppen mit unterschiedlichen Rechten unterschieden: Mitarbeiter von Verbünden, Geschäftsführer von Verbünden und Administratoren von abayfor. Mitarbeiter können ausschließlich Daten zu Projekten und Mitarbeitern in eine Pufferdatenbank eintragen oder ändern. Bei jedem Eintrag erhält die Geschäftsführung des jeweiligen Verbundes eine Email mit der Aufforderung die eingetragenen Daten zu verifizieren. Erst nach der Freischaltung durch die Geschäftsführung werden die Daten von der Pufferdatenbank in die abayfor-Datenbank übernommen und stehen für die allgemeine Recherche zur Verfügung. Geschäftsführer dürfen beliebige Daten zu ihrem eigenen Verbund eingeben, löschen und ändern. Die abayfor-Administratoren haben lesenden und schreibenden Zugriff auf sämtliche Daten aller Verbünde der abayfor-Datenbank.

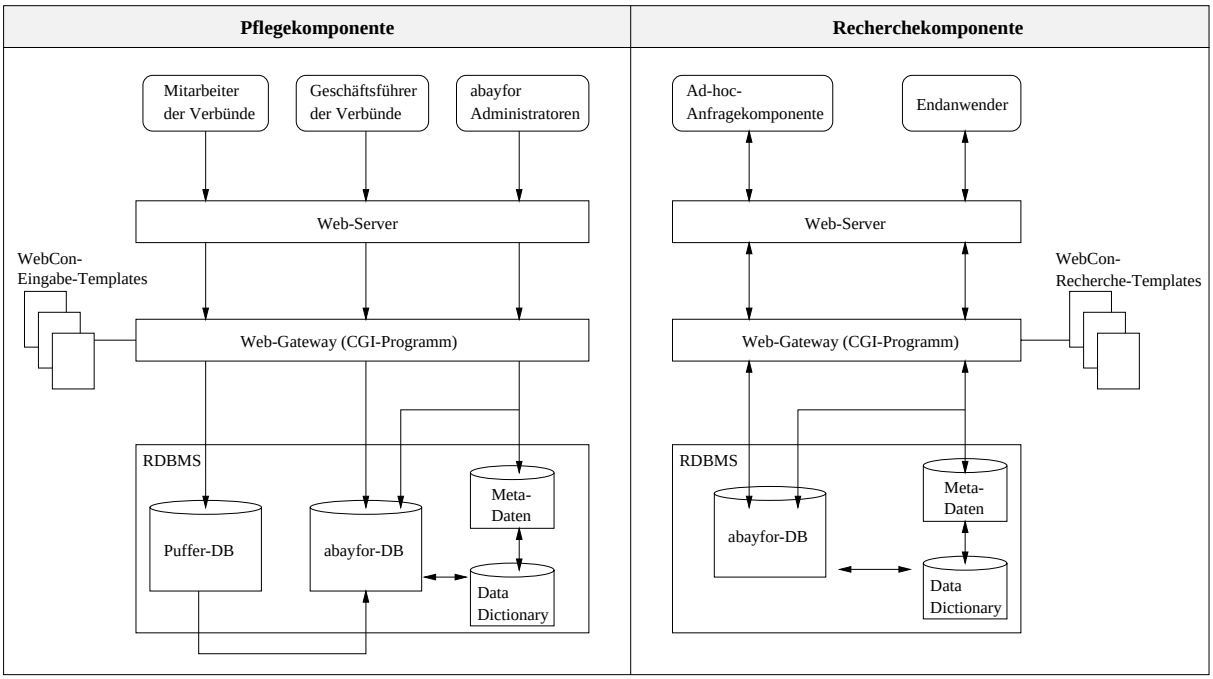


Abbildung 4.1: Schematische Darstellung der abayfor-online-Systemarchitektur

Um ein schnelles Prototyping und die einfache Skalierbarkeit der Hypertext-Anwendung zu gewährleisten, wurden innerhalb der abayfor-Datenbank Meta-Daten zu Hypertext-Struktur, Hypertext-Layout und Hypertext-Navigation in Meta-Relationen abgelegt [SZ99], [Zol96]. Die Meta-Daten werden durch hierfür entwickelte HTML-Formulare in die Datenbank eingegeben. Dies erlaubt die einfache Entwicklung und Wartung der Hypertext-Anwendung ohne Programmierkenntnisse. Die Befüllung der Meta-Relationen erfolgt ausschließlich durch abayfor-Administratoren. Bei der Generierung der dynamischen HTML-Seiten werden die Meta-Daten zur Bestimmung des Seitenlayouts durch HTML-Templates ausgewertet. Die HTML-Templates beinhalten demnach neben den SQL-Anfragen an den Datenbestand der abayfor-Datenbank auch SQL-Anfragen an die Meta-Relationen zur Bestimmung der Seitenstruktur und des Seitenlayouts. Folgendes Beispiel soll dies verdeutlichen:

```
<!-- OMNIS SQL select header, footer, hintergrund from layoutInfo
              where relname = '&(relname)' and seite = 'detail'
              and version = &(version) -->
   <!-- OMNIS SET header = &(table[0]) -->
   <!-- OMNIS SET footer = &(table[1]), hintergrund = &(table[2]) -->
<!-- /OMNIS -->
```

Durch eine SQL-Anfrage an die Meta-Relation `layoutInfo` wird Header, Footer und Hintergrund der HTML-Seite bestimmt, in die das Ergebnis der SQL-Anfragen an die abayfor-Datenbank eingebettet werden soll.

Für die Recherchekomponente werden im wesentlichen vier verschiedene Meta-Relationen unterschieden: `queryInfo`, `ergebnisInfo`, `detailInfo` und `layoutInfo`. Daraus ergibt sich eine dreistufige Navigationsstruktur durch die Datenbank: In einem HTML-Formular schränkt der Anwender die Datenmenge durch Suchkriterien ein (Query-Seite). Das Ergebnis der Anfrage wird anschließend in Form einer Treffer-Liste dargestellt (Ergebnis-Seite). Nach Auswahl eines Treffers erhält der Anwender die Detailansicht des aus der Ergebnis-Liste selektierten Treffers (Detail-Seite). Die Meta-Relationen beinhalten Referenzen auf das DataDictionary der Datenbank. Weiterhin werden Fremdschlüsselbeziehungen zwischen Relationen genutzt. Die einzelnen Meta-Relationen der Recherche-Komponente sollen im folgenden genauer erläutert werden.

Die Relation `queryInfo` beinhaltet die Informationen, die zur Generierung der HTML-Suchformulare (Query-Template) nötig sind. Jedes HTML-Formular ist eindeutig einer Relation zugeordnet und besteht aus verschiedenen Attributen dieser Relation. Die Attribute `relname` und `attr` der Meta-Relation modellieren den Namen einer DB-Relation und eines DB-Attributs. Die Bezeichnung eines DB-Attributs auf einer HTML-Seite wird durch das Attribut `bezeichnung` modelliert. Da es zu jeder Relation verschiedene HTML-Suchformulare geben kann, werden verschiedene Versionen unterschieden.

Relation queryInfo

Attribute	*Beschreibung*
relname	Name einer Relation
attr	Name eines DB-Attributs
pos	Position des Attributs im HTML-Formular
bez	Bezeichnung des Attributs im HTML-Formular
typ	Typ des Attributs: Popup, Check-Buttons oder Bild
version	Jeder Relation können durch die Versionsnummer verschiedene HTML-Suchformulare zugeordnet werden.

Die Relation `ergebnisInfo` beinhaltet die Informationen zur Generierung der Trefferlisten und wird durch das Ergebnis-Template ausgewertet.

Relation ergebnisInfo

Attribute	*Beschreibung*
relname	Name einer Relation
attr	Name eines DB-Attributs
pos	Position des Attributs auf einer HTML-Seite
bez	Bezeichnung des Attributs auf der Treffer-Liste
version	Versionsnummer

Die Relation `detailInfo` beinhaltet Informationen zur Generierung der Detailseiten und wird durch das Detail-Template ausgewertet.

Relation detailInfo

Attribute	*Beschreibung*
relname	Name einer Relation
attr	Name eines DB-Attributs
pos	Position des Attributs auf einer HTML-Seite
bez	Bezeichnung des Attributs
typ	Typ des Attributs: Email, Download, Text, URL oder Bild
version	Versionsnummer

Die Relation `layoutInfo` beinhaltet Layout-Informationen zu Anfrageformular, Trefferliste und Detailseite einer Relation und wird von allen drei Templates (Query-, Ergebnis-, Detail-Template) ausgewertet.

Relation layoutInfo

Attribute	*Beschreibung*
relname	Name einer Relation
seite	Seitentyp (Detail, Ergebnis, Anfrage)
header	Header der HTML-Seite
footer	Footer der HTML-Seite
hintergrund	Hintergrundfarbe der HTML-Seiten
version	Versionsnummer

Weiterhin werden in den drei separaten Relationen `queryInfoJoins`, `ergebnisInfoJoins` und `detailInfoJoins` Informationen zu Joins zwischen Relationen abgelegt. Dies ermöglicht die Anzeige von Attributen auf einer HTML-Seite, die zu verschiedenen miteinander in Beziehung stehenden Relationen gehören. Dabei werden vier verschiedene Join-Typen unterschieden: `links(inPlace)`, `links(extraPage)`, `details(inPlace)` und `details-(extraPage)`. Die ersten zwei Join-Typen entsprechen Hyperlinklisten, die entweder innerhalb der gleichen HTML-Seite (inPlace) oder auf einer separaten HTML-Seite (extraPage) angezeigt werden. Die zwei letzten Join-Typen ermöglichen die Anzeige von Detail-Daten einer anderen Relation ohne Hyperlinks auf der gleichen oder einer separaten HTML-Seite.

Die Meta-Relationen zusammen mit den darauf basierenden Werkzeugen zur Eingabe und Wartung der Meta-Daten wurden als *WebCon-Paket* bezeichnet.

4.3 Modellierung der Anwendung

Die Modellierung der abayfor-online-Anwendung ist unterteilt in die Modellierung des Anwendungsbereichs und die Modellierung der Hypertext-Oberfläche. Während die Modellierung des Anwendungsbereichs durch das OMT-Modell realisiert wurde, erfolgte die Modellierung der Hypertext-Applikation mit Hilfe des leicht modifizierten Application-Diagramms der RMM. Beide Modellierungen sollen im folgenden kurz vorgestellt werden. Ein Ausschnitt aus dem hier vorgestellten abayfor-Modell dient im Verlauf der weiteren

Arbeit als Beispiel für die Veranschaulichung der neu entwickelten HDBM (Hypermedia Database Design Methodology).

4.3.1 Datenbank-Design

Das für abayfor-online erarbeitete Datenmodell umfaßt neben allgemeinen Informationen zu den Bayerischen Forschungsverbünden auch Informationen zu Mitarbeitern, Forschungsbereichen, Fördermittelgebern und Projekten (siehe Abbildung 4.2). Im Mittelpunkt des Datenbankschemas steht der einzelne Forschungsverbund mit Informationen über Name, Acronym, Logo, Kurzbeschreibung und Verweisen zu diversen anderen Bereichen. In einem Forschungsverbund arbeiten Mitarbeiter in verschiedenen Positionen (wiss. Mitarbeiter, Techniker...). Jeder Mitarbeiter arbeitet an einem Standort, der eine bestimmte Ausstattung hat (z.B. Rasterelektronenmikroskop). Die Geschäftsstelle eines Verbundes entspricht dem Standort des Geschäftsführers.

Ein Forschungsverbund ist in mindestens einen thematischen oder organisatorischen Forschungsbereich unterteilt, in dem Projekte ablaufen können. An jedem Projekt arbeiten Mitarbeiter von mindestens einem Forschungsverbund. Jeder Mitarbeiter hat innerhalb der Projektarbeit eine Position (Wissenschaftler, Leiter). Ein Mitarbeiter kann dabei auch in verschiedenen Forschungsverbünden tätig sein und verschiedene Positionen ausüben (z.B. Sprecher, Vorstand). Ein Forschungsprojekt wird häufig mit Partnern aus der Wirtschaft durchgeführt. Einige Projekte werden von öffentlichen Fördermittelgebern finanziert, wie beispielsweise durch die Bayerische Forschungsstiftung. Insgesamt wird ein Forschungsverbund durch öffentliche Gelder oder/und Drittmittel aus der Industrie gefördert und finanziert.

Jeder Forschungsverbund kann Veranstaltungen abhalten oder sich an Veranstaltungen beteiligen (Messen). Weiterhin agieren Forschungsverbünde häufig als Informationsvermittler zu einem speziellen Themenkomplex und bieten deshalb Forschungsinformationssysteme zu einzelnen Gebieten an. Ebenfalls werden wissenschaftliche Veröffentlichungen durch die Mitarbeiter eines Forschungsverbundes publiziert, die häufig mit einem Projekt assoziiert sind.

4.3.2 Hypertext-Design

Die Oberfläche des abayfor-Informationssystems wurde mit Hilfe des Application-Diagramms der RMM (Relationship Management Methodology) modelliert (siehe Kapitel 5.3.4) und in HTML und JavaScript umgesetzt. Ziel hierbei war der Entwurf und die Implementierung einer benutzerfreundlichen Interaktionsschnittstelle im World Wide Web, die eine leichte Orientierung und intuitive Bedienung des Informationssystems ermöglicht.

In abayfor-online werden dem Anwender sowohl Informationen in Form von statischen als auch dynamisch aus der Datenbank generierten HTML-Seiten geboten. Zur sinnvollen Strukturierung der verschiedenen Bereiche wurden alle Informationen in die

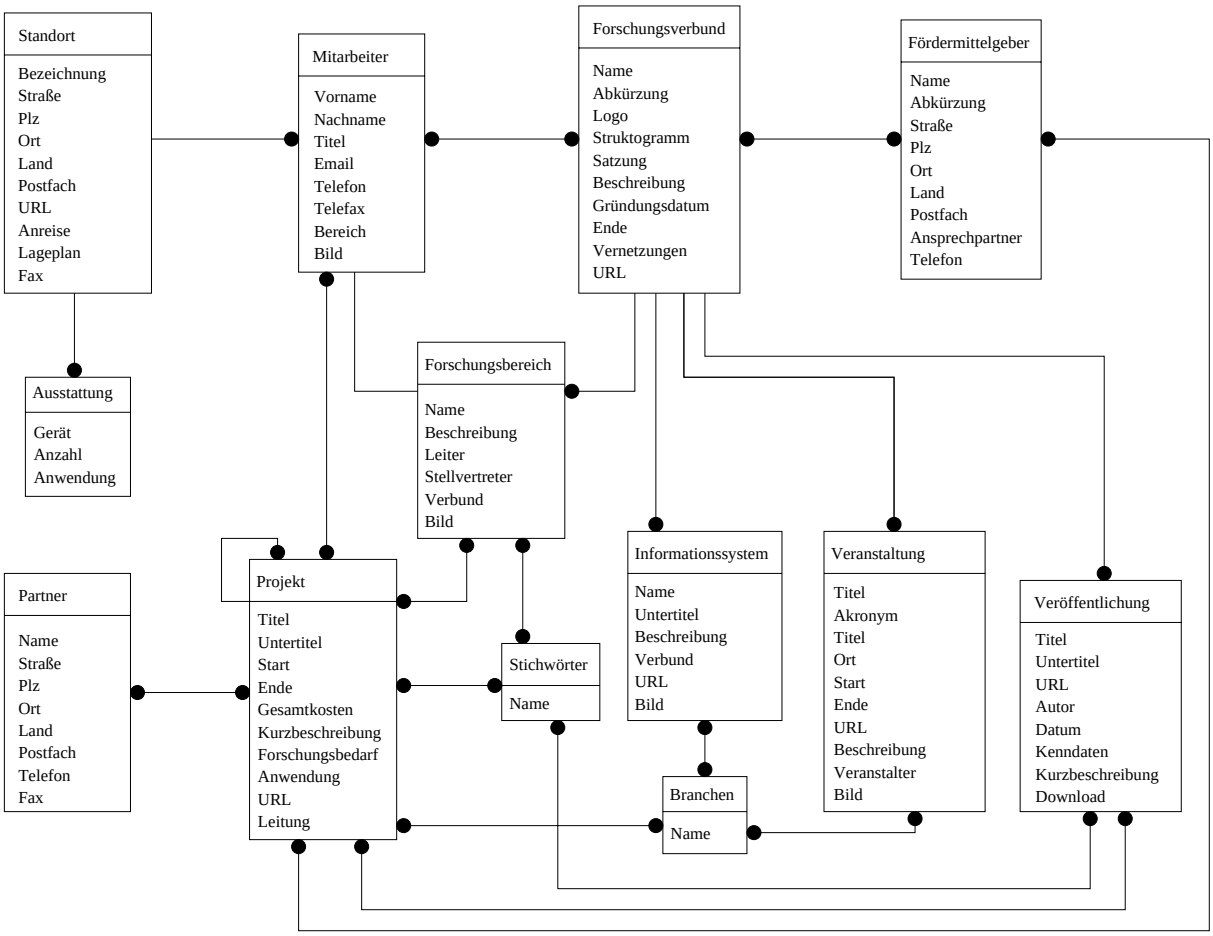


Abbildung 4.2: OMT-Modell der abayfor-Datenbank

folgenden fünf Hauptgebiete eingeteilt: 'Über abayfor', 'Forschungsverbünde', 'Datenbanksuche', 'Fördermittelgeber' und 'Forum'. Der Web-Site von abayfor liegt dabei eine baumartige inhaltliche Strukturierung zugrunde. Für die beiden Menüpunkte 'Forschungsverbünde' und 'Datenbanksuche' werden alle HTML-Seiten dynamisch aus der abayfor-Datenbank generiert. Der Menüpunkt 'Forschungsverbünde' erlaubt einen navigierenden Zugriff auf die Verbund-Homepage über einen Verbund-Index während der Menüpunkt 'Datenbanksuche' das Absetzen von vordefinierten Datenbank-Anfragen über ein Suchformular ermöglicht. Dabei gliedert sich die Verbund-Homepage in fünf Bereiche, die über eine Navigationsleiste von jeder darunterliegenden HTML-Seite erreichbar sind: 'Über uns', 'Kontakt', 'Mitarbeiter', 'Forschung' (Forschungsbereiche), 'Projekte'. Abbildung 4.3 zeigt das Application-Diagramm der abayfor-Web-Site. Jedes Rechteck markiert einen HTML-Seitentyp.

Aus den einzelnen HTML-Seitentypen des Application-Designs ist nicht immer ersichtlich mit Hilfe welcher Datenbanktabellen sie generiert werden. So wird beispielsweise die Kontakt-Seite aus dem Geschäftsführer der Mitarbeiter-Relation generiert. Die Abbildung zwischen Datenbankrelationen und Application-Diagramm wurde vorwiegend durch die Befüllung des Meta-Schemas definiert. Ein konzeptuelles Modell, das diese Abbildung

veranschaulicht und die automatische Generierung der zugehörigen Datenbankeinträge ermöglicht, wurde nicht verwendet.

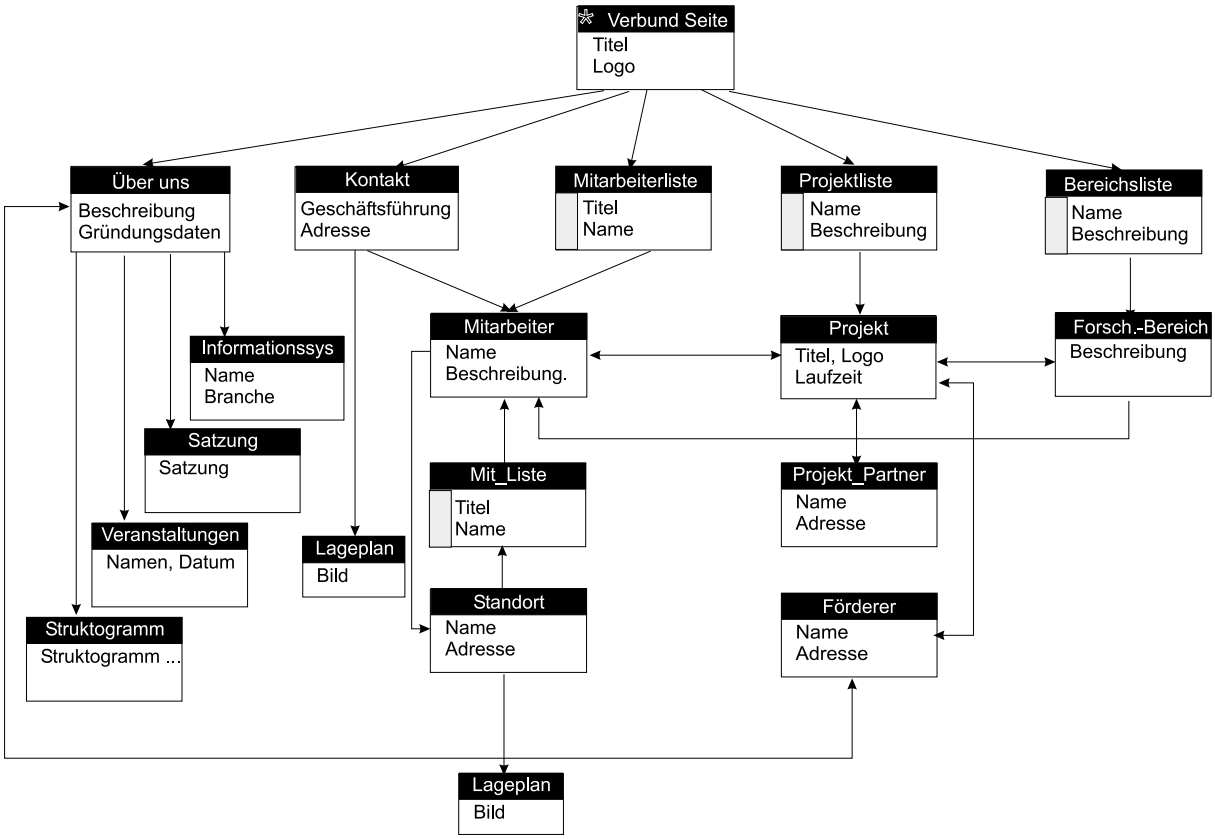


Abbildung 4.3: RMM-Application-Diagramm der abayfor Web-Site

4.4 Limitierungen des WebCon-Modells

Basierend auf den Erfahrungen aus dem abayfor-online-Projekt ergaben sich im WebCon-Meta-Modell insbesondere folgende Limitierungen:

- **Kein durchgängiger Modellierungsprozeß:** Datenbankschema und Hypertext-Schema werden getrennt voneinander entworfen und müssen anschließend durch das WebCon-Modell aufeinander abgebildet werden.
- **Keine Titel für HTML-Seitentypen:** Zu den einzelnen Detail-Seiten kann nur die Owner-Relation und eine Versionsnummer angegeben werden. Die HTML-Seitentypen haben demnach keine eindeutigen Namen.
- **Mangelnde Modellierung der Zugriffsstrukturen:** Das WebCon-Datenmodell unterstützt vier verschiedene Join-Typen, die verschiedene Zugriffsstrukturen ermöglichen: `links(inPlace)`, `links(extraPage)`, `details(inPlace)`, `details-(extraPage)`. Dabei gibt es verschiedene Restriktionen: bei Index-Listen, die

einen Hyperlink beinhalten (`links(inPlace)`, `links(extraPage)`) wird immer das erste Attribut als Hyperlink angezeigt. Weiterhin können diese Listen nur aus Attributen einer Relation aufgebaut sein. Ein Index ohne Hyperlinks wird als `details(inPlace)` oder `details(extraPage)` bezeichnet. Mit Hilfe von `details-(inPlace)` werden Attribute einer anderen Relation modelliert, die auf dieser Detail-Seite angezeigt werden sollen. Bei `details(extraPage)` wird nur ein Tupel auf einer separaten HTML-Seite angezeigt. Die Anzeige einer Liste von Tupeln (z.B. Veröffentlichungsliste) ist nicht möglich. Guided Tour wird als Zugriffsstruktur nicht unterstützt.

- **Mangelnde Modellierung der Hyperlinks:** Hyperlinks werden nicht separat modelliert. Dies hat zur Folge, daß der Inhalt eines Ankers nicht beliebig bestimmt werden kann. Es ist beispielsweise nicht möglich verschiedene, aufeinander folgende Attribute mit einem Hyperlink zu unterlegen. Bei einem Index wird immer das erste Attribut als ein Hyperlink angezeigt.

- **Keine Schachtelung von Seitentypen**: Das WebCon Meta-Modell wurde um die Fähigkeit erweitert, auf einer HTML-Seite auch Attribute von in Beziehung stehenden Relationen anzuzeigen. Es ist allerdings keine rekursive Schachtelung von Detail-Seiten möglich. Dies wäre sinnvoll, um beispielsweise Attribute von Relationen anzuzeigen, die über einen längeren Join-Pfad erreichbar sind.

- **Keine durchgängige Positionierung der Attribute:** Zuerst werden auf einer HTML-Seite nach der Position geordnet immer alle Attribute angezeigt, deren Namen in der Relation `detailInfo` abgelegt sind. Anschließend werden die in der Relation `detailInfoJoins` definierten Joins durchgeführt und deren Ergebnisse am Ende der HTML-Seite angezeigt. Eine durchlaufende Numerierung aller Elemente einer HTML-Seite ist so nicht möglich.

- **Feste Implementierung der Web-Typen:** WebCon bietet verschiedene Web-Typen zur Anzeige der Datenbank-Attribute auf Hypertext-Ebene. Diese Typen sind nicht erweiterbar, da sie fest im Web-Gateway implementiert sind.

4.5 Zusammenfassung

Im Rahmen des abayfor-online-Projektes wurde eine datenbankbasierte Web-Anwendung entwickelt, die über das Web Informationen zu den bayerischen Forschungsverbünden bereitstellt. Anforderungen an das zu entwickelnde System waren insbesondere: Schnelles Prototyping, Einfache Wartbarkeit und Skalierbarkeit, deklarative und navigierende Suchmöglichkeiten und konsistente Verwaltung und Darstellung des Informationsbestandes. Zur Gewährleistung dieser Anforderungen wurden neben der Anwendungsdomäne, Meta-Daten zur Hypertext-Struktur der Web-Site in einer relationalen Datenbank erfaßt.

Der verfolgte Meta-Daten-Ansatz bietet dem Anwender viele Vorteile. Durch die Erfassung von Meta-Daten innerhalb der Datenbank ist ein schnelles Prototyping möglich. Änderungen der Meta-Daten übertragen sich sofort auf die Hypertext-Oberfläche und werden für den Endanwender sichtbar. Weiterhin wird die einfache Wartung und Skalierbarkeit des Systems gewährleistet. Die Eingabe und Wartung der Meta-Daten erfolgt ohne Programmierkenntnisse über einen Web-Browser.

Neben diesen Vorteilen ergaben sich auch einige Probleme aus der Modellierung der Hypertext-Struktur innerhalb einer Datenbank. Hierzu gehört vor allem die eingeschränkte Mächtigkeit der generierten HTML-Seiten. Sowohl das Layout als auch die Strukturierung der angezeigten Informationen sind auf die im Meta-Modell vorgesehenen Attribute beschränkt. Die Modellierung der Hypertext-Anwendung ist demnach stark von dem zugrunde liegenden Meta-Modell abhängig. In abayfor-online mußten deshalb die HTML-Templates von ungefähr der Hälfte aller Detail-Seiten der Forschungsverbünde hardcodiert werden.

Aus den Erfahrungen, die aus dem abayfor-online-Projekt resultierten, wurde die HDBM-Design-Methode entwickelt, die einen durchgängigen Designprozeß unterstützt und die einfache und schnelle Generierung von dynamischen HTML-Seiten aus Datenbankinhalten auf der Basis eines umfassenden Meta-Modells ermöglicht.

Kapitel 5

Management großer Web-Sites mit HDBM

In diesem Kapitel wird eine neue Methodik für das Design und die integrierte Verwaltung großer Web-Sites vorgestellt. Die HDBM (Hypermedia Database Methodology) ist speziell auf den Entwurf von Web-Anwendungen abgestimmt, in denen ein Datenbanksystem als Speicherkomponente dient. Der neu entwickelte Ansatz basiert auf der logischen Strukturierung von Hypertext in einem relationalen Datenmodell mit Hilfe von Meta-Daten. Dabei ist der Design-Prozeß der HDBM unterteilt in Datenbank-Design und Hypermedia-Design, wobei beide Phasen wiederum in konzeptuelles und logisches Design untergliedert sind. Als semantische Datenmodelle kommen im Verlauf des konzeptuellen Designs das ER-Modell zur Modellierung der Anwendungsdomäne und das erweiterte RMDM zur Modellierung von Hypertext-Struktur, -Navigation und -Layout zum Einsatz. Die semantischen Datenmodelle werden im Verlauf des logischen Designs auf ein relationales Web-Datenbank-Schema abgebildet. Dabei erfolgt die Speicherung der Hypertext-Struktur in einem Meta-Schema, das als Erweiterung des Systemkatalogs der Datenbank dient.

Im folgenden Kapitel werden zunächst die Anforderungen an ein Application Framework für datenbankbasierte Informationssysteme im Web gesammelt. Anschließend wird ein integrierter Entwurfsprozeß zur Entwicklung datenbankbasierter Web-Anwendungen vorgestellt. In diesem Zusammenhang werden die einzelnen Entwicklungsphasen mit den zugehörigen semantischen und logischen Datenmodellen diskutiert. Auf der Basis der erweiterten RMM erfolgt die Entwicklung des HDBM-Meta-Modells, das in Kapitel 5.5 vorgestellt wird.

5.1 Anforderungen

Auf der Basis der in den vergangenen Kapiteln erörterten Problemstellung wird ein Pflichtenheft für Entwurf und Realisierung von datenbankbasierten Informationssystemen im Web erarbeitet. Dabei sollen folgende Grobziele verfolgt werden:

- Einfache Wartung und verbesserte Skalierbarkeit durch Anwendung einer strukturierten Entwurfsmethode und Aufteilung der Applikation in eine Drei-Ebenen-Architektur
- Konsistente Generierung von Hypertext auf der Basis eines vorgegebenen Schemas und Überwachung von Integritätsbedingungen mit Hilfe von Constraints
- Verbesserte Suchmöglichkeiten durch strukturelle Anfragetechniken über Hypertext
- Hohe Konfigurierbarkeit auf spezifische Applikationen durch ein generisches Modell

Um eine Methodik zu konzipieren, die diesen Zielen gerecht wird, wurde der nachfolgende Anforderungskatalog aufgestellt.

Anforderung 1: Durchgängiger Hypermedia-Entwurfsprozeß

Die unterschiedlichen Entwurfsmethoden im Hypertext- und Datenbankbereich verwenden verschiedene semantische und logische Datenmodelle, die häufig nicht aufeinander abgestimmt sind [AMM97b]. Ziel ist deshalb die Entwicklung eines durchgängigen Entwurfsprozesses zur Beschreibung der gesamten datenbankgestützten Web-Anwendung auf der Basis semantischer und logischer Entwurfs- und Datenmodelle aus dem Datenbank- und Hypermedia-Bereich (siehe auch Kapitel 5.2).

Anforderung 2: Suchmöglichkeiten über Hypertext

Bei der herkömmlichen Anbindung von Datenbanken an das Web werden die Informationen zur Strukturierung, Interaktion und Präsentation von HTML-Seiten außerhalb der Datenbank abgelegt und können deshalb bei der Auswertung von Ad-hoc-Anfragen an die Datenbank nicht ausgewertet werden. Dies hat zur Folge, daß Anfragen an die Datenbank vordefiniert und HTML-Layout und Struktur der Anfrageergebnisse fest in HTML-Vorlagedateien oder Programmen codiert werden. Bei Ad-hoc-Anfragen an die Datenbank ist die Struktur des Anfrageergebnisses vorher nicht bekannt. Die Ergebnisse werden deshalb in einfache HTML-Listen oder -Tabellen eingebettet. Das der Web-Site zugrunde liegende Schema sollte im Gegensatz hierzu bei der Ergebnispräsentation von Ad-hoc-Anfragen berücksichtigt werden [HB98b] (siehe auch Kapitel 7.1).

Anforderung 3: Integritätsbedingungen mit Hilfe von Constraints

Wichtig für die langjährige Nutzung einer Datenbank sind Methoden zur Schemaerweiterung. Dieses auch als *Schemaevolution* bezeichnete Vorgehen wird von vielen Datenbanksystemen in einem gewissen Rahmen unterstützt (siehe z.B. Oracle 8 [CHRS98]). Hierunter fallen beispielsweise das Anlegen neuer Tabellen, die Veränderung von Datentypen, das Hinzufügen von Attributen und das Löschen von Tabellen. Derartige nachträglich

durchgeführte Änderungen am DB-Schema führen zu erheblichem Implementierungsaufwand bei der Überarbeitung der Hypertext-Anwendung und sind leicht mit Konsistenzverletzungen der Oberfläche verbunden. Es ist deshalb wichtig, daß Änderungen des Datenbankschemas einfach auf die Hypertext-Struktur übertragen werden können. Wird beispielsweise eine Tabelle in der Datenbank gelöscht, so sollen automatisch alle hierzu gehörenden HTML-Templates gelöscht werden. Die Abbildung zwischen dem Datenbank- und dem Hypertext-Schema sollte durch Integritätsbedingungen überprüft und gegebenenfalls dynamisch angepaßt werden (siehe auch Kapitel 5.6.4).

Anforderung 4: Einfache Wartbarkeit

Die mangelnde Trennung von Daten und Präsentation im Web führt zur schweren Wartbarkeit von Web-Sites. Änderungen der Hypertext-Anwendung bezüglich der Navigation, des Layouts, der Strukturierung und der Dateninhalte sollten deshalb einfach und unabhängig voneinander vorgenommen werden können. Das Informationssystem sollte demnach durch eine klare Trennung der verschiedenen Ebenen logische und physische Datenunabhängigkeit gewährleisten.

Anforderung 5: Schnelles Prototyping

Der verstärkte Einsatz von multimedialen Elementen, die unterschiedlichen Fachkenntnisse der Mitarbeiter innerhalb des Projektteams und die starke Gewichtung des Oberflächendesigns machen ein frühes Prototyping bei Anwendungen im Web sehr wichtig. Die Erstellung eines Prototypen sollte deshalb einfach und schnell möglich sein. Weiterhin sollte die Hypertext-Anwendung ohne Programmierkenntnisse auf der Basis des DB-Schemas vom Anwender erstellt werden können.

Anforderung 6: Verschiedene Benutzersichten

Im Gegensatz zu *data-driven*-Modellierungsansätzen, kann man das Hypermedia-Design auch als *user-centered* bezeichnen [Tro98]. Die verfügbaren Daten werden aus der Sicht verschiedener Anwenderklassen im Web modelliert und implementiert. Das Ergebnis sind Web-Sites, die besser an unterschiedliche Benutzerklassen und deren Anforderungen angepaßt sind. Die Modellierung verschiedener Sichten auf das Informationssystem sollte deshalb möglichst einfach und ohne zusätzlichen Implementierungsaufwand umgesetzt werden können. Das Informationssystem sollte sich adaptiv an verschiedene Benutzergruppen anpassen lassen.

Anforderung 7: offline- und online-Generierung von HTML-Seiten

Bei der Entwicklung von Web-Applikationen, die auf DB-Systeme zugreifen, werden die Inhalte der Datenbank ausgespoolt, in HTML-Seiten eingebettet und in einem Web-Browser angezeigt. Die Kopplung der Datenbank kann dabei fest oder lose sein. Man

spricht in diesem Zusammenhang auch von einer online- oder offline-Seitengenerierung abhängig davon, ob die HTML-Seiten dynamisch zur Laufzeit oder bereits im Vorfeld generiert und im Dateisystem abgelegt werden. Beide Varianten haben verschiedene Vor- und Nachteile und sollten unterstützt werden. Hierzu muß ein geeignetes Tool zur Generierung der HTML-Seiten einschließlich der URLs und zur inkrementellen Wartung der im Dateisystem materialisierten HTML-Seiten angeboten werden (siehe auch Kapitel 7.2).

Anforderung 8: Hohe Performance, einfache Skalierbarkeit

Für die Benutzerakzeptanz bei Anwendungen im World Wide Web spielt die Performance des Gesamtsystems eine wichtige Rolle. In diesem Zusammenhang sind unterschiedliche Aspekte der Systemarchitektur (wie z.B. Verteilung der Komponenten auf Server/Client, Two-Tier vs. Three-Tier-Architektur) und Ladezeiten von Informationen (z.B. Oberflächengestaltung) zu berücksichtigen. Wichtig ist auch die einfache Skalierbarkeit der Systemarchitektur. Web-Server, Application-Server und Datenbanksystem sollten beispielsweise auf verschiedenen Rechnern installiert werden können, um eine optimale Lastverteilung zu erzielen (siehe auch Kapitel 6).

5.2 Entwurfsprozeß

Betrachtet man die Phasen des Datenbankentwurfs und des strukturierten Hypermedia-Designs, sind bei der Entwicklung einer datenbankbasierten Hypertext-Anwendung folgende zwei Entwurfsszenarien denkbar:

- **Top-Down-Strategie:** Das Datenbank-Design erfolgt auf der Basis des Hypertext-Designs der Web-Site.
- **Bottom-Up-Strategie:** Das Hypertext-Design wird auf der Basis des entworfenen Datenbank-Schemas durchgeführt.

Bei der *Top-Down-Strategie* erfolgt zunächst der Entwurf der Web-Anwendung mit Hilfe einer Hypermedia-Entwurfsmethode. Erst darauf abgestimmt wird das Datenbank-Schema entworfen und die Implementierung der Hypermedia-Anwendung durchgeführt. Diese Vorgehensweise wird beispielsweise im Autoweb-Ansatz [FP98] verfolgt.

Die *Bottom-Up-Strategie* entspricht in ihrem Ansatz dem klassischen Datenbankdesign. Auf der Basis des entwickelten Datenbankschemas erfolgen der Entwurf und die Implementierung der Hypertext-Anwendung. Dieser Ansatz wird insbesondere bei der Anbindung von legacy-Datenbeständen an das Web verwendet.

Der Entwurf verschiedener Online-Datenbanken am FORWISS hat gezeigt, daß ein optimales Design-Ergebnis durch die parallele, inkrementelle und iterative Abarbeitung der einzelnen Phasen des Datenbank- und Hypertext-Entwurfs erzielt wird. Dabei beeinflussen sich Hypertext-Design und Datenbank-Design gegenseitig. Aus einer Datenbank-Tabelle werden verschiedene HTML-Seitentypen generiert. Jedes Tupel einer Tabelle

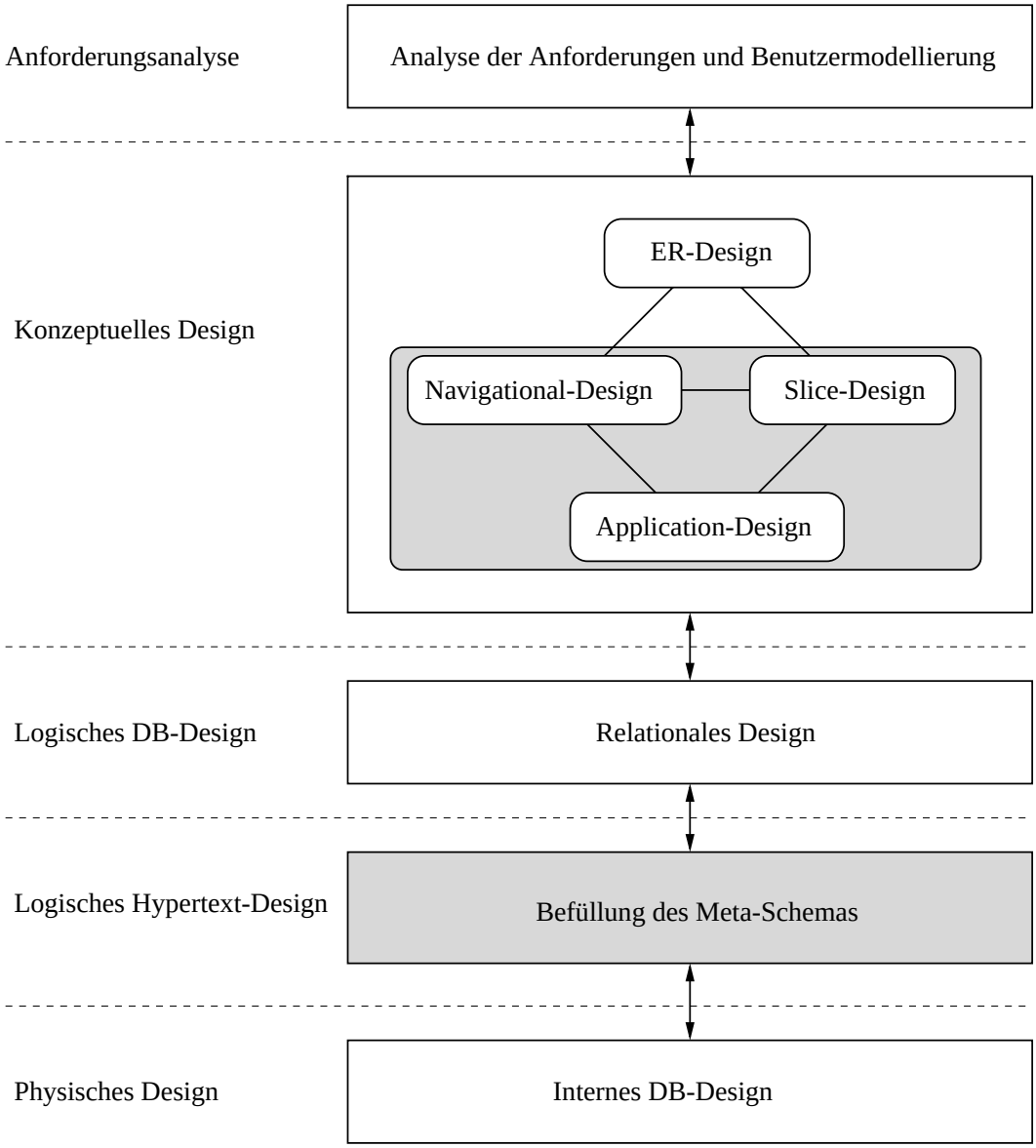


Abbildung 5.1: HDBM-Entwurfsprozeß

entspricht demnach einer Instanz des zugehörigen HTML-Seitentyps. Hypertext-Links werden auf Datenbankebene als ein Join zwischen zwei Relationen realisiert.

Bei dem Entwurf von Datenbanken können drei verschiedene Abstraktionsebenen unterschieden werden: konzeptuelle Ebene, logische Ebene und physische Ebene. Die konzeptuelle Ebene dient dazu, den Anwendungsbereich mit Hilfe semantischer Datenmodelle zu strukturieren. Diese Ebene ist unabhängig von dem zum Einsatz kommenden Datenbanksystem. Auf der logischen Ebene wird die Datenbankanwendung in den Konzepten des zum Einsatz kommenden Datenbanksystems modelliert. Im relationalen Fall wären dies beispielsweise Relationen, Tupel und Attribute. Die unterste Abstraktionsebene ist die physische Ebene. Ziel des physischen Entwurfs ist die Verbesserung der Leistungsfähigkeit der Datenbankanwendung durch die Betrachtung von Speicherstrukturen und Indexstrukturen.

In HDBM werden die aus der Datenbankmodellierung bekannten Entwurfsphasen: *Anforderungsanalyse*, *Konzeptuelles Design*, *Logisches Design* und *Physisches Design* beibehalten [EN94], die Inhalte der einzelnen Phasen aber um Hypertext-spezifische Modelle und Methoden ergänzt. Abbildung 5.1 zeigt den Entwurf einer datenbankbasierten Hypertextanwendung im Web mit Hilfe von HDBM.

Im Verlauf der Analysephase erfolgt neben der Charakterisierung des Problembereichs auch die Modellierung der Benutzerklassen [Tro98]. In diesem Rahmen müssen mögliche Benutzer des Internet-Informationssystems klassifiziert und deren Wissensstand, Erwartungen und Anforderungen an das System gesammelt werden. Die Analysephase wird detailliert in [GMP95], [NN95] besprochen und soll in dieser Arbeit nicht genauer diskutiert werden.

Während des anschließenden konzeptuellen Designs erfolgt die Modellierung des Anwendungsbereichs mit Hilfe semantischer Datenmodelle. Hier kommen neben dem ER-Modell die aus der RMM (Relationship Management Methodology) bekannten Modellierungskonzepte zum Einsatz.

Im Verlauf des logischen DB-Designs erfolgt die Umsetzung des ER-Modells in ein relationales Datenmodell auf der Basis bekannter Abbildungsregeln.

Das logische Hypertext-Design umfaßt das Design des Layouts und die Befüllung des Meta-Schemas. Für die Abbildung der Hypertext-Struktur auf eine Datenbank wurde ein neuartiges Meta-Modell entwickelt. Das Speichern der Hypertext-Struktur innerhalb einer Datenbank bringt viele Vorteile bei Personalisierung, Anfragebearbeitung, Generierung und Materialisierung von HTML-Seiten mit sich und unterscheidet HDBM von bestehenden Entwurfsmethoden auf diesem Gebiet.

Abschließend erfolgt das physische Datenbank-Design. Im Verlauf des physischen Designs werden das interne Schema und die damit zusammenhängenden Systemparameter festgelegt. In den nächsten Kapiteln werden die einzelnen Entwurfsphasen und das HDBM-Meta-Schema genauer beschrieben.

5.3 Erweiterte RMDM

Die Abbildung von Informationsstrukturen der realen Welt auf das logische Schema einer Datenbank wird meist nicht direkt vorgenommen, sondern vielmehr über den Zwischenschritt der Abbildung auf ein semantisches Datenmodell, das dann auf das logische Schema transformiert wird. Das am häufigsten zu diesem Zweck eingesetzte Datenmodell ist das Entity-Relationship-Modell [Che76], das in der heute gebräuchlichen Form 1976 von P. Chen spezifiziert wurde und in den folgenden Jahren zahlreiche Erweiterungen erfuhr. Dabei eignet sich das ER-Modell vor allem zur Abbildung auf relationale Datenbanken. Hierzu existieren klare Abbildungsregeln, die eine systematische, weitgehend verlustfreie Abbildung erlauben.

Die RMM gehört zu den bekanntesten Hypermedia-Entwurfsmethoden. Sie setzt auf

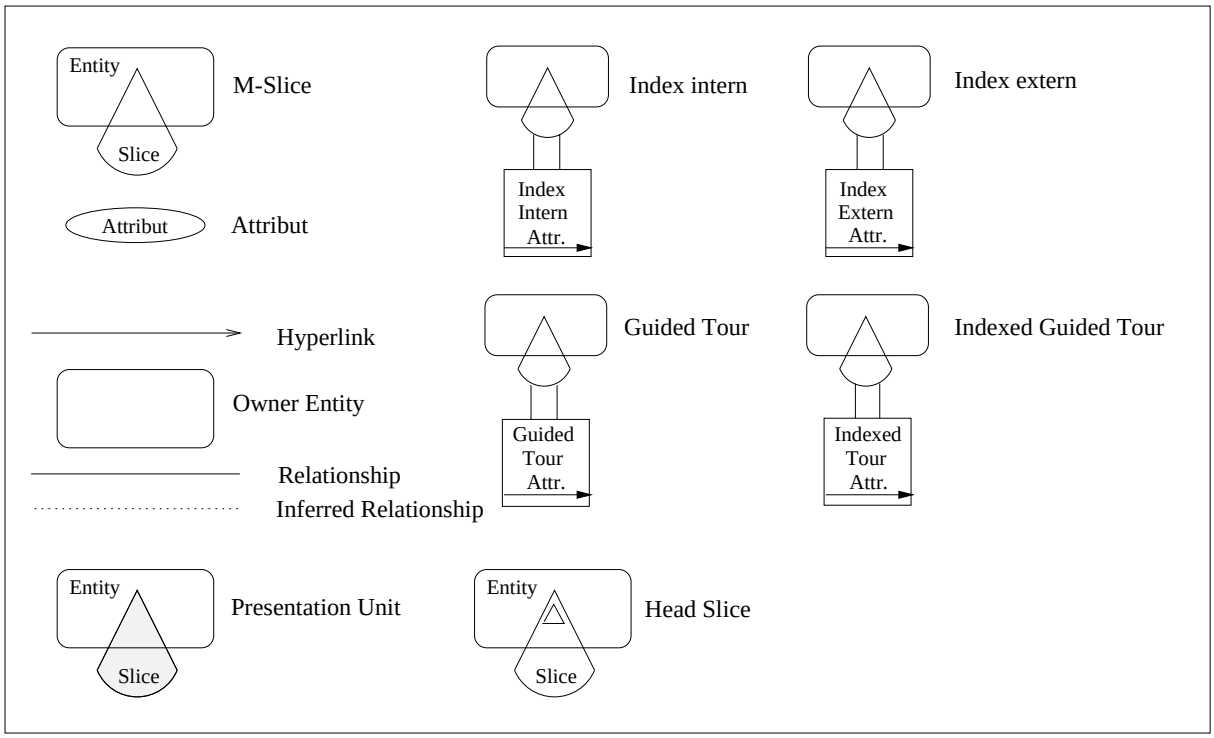


Abbildung 5.2: Graphische Elemente des erweiterten RMDM

bestehenden Hypermedia-Modellen auf, erweitert diese aber um verschiedene Konzepte. Gleichzeitig basiert die RMM auf einem vereinfachten ER-Modell und definiert damit die Abbildung zwischen den wichtigsten Modellierungskonzepten des ER-Modells (Entities und Relationships) und den Modellierungskonzepten des RMM-Datenmodells (Slices, Zugriffsstrukturen und Hyperlinks). Diese Abbildungsvorschriften können auf relationale Ebene übertragen und zur Modellierung eines Meta-Schemas genutzt werden. Allen Design-Schritten der RMM liegt das Relationship Management Data Model (RMDM) zugrunde. Für die HDBM-Entwurfsmethode wurden die Elemente des RMDM erweitert und im Hinblick auf die spätere Abbildung auf ein Meta-Modell angepaßt. Abbildung 5.2 zeigt die im Verlauf der HDBM eingesetzten grafischen RMDM-Modellierungskonzepte.

Im folgenden sollen das ER-Design, Slice-Design und Navigational-Design genauer erläutert werden. Als einheitliches Beispiel zur Erläuterung der einzelnen Designschritte dient der Entwurf einer einfachen Web-Site für einen Forschungsverbund.

5.3.1 ER-Design

Die grundlegenden Modellierungsstrukturen des ER-Modells sind *Entities* (Gegenstände), *Relationships* (Beziehungen) zwischen diesen Entities und *Attribute*.

Ein Entity ist als ein eigenständiges, abgrenzbares Objekt der zu modellierenden Welt definiert. Eine Menge von Objekten, die einander nach gewissen Kriterien ähnlich sind, werden zu einem *Entity-Typ* abstrahiert. Entity-Typen werden graphisch als Rechteck dargestellt, wobei der Name des Entity-Typs innerhalb des Rechtecks angegeben wird.

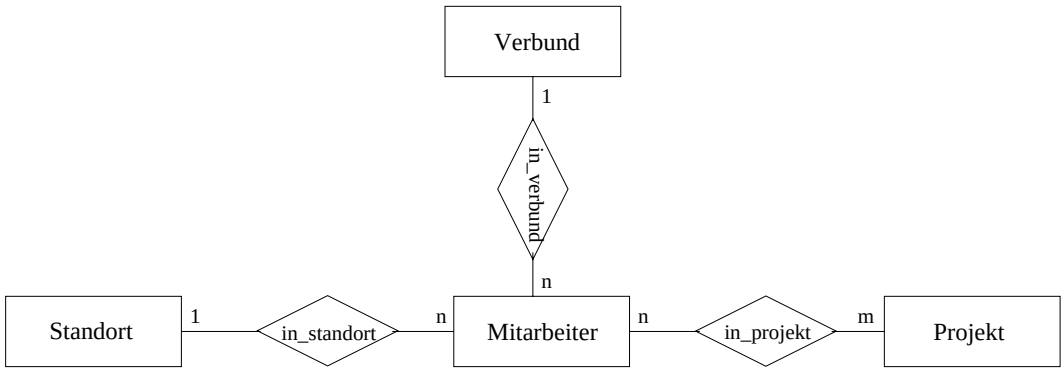


Abbildung 5.3: ER-Modell eines Forschungsverbundes

Ein Entity ist ein strukturiertes Datenobjekt, das durch ein oder mehrere *Attribute* in seinen Eigenschaften beschrieben wird. Einem Attribut ist ein *Wertebereich* zugeordnet, der die Menge der möglichen Attributwerte eines Entities definiert. Attribute werden durch Kreise oder Ovale graphisch beschrieben und den Rechtecken durch verbindende Kanten zugeordnet.

Eine Relationship wird als Beziehung zwischen Entities definiert. Eine Relationship kann ebenfalls durch Attribute beschrieben werden. Die Entities, die an einer Relationship beteiligt sind, werden die *Teilnehmer* (participants) der Relationship genannt. Die Anzahl der Teilnehmer ist der *Grad* (degree) der Relationship. Ferner werden Relationships in totale und partielle Relationships gegliedert. Falls jede Instanz des Entity-Typs A mit mindestens einer Instanz des Entity-Typs B eine Relationship bildet, handelt es sich um eine totale Teilnahme von A an der Relationship. Ansonsten ist die Teilnahme von A an der Relationship partiell. Zur näheren Spezifikation der Zahl von Entities, die durch Relationships zueinander in Beziehung gesetzt werden dürfen, können sogenannte *Kardinalitäten* für Relationships vergeben werden. Dabei unterscheidet man: 1:1-, 1:n- und n:m-Relationships. Eine Menge von Relationships, die einander ähnlich sind, werden unter einem *Relationship-Typ* zusammengefaßt und als Raute mit entsprechender Beschriftung dargestellt.

Ein gerade auch mit dem objektorientierten Paradigma in den Vordergrund tretendes Konzept ist die Bildung von Untertypen, deren Attributstruktur teilweise von anderen Typen ererbt wird und deren Elemente zugleich auch Elemente der vererbenden Typen sind. Im erweiterten Entity Relationship (EER) Modell sind solche Zusammenhänge durch einen speziellen Relationship-Typ ausdrückbar, der als *Generalisierung* bezeichnet wird. Durch das RMDM werden nur die Modellierungskonzepte eines vereinfachten ER-Modells unterstützt.

Die vorgestellten Konzepte des ER-Modells sollen anhand eines Beispiels erläutert werden. Abbildung 5.3 zeigt die vereinfachte Modellierung eines Forschungsverbundes. Jeder Forschungsverbund beschäftigt verschiedene Mitarbeiter, die eindeutig einem Verbund zugeordnet sind. Jeder Mitarbeiter arbeitet an genau einem Standort und kann an verschiedenen Projekten beteiligt sein.

5.3.2 Slice-Design

Im Verlauf des Slice-Designs werden die *M-Slices* entworfen. Ein M-Slice gruppiert zusammengehörige Attribute eines Entities, die später eine Einheit auf einer HTML-Seite bilden. Das *M* vor dem Slice steht für die russischen Matrjeska-Puppen und soll die mögliche Schachtelung von Slices verdeutlichen. Eine Menge gleich strukturierter Slices wird unter einem Slice-Typ zusammengefaßt. Im Rahmen dieser Arbeit werden zur Vereinfachung die Begriffe M-Slice für M-Slice-Typ bzw. Entity für Entity-Typ verwendet, soweit die Bedeutung der Begriffe klar aus dem Kontext hervor geht.

Jeder Slice ist eindeutig einem Entity zugeordnet und kann rekursiv aus anderen Slices aufgebaut sein. Die grafische Notation für M-Slices entspricht einem Kreisausschnitt. Entity und Slice überlappen sich teilweise, wobei das Entity das *Owner-Entity* des zugehörigen M-Slices bildet. Der Name des Entities wird in die obere linke Ecke geschrieben, der Name des Slices erscheint im unteren Teil des Slice-Ausschnitts (siehe Abbildung 5.4). Die eindeutige Bezeichnung eines M-Slices setzt sich aus dem Namen des Owner-Entities und dem Namen des Slice-Typs zusammen (z.B. `Mitarbeiter.Info`). Mehrere einem Entity zugeordneten Slices, die über Intra-Record-Navigation miteinander in Beziehung stehen, sind einem *Head Slice* untergeordnet, der den Default-Einstiegspunkt für dieses Entity bildet. Slices, die eigenständig als HTML-Seite angezeigt werden, bezeichnet man als *Presentation Units*; Slices, die Bestandteil eines anderen Slices sind, bezeichnet man als *Component Slice*. Sowohl für Head Slices als auch für Presentation Units existieren in der RMDM keine Modellierungskonzepte. Gerade für die Materialisierung der Hypertext-Views (Kapitel 7.2) und zur Anfragebearbeitung über Hypertext (Kapitel 7.1) ist diese Unterscheidung, wie später noch gezeigt wird, wichtig. Das RMDM wurde deshalb um zwei neue Modellierungskonzepte erweitert: Presentation Units und Head Slices. Presentation Units werden schraffiert dargestellt, Head Slices sind durch ein Dreieck gekennzeichnet.

Bei einem M-Slice werden physische von konzeptuellen Bindungen unterschieden. Der Slice-Ausschnitt zeigt die physische Bindung. Alle Elemente, die in dem jeweiligen M-Slice direkt angezeigt werden, erscheinen in diesem Bereich. Elemente, die Teil der direkten Umgebung des M-Slices sind, wie beispielsweise Ziel-Slices, die innerhalb des M-Slices über einen Hyperlink referenziert werden, erscheinen außerhalb des Slice-Ausschnitts. Alle Elemente eines M-Slices, die innerhalb des überlappenden Bereichs des Owner-Entities liegen gehören zu diesem Entity. Alle Elemente, die außerhalb des Owner-Entities aber innerhalb des M-Slices liegen, gehören zu anderen Owner-Entities. Dabei bleibt festzuhalten, daß ein M-Slice nur beschreibt, welche Informationen angezeigt werden sollen aber nicht, wie diese Informationen dargestellt werden. Dies ist Inhalt des Layout-Designs.

Das konzeptuelle Hypertext-Design soll anhand der Modellierung eines Mitarbeiters eines Forschungsverbundes verdeutlicht werden (siehe Abbildung 5.4). Zu jedem Mitarbeiter eines Forschungsverbundes werden verschiedene Attribute des Owner-Entities, wie beispielsweise der *Nachname*, *Vorname* und *Titel* angezeigt. Diese Attribute erscheinen deshalb in dem überlappenden Bereich zwischen Slice und Owner-Entity *Mitarbeiter*.

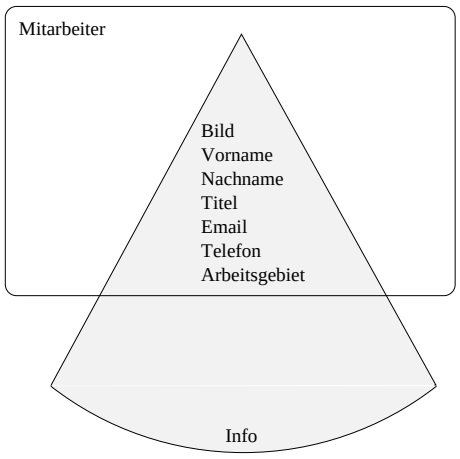


Abbildung 5.4: Slice-Design von Mitarbeitern

5.3.3 Navigational-Design

Im Verlauf des Navigational-Designs werden die Slices mit Hilfe von *Zugriffsstrukturen* und *Hyperlinks* zueinander in Beziehung gesetzt. Eine Zugriffsstruktur definiert dabei die Art und Weise, wie auf eine Menge von Instanzen eines Slice-Typs zugegriffen wird. Es werden vier verschiedene Zugriffsstrukturen, *Index Intern*, *Index Extern*, *Guided Tour* und *Indexed Guided Tour* unterschieden. Weiterhin wurde bei den Zugriffsstrukturen das Attribut des Owner-Entities, nach dem der geführte Rundgang bzw. der Index sortiert werden soll, modelliert. Die einzelnen Zugriffsstrukturen sollen im folgenden genauer erläutert werden:

- *Index Intern*: Die Instanzen eines M-Slices werden als Liste dargestellt. Häufig sind dies reine Hyperlinklisten, wie beispielsweise eine Liste von Mitarbeiternamen, die mit Hyperlinks unterlegt sind. Die Elemente eines Index können aber auch nur Attribute sein. Ein Beispiel hierfür wäre eine Veröffentlichungsliste. Bei einem internen Index wird die Tupelliste auf der gleichen HTML-Seite angezeigt, auf der diese definiert wurde.

- *Index Extern*: Im Unterschied zum internen Index wird die Tupelliste auf einer separaten HTML-Seite angezeigt. Dies ist beispielsweise sinnvoll, wenn man auf eine große Anzahl von Tupeln zugreifen möchte.

- *Guided Tour*: Der Anwender navigiert sich mit Hilfe eines geführten Rundgangs durch die Instanzen eines M-Slices. Die Instanzen des M-Slices müssen nach einem bestimmten Kriterium sortiert werden. Jede HTML-Seite beinhaltet einen Verweis zur vorherigen und zur nächsten HTML-Seite.

- *Indexed Guided Tour*: Ist eine Kombination eines externen Index mit einer Guided Tour. Der Anwender kann sowohl über eine Hyperlinkliste als auch über einen Rundgang zu den einzelnen HTML-Seiten gelangen. Als Index wird automatisch das erste Text-Attribut des Inhaltes der Guided Tour verwendet.

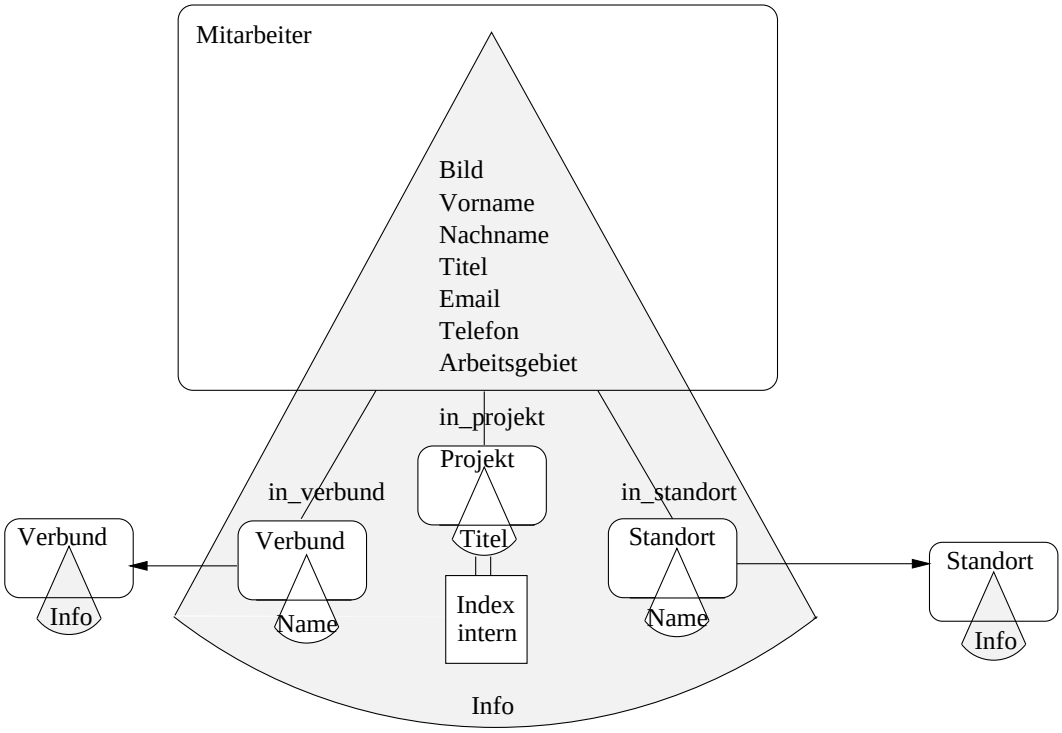


Abbildung 5.5: Navigational-Design von Mitarbeitern

Ein Hyperlink besteht aus einem Quellanker und einem Ziel, wobei das Ziel ein Hypertext-Knoten ist. Quellanker und Zieldokument werden in der RMM durch M-Slices mit dem gleichen Owner-Entity modelliert (z.B. `Standort.Name` und `Standort.Info`). Alle Beziehungen zwischen Slices werden auf Beziehungen zwischen den beteiligten Owner-Entities des ER-Modells abgebildet. Ein Slice kann demnach nur mit einem anderen Slice in Beziehung stehen, wenn die zugehörigen Owner-Entities direkt oder indirekt (über mehrere Relationships) in Beziehung stehen. Zu jeder Zugriffsstruktur wird die Bezeichnung der Beziehung angegeben. Direkte und indirekte Beziehungen werden durch eine durchgezogene bzw. unterbrochene Linie dargestellt.

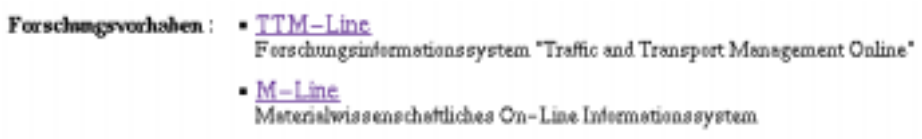
Abbildung 5.5 zeigt die um Zugriffsstrukturen ergänzte Modellierung eines Mitarbeiters. Neben den Attributen des Entities *Mitarbeiter* beinhaltet der Slice noch jeweils einen Hyperlink auf den Standort und den zugehörigen Forschungsverbund. Als Link-Anker wird einmal der Slice *Name* von Owner-Entity *Standort* und das andere Mal der Name des Forschungsverbundes (Slice *Name* von Owner-Entity *Verbund*) angezeigt. `Standort.Name` ist ein Aggregat aus Name, PLZ und Ort des Standorts. Die beiden Entities *Verbund* und *Standort* stehen mit dem Owner-Entity *Mitarbeiter* in einer 1:n-Beziehung.

Standort : FORWISS München, 81667 München

Verbund : FORWISS

Jeder Mitarbeiter kann an verschiedenen Projekten arbeiten. Diese n:m-Beziehung zwischen den Owner-Entities *Mitarbeiter* und *Projekt* wird durch die Zugriffsstruktur *In-*

dex Intern dargestellt. Der Slice `Projekt.Titel` beinhaltet neben dem Titel eines Projektes auch die genaue Bezeichnung des Forschungsvorhabens. Nur der Titel des Projektes soll dabei mit einem Hyperlink unterlegt sein. Der Slice `Projekt.Titel` ist deshalb kein Ankerslice, wie beispielsweise Standort oder Verbund.



Die verschiedenen Modellierungsvarianten von Projekten und Standorten sollen durch die Abbildungen 5.6 und 5.7 nochmal verdeutlicht werden. Die hierzu gehörenden Slice-Typen werden durch Anhang B veranschaulicht. Abbildung 5.6 zeigt drei verschiedene Varianten zur Modellierung eines Standorts auf einer Mitarbeiter-Seite. Alternative 1 zeigt einen Hyperlink-Anker, der aus drei Attributen des Entities *Standort* (Name des Standorts, Postleitzahl und Ort) aufgebaut ist. Alternative 2 zeigt einen Slice, der sich aus einem Hyperlink-Anker (Standort-Name) und zwei Attributen (Postleitzahl und Ort) zusammensetzt. Alternative 3 veranschaulicht einen geschachtelten Slice. Die Attribute Name, Straße, Postleitzahl und Ort werden innerhalb des Mitarbeiter-Slices angezeigt.

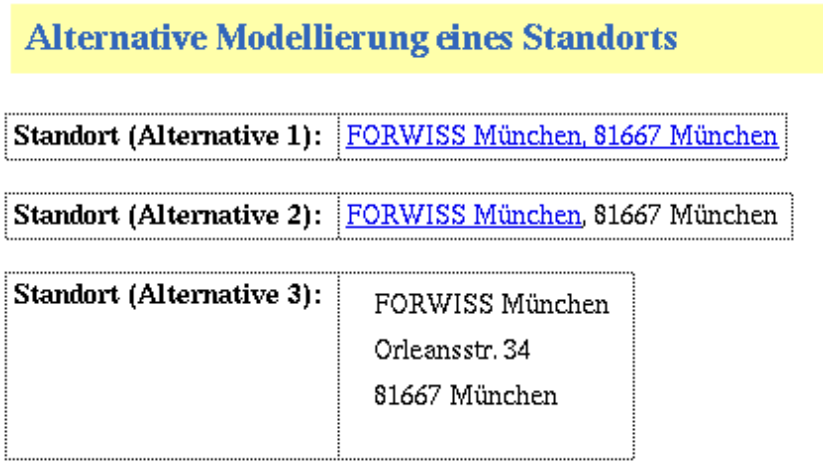


Abbildung 5.6: Alternative Modellierung eines Standorts

Abbildung 5.7 zeigt die verschiedenen Modellierungsvarianten von Zugriffsstrukturen anhand von Projekten. Alternative 1 veranschaulicht eine Slice-Liste, wobei der Slice aus einem Anker und einem weiteren Attribut aufgebaut ist. Alternative 2 entspricht einer Hyperlink-Liste. Der Inhalt der Zugriffsstrukur ist ein Hyperlink. Alternative 3 zeigt den Zugriff auf eine externe Liste. In dem Beispiel Abbildung 5.5 wurde für die Modellierung des Standortes und der Projekte jeweils die Alternative 1 gewählt.

Abschließend sollen die wichtigsten Merkmale von M-Slices nochmal zusammengefaßt und formal beschrieben werden. Die präzise Definition des erweiterten RMDM ermöglicht die automatisierte Befüllung des Meta-Schemas (siehe Kapitel 5.7.2). Die konkrete Syntax

Abbildung 5.7: Alternative Modellierung einer Projektliste

der Programm-Spezifikationssprache wird in BNF beschrieben und ist an die in [IKK97] vorgestellte Syntax angelehnt:

- Jeder *M-Slice* ist genau einmal definiert und eindeutig einem Owner-Entity zugeordnet. Die Meta-Notation eines M-Slices lautet wie folgt:
  ⟨m-slice⟩ :== ⟨owner-entity⟩.⟨slice⟩
  ⟨owner-entity⟩ :== (Name des Owner-Entities)
  ⟨slice⟩ :== (Name des Slices)
  Die Notation zur Definition eines M-Slices sieht wie folgt aus:
  ⟨m-slice⟩**: m-slice**
  begin
  ⟨body⟩;
  end;

- Der *Inhalt* (body) jedes M-Slices ist eine Menge möglicherweise leerer *Elemente*. Jedes Element ist entweder:

  – ein *Attribut*
  – ein *M-Slice*
  – ein *Hyperlink* oder
  – eine *Zugriffsstruktur*

  ⟨body⟩ :== {⟨attribute⟩|⟨m-slice⟩|⟨hyperlink⟩|⟨access_structure⟩}

- Ein *Attribut* stammt entweder direkt aus dem Owner-Entity des M-Slices oder von einem in Beziehung stehenden Entity. Im zweiten Fall bildet ein anderer Slice eine Komponente dieses M-Slices:

⟨attribute of an entity⟩ :== ⟨attribute⟩
⟨attribute⟩ :== (Name des Attributs)
⟨m-slice of an entity⟩ :== [⟨relation⟩]→ ⟨m-slice⟩
⟨relation⟩ :== (Bezeichnung des Relationship-Typs)

- Ein *Hyperlink* besteht aus einem Anker und einem Ziel, wobei Anker und Ziel M-Slices sind. Der Anker ist der Bereich, der als Ausgangspunkt für einen Link verwendet wird (anklickbarer Bereich). Das Ziel ist eine HTML-Seite, die eine Instanz des Zielslices ist. Es ist nicht möglich, an eine bestimmte Stelle in einem anderen Dokument zu navigieren, sondern nur auf das gesamte Zieldokument zu verweisen:
  ⟨hyperlink⟩ :== [⟨relation⟩]→*⟨anchor⟩ ⇒ ⟨destination⟩
  ⟨anchor⟩ :== ⟨m-slice⟩
  ⟨destination⟩ :== ⟨m-slice⟩
  Im Falle einer Intra-Record-Navigation (siehe Kapitel 7.1.1) wird die [relation] mit *this* bezeichnet.

- Eine *Zugriffsstruktur* ist ein externer Index, interner Index, eine Guided Tour oder eine Indexed Guided Tour. Jede Zugriffsstruktur wird durch die *Beziehung* (relation) zwischen den beteiligten Owner-Entities und den *Inhalt* beschrieben. Der Inhalt einer Zugriffsstruktur ist ein M-Slice oder ein Hyperlink:
  ⟨access_typ⟩ :== **index intern** | **index extern** | **guided tour** | **indexed tour**
  Die Notation zur Definition einer Zugriffsstruktur lautet:
  ⟨access_structure⟩ :== [⟨relation⟩] → ⟨access_typ⟩ (⟨m-slice⟩|⟨hyperlink⟩)

Angewendet auf das Beispiel eines Mitarbeiters, ergibt sich folgende Definition des Mitarbeiter-Slices. Die zugehörige HTML-Seite wird durch Abbildung 5.18 verbildlicht:

```
Mitarbeiter.Info: m-slice
begin
     Bild;
     Vorname;
     Nachname;
     Titel;
     Email;
     Telefon;
     Arbeitsgebiet;
     [in_standort] ⟶ *Standort.Name ⟹ Standort.Info;
     [in_verbund]  ⟶ *Verbund.Name ⟹ Verbund.Info;
     [in_projekt]  ⟶ index intern (Projekt.Titel);
end;
```

5.3.4 Application-Design

Das *Application-Design* ermöglicht eine globale Sicht auf die Hypertext-Struktur auf der Ebene von HTML-Seiten (im Unterschied zu Slice-Typen) und gewährleistet so die Abbildung der gesamten Web-Site. Es entspricht demnach der Modellierung von Presentation Units aus der RMM. Das Hauptkonstrukt bei der Modellierung bilden HTML-Seitentypen, die eine homogene Menge gleich strukturierter HTML-Seiten beschreiben. Eine HTML-Seite entspricht einem Objekt mit einem eindeutigen Identifikator (URL) und verschiedenen Attributen, die einfache (z.B. Text, Bild) oder komplexe Typen (z.B. Listen) haben. Eine HTML-Seite bildet eine Instanz eines HTML-Seitentyps. Die RMM bietet zwar umfangreiche Modellierungskonzepte für das konzeptuelle Design von Web-Sites, unterstützt aber nur ein eingeschränktes Application-Design auf der Basis eines seitenorientierten Datenmodells, wie es dem Web zugrunde liegt. Die im folgenden erläuterten Modellierungskonzepte stützen sich deshalb im wesentlichen auf [ED99], [BN96] und [AMM97b] ab.

Jeder HTML-Seitentyp wird durch ein Rechteck verdeutlicht, das die Attribute des Seitentyps beinhaltet. Zugriffsstrukturen, wie z.B. Listen, werden durch einen Balken auf der linken Seite veranschaulicht (siehe Abbildung 5.8). Hyperlinks werden durch eine durchgezogene Linie dargestellt.

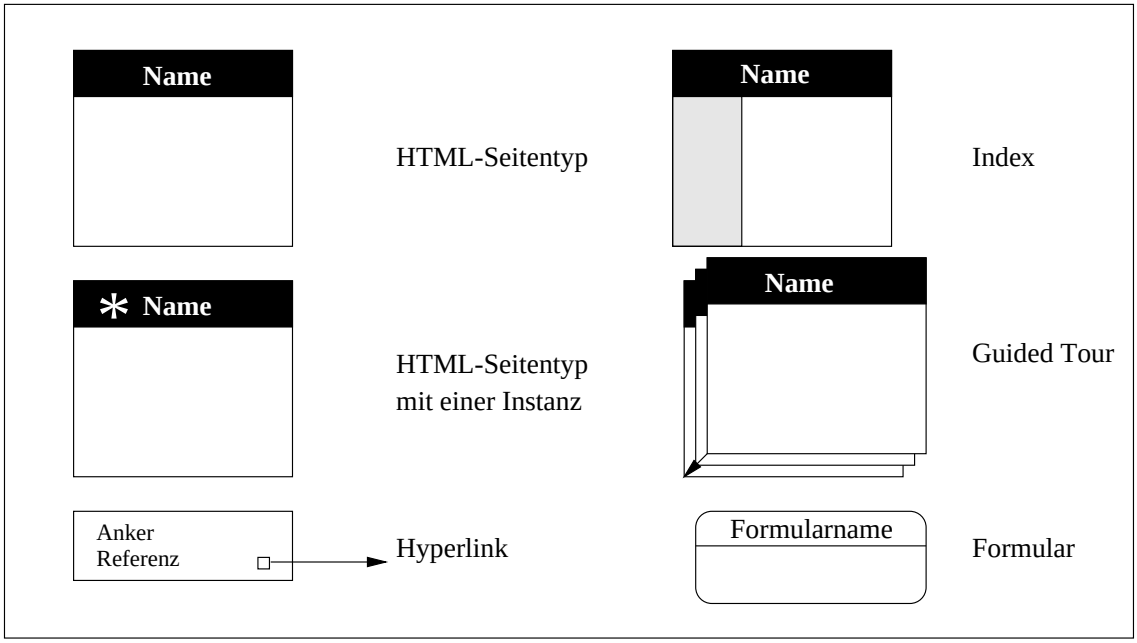


Abbildung 5.8: Application-Design der HDBM

Die Modellierung des Forschungsverbundes mit Hilfe eines Application-Diagramms wird durch Abbildung 5.9 dargestellt. Von der Verbund-Homepage gelangt man zu den drei Bereichen: Standorte, Mitarbeiter und Forschungsprojekte. Auf alle drei Bereiche wird über einen Index zugegriffen. Zwischen den drei Bereichen kann man durch Hyperlinks oder Hyperlink-Listen navigieren.

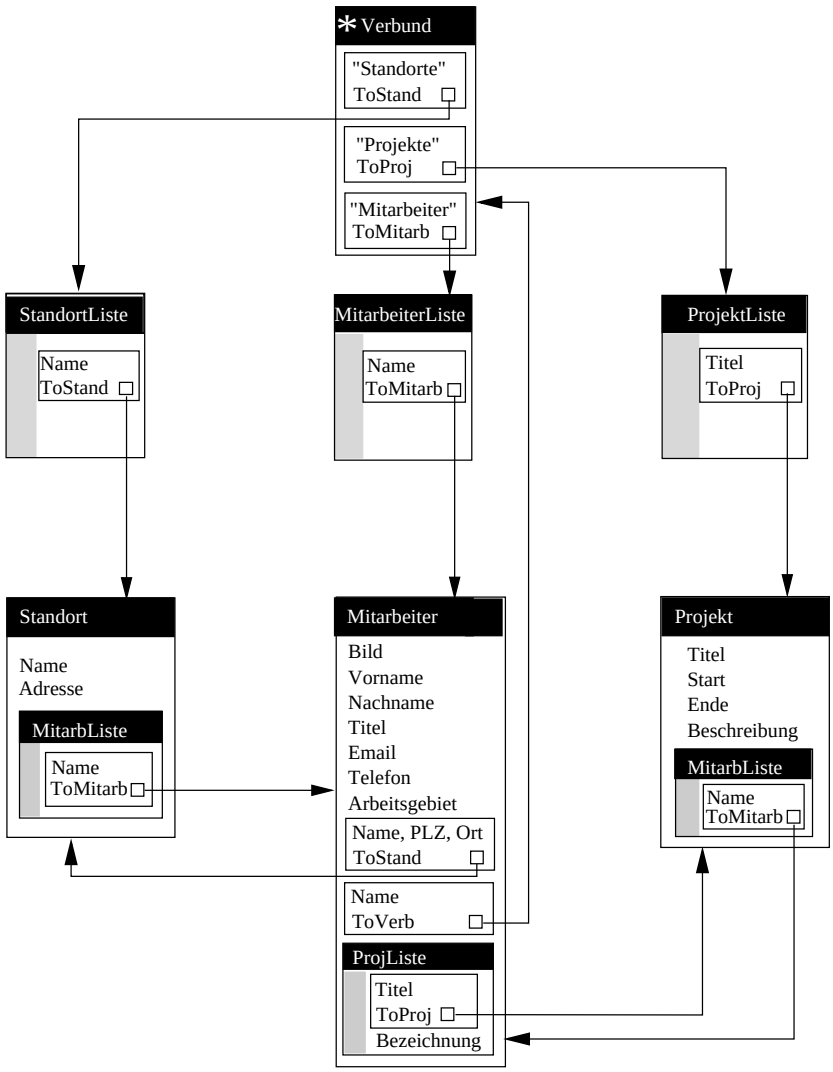


Abbildung 5.9: Application-Diagramm von einem Forschungsverbund

Betrachtet man das seitenorientierte Datenmodell als ein logisches Hypertext-Modell, so ist es möglich auf dieser Basis, ähnlich der relationalen Algebra eine *navigationale Algebra* zu definieren [MMM98]. Neben den traditionellen Operatoren *Selektion*, *Projektion*, und *Join*, werden zwei einfache neue Operatoren ergänzt, um die Navigation zwischen HTML-Seiten zu beschreiben: der *unnest page*-Operator und *follow link*-Operator.

- *unnest page* $S \diamond A$ ist ein binärer Operator, der als Eingabe einen geschachtelten Seitentyp S und ein Attribut A nimmt. Er ermöglicht den Zugriff auf Daten unterschiedlicher Schachtelungstiefe auf einer HTML-Seite.

- *follow link* $S_1 \longrightarrow S_2$ ist ein binärer Operator, der als Eingabe zwei Seitentypen S_1 und S_2 nimmt, so daß ein Hyperlink von S_1 auf S_2 existiert.

Im Verlauf der Arbeit werden beide Operatoren verwendet, um die korrespondierenden Abbildungen zwischen Relationen und Seitentypen zu formalisieren.

5.4 Integrationsmethode

Im Verlauf des logischen Hypertext-Designs erfolgt die Abbildung des konzeptuellen Hypertext-Modells auf eine logische Ebene. Liegt der entworfenen Web-Site keine Datenbank zugrunde werden die HTML-Seiten mit Hilfe von HTML-Editoren oder CASE-Tools erzeugt und im File-System abgelegt. Eine Trennung zwischen logischer und physischer Ebene erfolgt nicht. Bei der Anbindung einer Datenbank an das Web gestaltet sich die logische Umsetzung des Hypertext-Designs etwas schwieriger, da die HTML-Seiten zur Laufzeit dynamisch generiert werden und die Ergebnisse von Datenbankanfragen beinhalten. Für die Speicherung der hierfür benötigten Informationen über HTML-Seitenlayout und Navigationsstruktur der Web-Site existieren verschiedene Ansätze:

1. **Programme, Skripten, HTML-Templates**
   Die Generierung der HTML-Seiten wird in Programmen bzw. Skripten oder Templates fest codiert und innerhalb der Datenbank (z.B. stored procedures) oder außerhalb der Datenbank abgelegt.

2. **Definition von Hypertext-Views:**
   Die Umsetzung des konzeptuellen Hypertext-Designs erfolgt mit einem logischen Datenmodell, zu dem eine Definitionssprache (DDL) und Anfragesprache (DML) definiert werden.

3. **Meta-Daten:**
   Zu jeder Relation (bzw. Klasse) des Datenbankschemas werden zusätzliche Informationen bezüglich Layout, Navigation und Hypertext-Struktur in einem Meta-Schema abgelegt, das als Erweiterung des Systemkatalogs der Datenbank dient. Mit Hilfe verschiedener Tools wird das Meta-Schema gefüllt und die Hypertext-Anwendung generiert. Dieser Ansatz wurde in HDBM verfolgt.

Abhängig von dem gewählten Ansatz zur logischen Modellierung von Hypertext ändert sich der Ablauf der zugehörigen Entwurfsphasen. Durch Abbildung 5.10 sollen die verschiedenen Entwurfsprozesse für datenbankbasierte Web-Anwendungen verbildlicht werden. Die Pfeile sind dabei entsprechend den drei oben vorgestellten Ansätzen von eins bis drei durchnumeriert.

Die verschiedenen Varianten zur logischen Umsetzung des konzeptuellen Hypertext-Schemas sollen im folgenden genauer erläutert werden.

5.4.1 Programme, Skripten, HTML-Templates

Die erste Alternative entspricht der gängigen Integrationsmethode zur Anbindung von Datenbanksystemen an das Web. In Skripten, Programmen oder HTML-Templates werden die auszuführenden DB-Anweisungen und das Layout der zu erstellenden HTML-Seiten fest codiert. Für die Definition der möglichen Anfragen und die Festlegung des zugehörigen Seitenlayouts gibt es verschiedene Methoden:

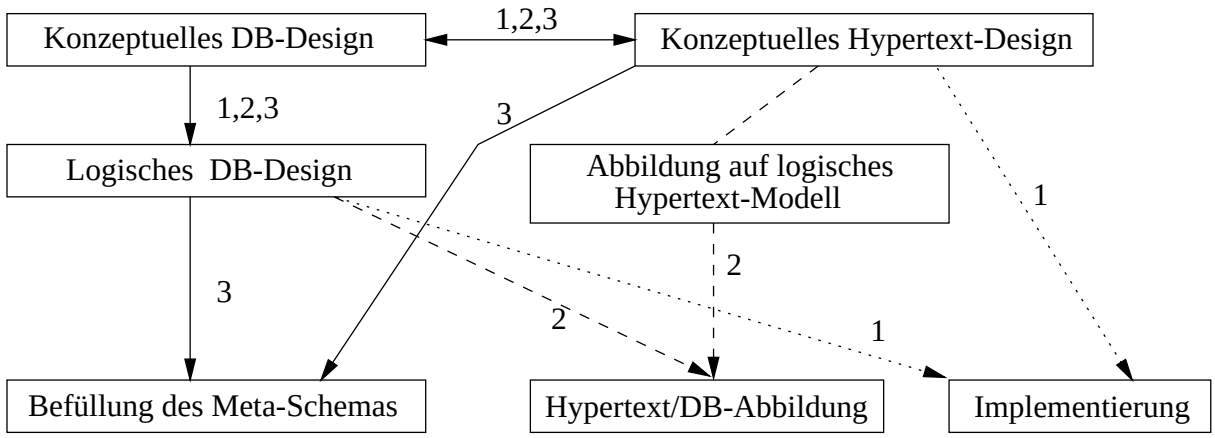


Abbildung 5.10: Verschiedene Abläufe des datenbankgestützten Hypermedia-Designs

- Stored Procedures
- Hard-Coded-Gateway
- URL-embedded Queries
- HTML-Masken/Makros

Stored Procedures sind Programme, die von vielen Datenbank-Servern, wie beispielsweise Oracle [HB98a], unterstützt werden. Bei Stored Procedures werden die möglichen Anfragen als modulare Programmteile geschrieben und auf Server-Seite gespeichert, compiliert und gestartet. Der Browser muß keine Anfrage sondern nur den Identifikator der entsprechenden Prozedur und die gewünschten Parameterwerte übergeben. Die großen Vorteile von Stored Procedures liegen in der Performanz und der Sicherheit. Der Anwender hat ausschließlich Ausführungsrechte bereits vordefinierter Anfragen. Dadurch besteht keine Gefahr, daß ein Benutzer unerlaubte Datenbank-Aktionen ausführt.

Die Anfragen können auch fest in entsprechenden Skripten bzw. Programmen codiert werden, die im normalen Dateisystem abgelegt sind und bei einer Anfrage aufgerufen und mit Parametern versorgt werden (*Hard-Coded-Gateway*). Nachteile dieser Architektur sind die schwere Wartbarkeit und Skalierbarkeit der DB-Anbindung.

Bei *URL-embedded-Queries* wird die Datenbankanfrage an den URL angehängt und über den Web-Server an die Datenbank übergeben. Regeln zur Erzeugung des HTML-Codes können beispielsweise in objektorientierten Systemen als Methoden für jede darzustellende Datenklasse abgelegt werden. Diese Art der Anfrageübergabe wurde bei der O2 Web-Architektur [O2 96] verwendet. Bei komplexen Anfragen können die URLs bei dieser Übergabetechnik sehr lang werden.

Bei der vierten Variante werden die Datenbank-Anfragen in *HTML-Masken* oder *HTML-Templates* eingebettet. HTML-Templates bezeichnen dabei HTML-Vorlageseiten, die um Datenbankoperationen ergänzt wurden. Die Syntax der Anfragen ist durch eine entsprechende Grammatik definiert, welche den gewünschten Befehlsumfang bietet. Bei

einer Datenbankanfrage analysiert bzw. übersetzt das Gateway die zugehörige HTML-Maske, reicht die Anfragen an die Datenbank weiter und fügt die Anfrageergebnisse wieder in die HTML-Maske ein. Die so erstellte HTML-Seite wird vom Web-Server zum Web-Browser übergeben und dort dem Benutzer angezeigt. Diese Architektur wird beispielsweise bei dem 'Web Data Blade Module' von Informix [Inf99] eingesetzt.

Bei 'DB2 WWW Connection Version 1' [NS96] werden für die Definition der Anfragen und die Bestimmung des Seitenlayouts sogenannte *Makro-Dateien* verwendet, die in vier Sektionen zur Definition der Variablen, der SQL-Anfragen, des HTML-Eingabeformulars und des HTML-Anfrageergebnisses eingeteilt sind.

Die Verwendung proprietärer Schnittstellen zwischen Datenhaltung und Hypertext-Applikation bringt die bekannten Nachteile mit sich. Darunter fallen schlechte Lesbarkeit, aufwendige Erstellung und schlechte Wartbarkeit. Ad-hoc-Anfragen an das Datenbanksystem sind nur begrenzt möglich.

5.4.2 Definition von Hypertext-Views

Bei dieser Variante werden über eine bestehende Datenbank sogenannte Hypertext-Views mit Hilfe einer geeigneten Hypertext-View-Definitionssprache definiert. Die Idee von Hypertext-Views entspricht in vielen Punkten der traditioneller Datenbank-Views. Über einen Datenbestand soll eine externe Sichtweise auf die Daten gewährleistet werden, die auf die speziellen Anforderungen einer Nutzergruppe abgestimmt ist. Dabei können Anfragen über Views effizienter und einfacher in ihrer Formulierung sein als eine Anfrage über den gesamten Datenbestand. Im Unterschied zu herkömmlichen Datenbank-Views definieren Hypertext-Views aber keine Datenbank-Elemente (Klassen, Relationen) sondern Hypertext-Knoten und Hypertext-Links. Sie schließen damit die Kluft zwischen dem Datenbank-Modell auf der einen Seite und dem Hypertext-Modell auf der anderen Seite. Eine Hypertext-View basiert im relationalen Fall entweder auf einer Datenbank-Relation oder einer Datenbank-View, durch die alle für eine HTML-Seite relevanten Attribute bestimmt werden. Die Hypertext-View-Definitionssprache ermöglicht die Definition von Struktur, Navigation und soweit möglich des Layouts einer HTML-Seite und basiert auf einem logischen Hypertext-Modell.

Ein Beispiel hierfür ist die Generierung von Hypertext mit Hilfe des im ARANEUS-Projekt entwickelten Seitengenerators PENELOPE [AMM97b]. PENELOPE ermöglicht die Generierung von HTML-Seiten aus Datenbankinhalten und die Abbildung von Relationen auf Hypertext-Views. Die Seiten-Definitionssprache basiert auf dem ARANEUS Datenmodell (ADM) (siehe auch Kapitel 3.2.1). Die Definition von Seitenschemata erfolgt mit Hilfe der DEFINE PAGE-Anweisung. Die Syntax einer DEFINE PAGE Anweisung sieht dabei wie folgt aus:

```
DEFINE PAGE P [UNIQUE]
AS           S
FROM         R
```

wobei: (i) P das Seitenschema, (ii) R die Relation oder Datenbank-View und (iii) S die Seitenstruktur der HTML-Seite beschreibt. UNIQUE ist ein Schlüsselwort und definiert, daß ein Seitenschema nur eine einzige Instanz hat.

Für das Beispiel einer Forschungsgruppe würde die Hypertext-View dann wie folgt aussehen [AMM97a]:

```
DEFINE PAGE RESEARCH_GROUP_PAGE
AS URL          URL (<GroupName>);
   GName:       TEXT <GroupName>;
   TopicList:   LIST OF (Topic:    TEXT   <Topic>;)
   MemberList:  LIST OF (LINK TO PROFESSOR_PAGE UNION STUDENT_PAGE
                        (Name: TEXT          <MemberName>;
                         ToMemberPage:     REF-TO URL(<MemberName>)));
FROM RESEARCH_GROUP_PAGE_VIEW
IN DEPTDB
```

wobei RESEARCH_GROUP_PAGE_VIEW eine relationale DB-View über den Datenbestand der Datenbank DEPTDB ist. URLs werden mit Hilfe eines Schlüssels generiert, der in Klammern hinter dem URL steht(z.B. `(<GroupName>)`). Das korrespondierende Attribut der Datenbank-Relation steht in spitzen Klammern hinter dem Datentyp.

Ein anderer Ansatz, der allerdings nicht die Möglichkeiten der ARANEUS-Definitionssprache bietet, wird in [FGP96] verfolgt. Eine Hyperview-Definitionssprache ermöglicht die Spezifikation von Hypertext-Struktur und -Layout auf der Basis des Oracle-Datenbanksystems. Durch einen *Node Server* werden die um Hypertext-Informationen angereicherten Datendefinitionen in SQL-Anfragen umgeformt und im Oracle-Datenbanksystem abgesetzt. Die Anzeige des Anfrageergebnisses erfolgt anschließend als HTML-Seite im Web-Browser. Auf Datenbankseite ist keinerlei Programmierung erforderlich.

Folgendes Beispiel soll diesen Ansatz verdeutlichen. Die Spezifikation eines HTML-Knotens in der Hyperview Definition Language sieht wie folgt aus [FGP96]:

```
node users
     display bold(username) break, ``user_id:  `` user_id
     selected by user_group
     ordered by username
from sys.dba_tables
```

Dabei ist `sys.dba_tables` entweder eine Datenbank-Relation oder eine Datenbank-View.

Übersetzt in eine Oracle-SQL-Anfrage sieht die DDL wie folgt aus:

```
create view http_users as
select '<b>' || username || '</b> || ' user_id: ' || user_id
               ndisplay  ,
```

```
        user_group n_select ,
        username n_order
from sys.dba_tables
```

wobei || der concatenation-Operator in Oracle ist. Ein weiterer Ansatz hierzu ist das web-at-a-glance-System (WAG) [CINS97].

Mit Hilfe einer Hypertext-View-Definitionssprache ist es möglich, über einen relationalen Datenbestand verschiedene Hypertext-Sichten zu definieren. Die Mächtigkeit der Hypertext-Definitionssprache und des zugrunde liegenden Hypertext-Modells variiert dabei stark. Es ist demnach kaum möglich, alle für die Generierung einer HTML-Seite notwendigen Informationen (Zugriffsstrukturen, Web-Typen, Layout), durch eine View-Definitionssprache zu definieren. Weiterhin ist die Definition der einzelnen Seitentypen teilweise sehr umfangreich und erfordert zunächst die Definition geeigneter Datenbank-Views, durch die alle Attribute einer HTML-Seite aus verschiedenen Relationen selektiert werden. Es ist außerdem fraglich inwieweit die Integration von Layout-Informationen, wie beispielsweise Bezeichnung, Position und Web-Typ eines Attributs, innerhalb einer View-Definition sinnvoll ist. Da das WWW ein grafisch orientiertes Hypertext-System ist, spielt aber gerade dies eine wichtige Rolle. Da Meta-Daten zu den Hypertext-Views nicht innerhalb der Datenbank abgelegt werden, ist die Nutzung und Wiederverwendbarkeit dieser Daten für die Ad-hoc-Anfragebearbeitung und inkrementelle Wartung von materialisierten Views nur beschränkt möglich.

5.4.3 Meta-Daten

Neben dem Data Dictionary, das Meta-Daten über den aktuellen Status der Datenbank enthält, werden weitere Meta-Daten zu Hypertext-Struktur, Layout und Navigation der Hypertext-Applikation in der Datenbank abgelegt. Diese Daten dienen zur dynamischen Generierung von HTML-Seiten, können aber auch bei der Ad-hoc-Anfragebearbeitung über HTML-Seiten und zur inkrementellen Wartung materialisierter HTML-Seiten genutzt werden.

In einem an der Wirtschaftsuniversität Wien entwickelten Prototypen [PM96] werden in einem Repository abgelegte Meta-Daten genutzt, um aus einer relationalen Datenbank Hypertext zu generieren. Dabei ist nur die Generierung sehr einfacher HTML-Seiten möglich. Ein HTML-Seitentyp entspricht einer DB-Tabelle. Die Gruppierung von Attributen einer Tabelle in Komponenten, die auf verschiedenen HTML-Seitentypen angezeigt werden, ist nicht möglich. Weiterhin können Informationen, die bei der Umsetzung von n:m-Relationships über mehrere Tabellen normalisiert wurden nicht auf eine HTML-Seite abgebildet werden. Als Zugriffsstrukturen werden nur einfache Hyperlinks und Hyperlink-Listen unterschieden.

In dem von Geert-Jan Houben und Paul De Bra entwickelten Ansatz [HB98b], [HB99] werden die Informationen des Data Dictionary eines relationalen Datenbanksystems für die semi-automatische Generierung von Hypertext genutzt. Einige wenige Informationen,

wie beispielsweise die Zugriffsstruktur auf eine Relation und das Layout, werden ebenfalls im Data Dictionary abgelegt, um eine Default-Darstellung der Relationen auf Hypertext-Ebene zu ermöglichen. Der Schwerpunkt des Ansatzes liegt aber auf der dynamischen Generierung beliebiger HTML-Seiten als Ergebnis von Ad-hoc-Anfragen, die mit einer speziell hierfür entwickelten Anfragesprache formuliert werden. Die SQL-basierte Anfragesprache ermöglicht es, die gewünschte Hypertext-Struktur und Präsentation des Anfrageergebnisses in die Query-Spezifikation einzubetten. Bei der Generierung der HTML-Seiten werden Informationen aus dem Data Dictionary der Datenbank (Fremdschlüssel, Primärschlüssel, Attribute, Tabellen ...) und der Anfragespezifikation (Zugriffsstruktur, Slices ...) verwendet.

In [FHN97] werden ebenfalls Meta-Daten für die Entwicklung von Hypermedia-Anwendungen eingesetzt. Syntax und Semantik des Meta-Modells werden durch die Telos-Sprache spezifiziert und durch ein deduktives DBMS ConceptBase interpretiert.

Es existieren einige wenige Ansätze, die das Data Dictionary zusammen mit anderen Meta-Daten für die dynamische Generierung von HTML-Seiten aus einer Datenbank nutzen. Dabei ist nur eine einfache Abbildung der Datenbank-Inhalte auf HTML-Seitenebene vorgesehen. Die Generierung komplexer HTML-Seiten, die aus Attributen verschiedener Relationen aufgebaut sind und vielfältige Querbeziehungen zu anderen HTML-Seiten ausbilden, wird durch die Meta-Daten alleine nicht unterstützt. Hierzu ist der Entwurf eines komplexen Meta-Schemas auf der Basis eines mächtigen Hypertext-Modells erforderlich, das es erlaubt Informationen über Struktur, Navigation und Layout der HTML-Seiten abzulegen. Dieser Ansatz wurde in HDBM verfolgt.

Basierend auf dem RMM-Datenmodell wurde ein umfangreiches Meta-Modell zur Ablage von Hypertext-Meta-Daten entwickelt, das die Generierung komplexer HTML-Seiten ermöglicht. Die einzelnen Komponenten des Meta-Schemas werden in den nächsten Kapiteln genauer vorgestellt.

5.5 Das Meta-Modell

Das HDBM-Meta-Modell dient zur Erfassung von Navigationsstruktur, Präsentation und Strukturierung einer Hypertext-Anwendung. Ähnlich dem Data Dictionary, das Meta-Daten über den aktuellen Status der Datenbank enthält, umfaßt das Hypertext-Meta-Schema von HDBM nicht die Anwendungsdaten selbst, sondern nur die Schema-Informationen des Hypertext-Modells. Zu dem vom Datenbanksystem automatisch verwalteten Systemkatalog, werden zur Erzeugung von Hypertext spezifische Meta-Daten ergänzt. Hierzu gehören beispielsweise Titel der HTML-Seite, Zugriffsstrukturen, Anzeigereihenfolge der Attribute, Bezeichnung des Attributs und Gruppierung der Attribute.

Das Anwendungsspektrum des Meta-Modells reicht von der Generierung ergonomischer Web-Sites bis hin zu erweiterten Hypertext-Anfragemöglichkeiten. Dabei ist es mit Hilfe des Meta-Modells in den meisten Fällen möglich, mehr als 80% einer Web-Site automatisiert zu generieren. Der Rest kann durch individuelle, fachspezifische Erweiterungen

ergänzt werden. Dies unterstützt ein *Rapid Application Development*, das gerade für Anwendungen im Web sehr wichtig ist.

Das Meta-Modell von HDBM zusammen mit dem Data Dictionary der Datenbank läßt sich zusammen in folgende Komponenten unterteilen (siehe Abbildung 5.11):

- **Anwendungsdomänen-Meta-Modell:** beschreibt die Anwendungsdomäne und modelliert die Basis-Relationen und Beziehungen des Anwendungsbereichs auf einer Meta-Ebene.
- **Slice-Meta-Modell:** modelliert die Abbildung des Anwendungsbereichs auf Hypertext-Knoten.
- **Navigations-Meta-Modell:** modelliert die Abbildung zwischen Relationships der Anwendungsdomäne auf Hyperlinks und Zugriffsstrukturen einer Web-Site.
- **Präsentations-Meta-Modell:** modelliert das HTML-Layout der verschiedenen Hypertext-Knoten.

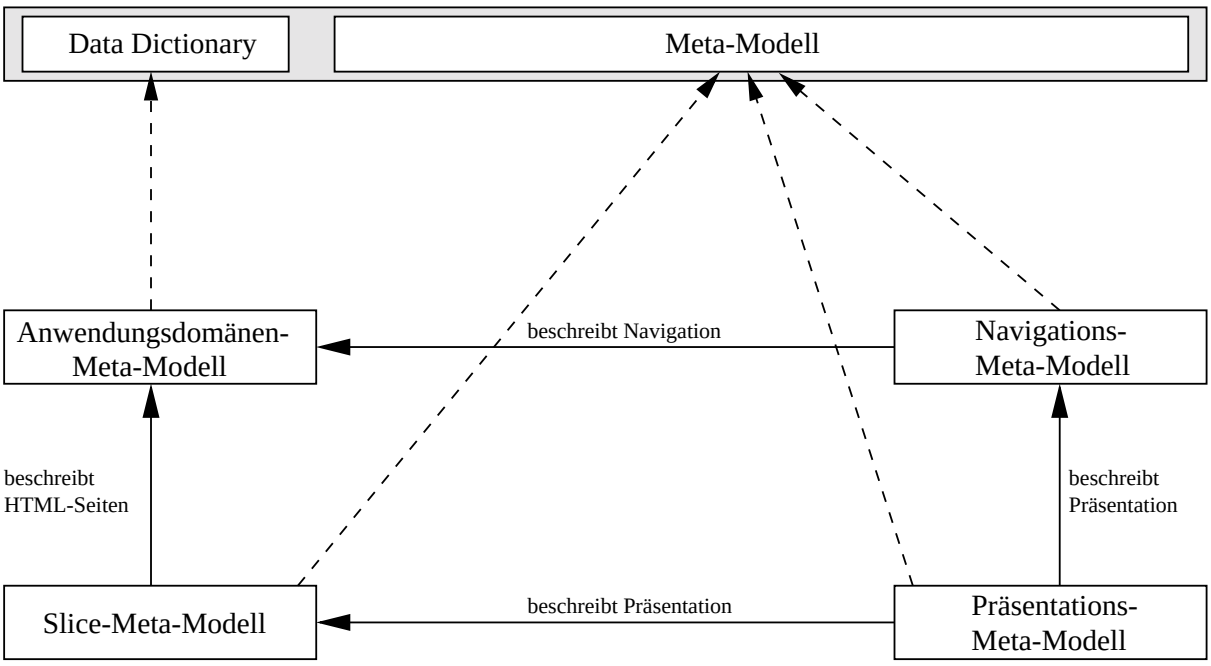


Abbildung 5.11: Komponenten des Meta-Modells

Im folgenden Kapitel werden die einzelnen Segmente des Meta-Modells genauer beschrieben und mit Hilfe der OMT [RBP+91] veranschaulicht. Eine genaue Beschreibung und Auflistung der einzelnen Attribute und Relationen erfolgt im Verlauf des relationalen Designs.

5.5.1 Anwendungsdomänen-Meta-Modell

Die Modellierung der Anwendungsdomäne erfolgt auf der Basis des ER-Modells, das im wesentlichen aus Entity-Typen, Attributen und Beziehungen zwischen Entity-Typen aufgebaut ist. Vereinfacht läßt sich das ER-Modell auf einer Meta-Ebene mit Hilfe der OMT durch Abbildung 5.12 veranschaulichen. Die Elemente des ER-Modells werden durch *Entities*, *Attribute* und *Relationships* zwischen zwei Entities modelliert. Relationships lassen sich weiter in 1:1-Relationships, 1:n- und n:m-Relationships unterscheiden.

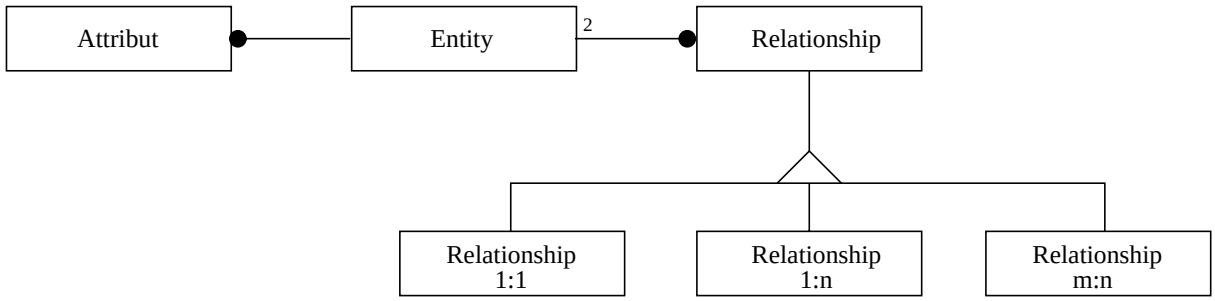


Abbildung 5.12: Vereinfachtes ER-Meta-Modell

5.5.2 Slice-Meta-Modell

Slices bilden das zentrale Modellierungskonzept des RMDM und gruppieren zusammengehörige Attribute, die auf einer HTML-Seite angezeigt werden sollen.

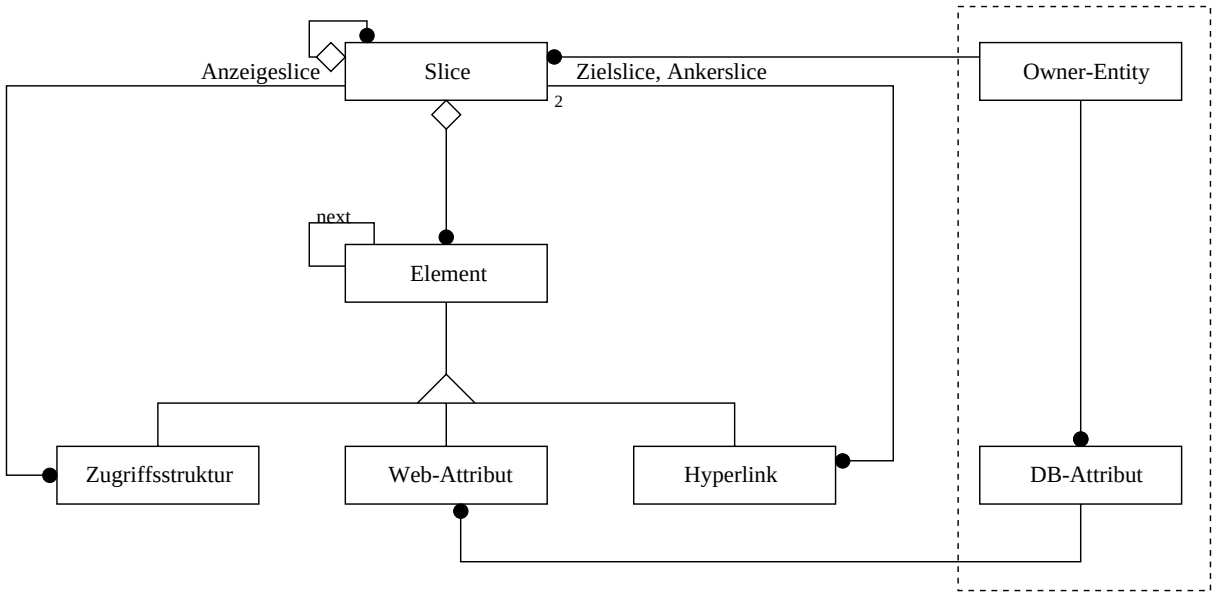


Abbildung 5.13: Slice-Meta-Modell

Jeder Slice ist eindeutig einem Entity zugeordnet. Dieses Entity wird als Owner-Entity bezeichnet. Jedes Entity kann dagegen verschiedenen Slices zugeordnet sein. Dies

tritt beispielsweise dann ein, wenn die Attribute eines Entities über verschiedene HTML-Seiten verteilt sind oder in unterschiedlichem Kontext verschieden angezeigt werden sollen. Ein Slice-Typ steht deshalb mit einem Owner-Entity in einer 1:n Beziehung. Ein Slice besteht aus verschiedenen Elementen, die in Attribute, Zugriffsstrukturen und Hyperlinks unterschieden werden. Jedes Web-Attribut ist genau einem Attribut des Owner-Entities zugeordnet. Ein Attribut eines Entities kann durch verschiedene Web-Attribute modelliert werden (siehe Abbildung 5.13). Weiterhin kann ein M-Slice rekursiv aus anderen M-Slices aufgebaut sein.

5.5.3 Navigations-Meta-Modell

Die Verknüpfung verschiedener Informationen mit Hilfe von Referenzen (Links) ermöglicht die nicht-lineare Repräsentation von Wissen im World Wide Web. Dabei erfolgt die Navigation durch Hypertext-Knoten mit Hilfe von Hyperlinks und verschiedenen Zugriffsstrukturen (Index, Guided Tour), die auf Datenbankebene durch Relationships zwischen Entities modelliert werden. Abbildung 5.14 veranschaulicht 1:1-, 1:n- und m:n-Relationships und deren Abbildung auf Zugriffsstrukturen.

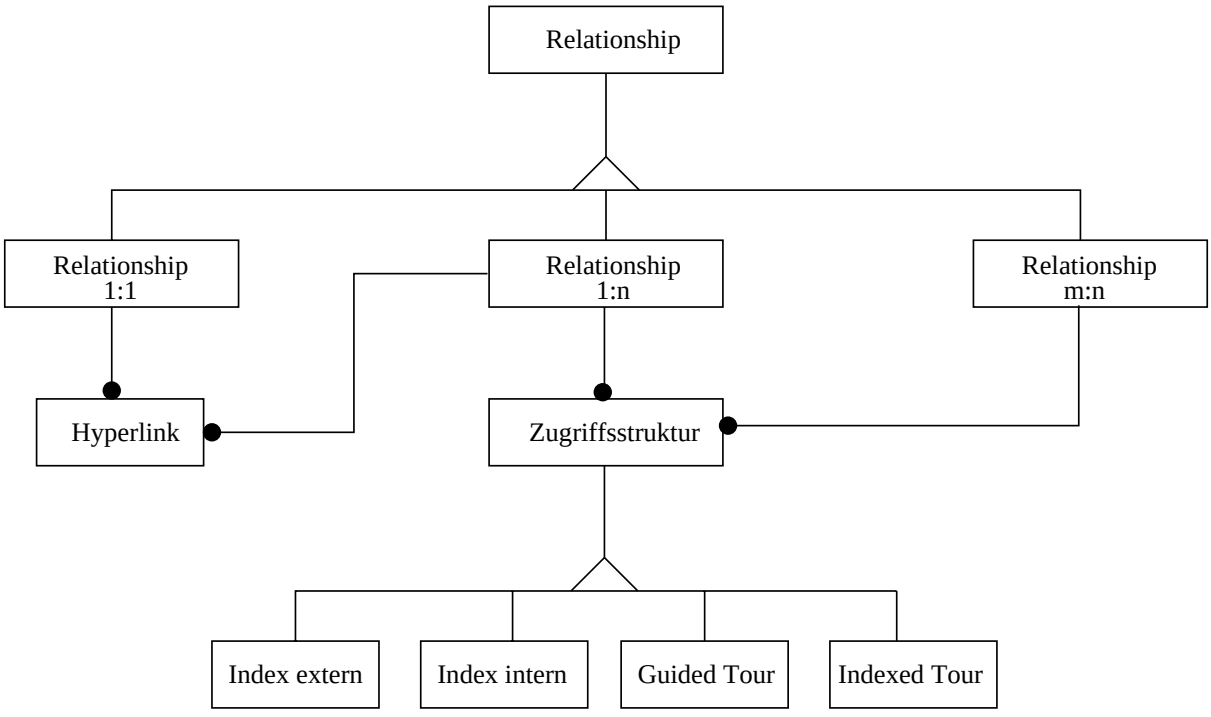


Abbildung 5.14: Navigations-Meta-Modell

Ein Hyperlink ist eine Referenz von einer HTML-Seite auf eine andere HTML-Seite und wird durch eine 1:1-Beziehung oder 1:n-Beziehung zwischen zwei Entity-Typen definiert. Eine Zugriffsstruktur modelliert die Zugriffsart auf eine Menge von Tupeln eines Entity-Typs. Hier lassen sich Guided Tour, Indexed Guided Tour, Index Extern und Index Intern unterscheiden. m:n-Beziehungen zwischen Entity-Typen können entweder auf zwei

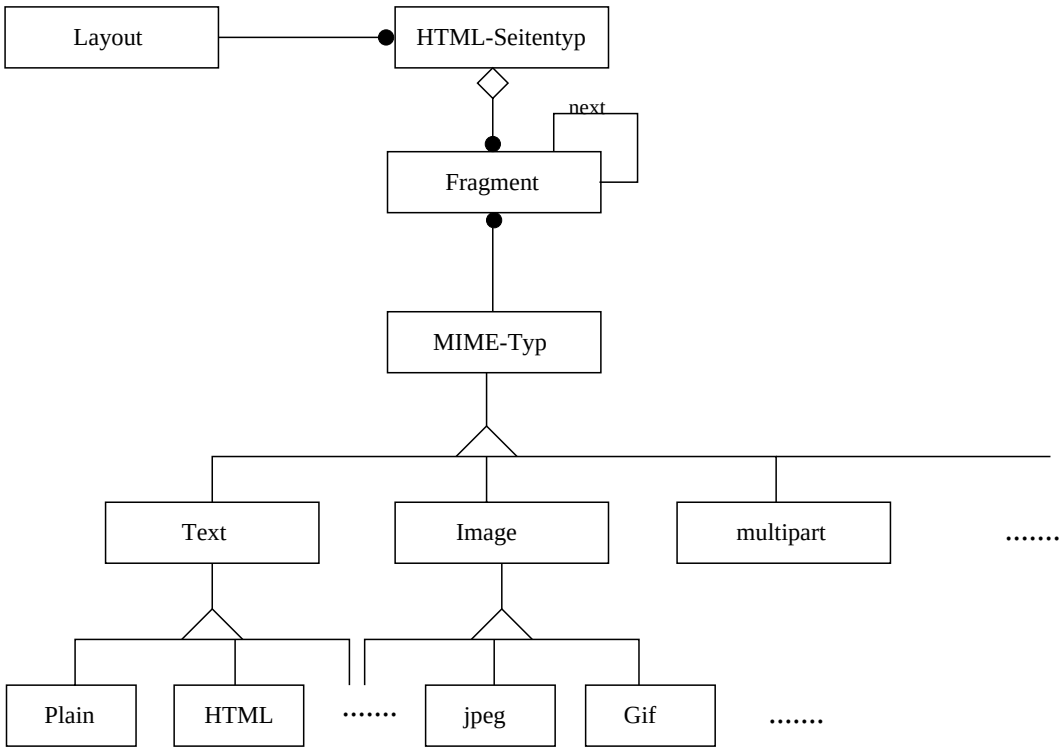


Abbildung 5.15: Präsentations-Meta-Modell

1:n-Beziehungen abgebildet werden oder durch einen Reference-Index modelliert werden, der den Zugriff auf beide beteiligten Entity-Typen von einer Index-Seite ermöglicht.

5.5.4 Präsentations-Meta-Modell

M-Slices beschreiben ausschließlich welche Informationen auf einer HTML-Seite gezeigt werden sollen aber nicht wie diese Informationen dargestellt werden. Aus diesem Grund wird in Abbildung 5.15 das Layout auf HTML-Seitenebene betrachtet. Jeder HTML-Seitentyp (Presentation Unit) zeichnet sich durch ein individuelles Layout aus, das unter anderem durch den Hintergrund der Seite und den Header und Footer der Seite bestimmt wird. Jedem HTML-Seitentyp wird genau ein Layout zugewiesen.

Gleichzeitig ist eine HTML-Seite aus verschiedenen Fragmenten aufgebaut, die jeweils MIME-Typen haben. MIME steht für Multipurpose Internet Mail Extensions und ist ein System von Internetstandards zur Kennzeichnung und Übertragung von Nicht-ASCII-Zeichen und multimedialen Binärdateien. MIME bettet sich in die alten Standards für Mail (RFC 822) und News (RFC 1036) ein und findet auch im HTTP Verwendung. Jedem Fragment ist genau ein MIME-Typ zugewiesen.

Für die Modellierung des Layouts in einem Meta-Modell gibt es verschiedene Varianten. Eine Möglichkeit ist die Ablage der wichtigsten Layout-Informationen zu Slice-Typen innerhalb von Datenbank-Relationen. Diese Variante wurde in der HDBM verwendet. Eine andere Variante ist die Verwendung von Cascading Style Sheets (siehe Kapitel 5.8).

5.5.5 Integration der Modelle

Zur Ablage von Hypertext-Informationen in der Datenbank wurde, basierend auf den vier vorgestellten Segmenten, ein Meta-Modell entworfen. Abbildung 5.16 veranschaulicht das, mit Hilfe der OMT [RBP+91] modellierte konzeptuelle Modell. Eine genaue Erläuterung der einzelnen Attribute erfolgt in Kapitel 5.6.

Ein Slice setzt sich aus einer Liste von verschiedenen *Elementen* zusammen. Jedes Element gehört zu genau einem Slice und hat eine Position und eine Bezeichnung innerhalb des Slices. Ein Element kann eine *Zugriffsstruktur*, ein *Attribut* oder ein *Hyperlink* sein. Die Klasse Element hat deshalb drei Subklassen.

Im einfachsten Fall ist ein Element ein Attribut des Owner-Entities. Dabei wird jedes Attribut eines Slices nur genau einem Attribut des Owner-Entities zugeordnet. Jedes Attribut eines Slices wird durch einen *Web-Typ* angezeigt, der zur richtigen Präsentation des Attributs auf der HTML-Seite dient. HDBM unterstützt eine fest vorgegebene Anzahl von Standard-Web-Typen, die beliebig ergänzt werden können. Dabei besteht eine Abhängigkeit zwischen Datenbank- und Web-Typen (siehe Tabelle 5.1). Ein in der Datenbank als BLOB gespeichertes Word-Dokument darf beispielsweise nicht mit dem Web-Typ EMAIL angezeigt werden. Da es sich beim World Wide Web um ein grafisch orientiertes Hypertext-Informationssystem handelt, beziehen sich die Web-Typen vorwiegend auf die Präsentation einzelner Datenbank-Attribute.

Meta-Typ	*HTML-Präsentation*	*Verfügbar für Datentyp*
TEXT	Textanzeige	CHAR(*)
EMAIL	Hyperlink vom Typ email	CHAR(*)
URL	Link vom Typ URL	CHAR(*)
BOOL	Ja/Nein	BOOL
DOWNLOAD	File Download	BLOB
IMAGE	Bildanzeige	BLOB

Tabelle 5.1: Erweiterbare Web-Typen der HDBM

Ein Element vom Typ *Hyperlink* ist definiert durch einen Anker (Ankerslice) und ein Ziel (Zielslice), die jeweils wieder M-Slices sind. Die Attribute des Ankerslices werden mit einem Hyperlink unterlegt. Ankerslice und Zielslice gehören zur gleichen Owner-Relation. Ein Beispiel hierfür wäre die Owner-Relation Standort der beiden Slices `Standort.Name` (Ankerslice) und `Standort.Info` (Zielslice) in Abbildung 5.5.

Durch *Zugriffsstrukturen* werden Tupel aus verschiedenen Datenbank-Tabellen miteinander über einen Join in Beziehung gesetzt. Der Anzeigeslice bestimmt Präsentation und Struktur eines Ergebnistupels der zugehörigen Owner-Relation. Die Zugriffsstruktur definiert die Art und Weise, wie auf eine Menge von Ergebnistupeln zugegriffen werden soll. Dabei werden die in Tabelle 5.2 dargestellten Zugriffsstrukturen unterschieden.

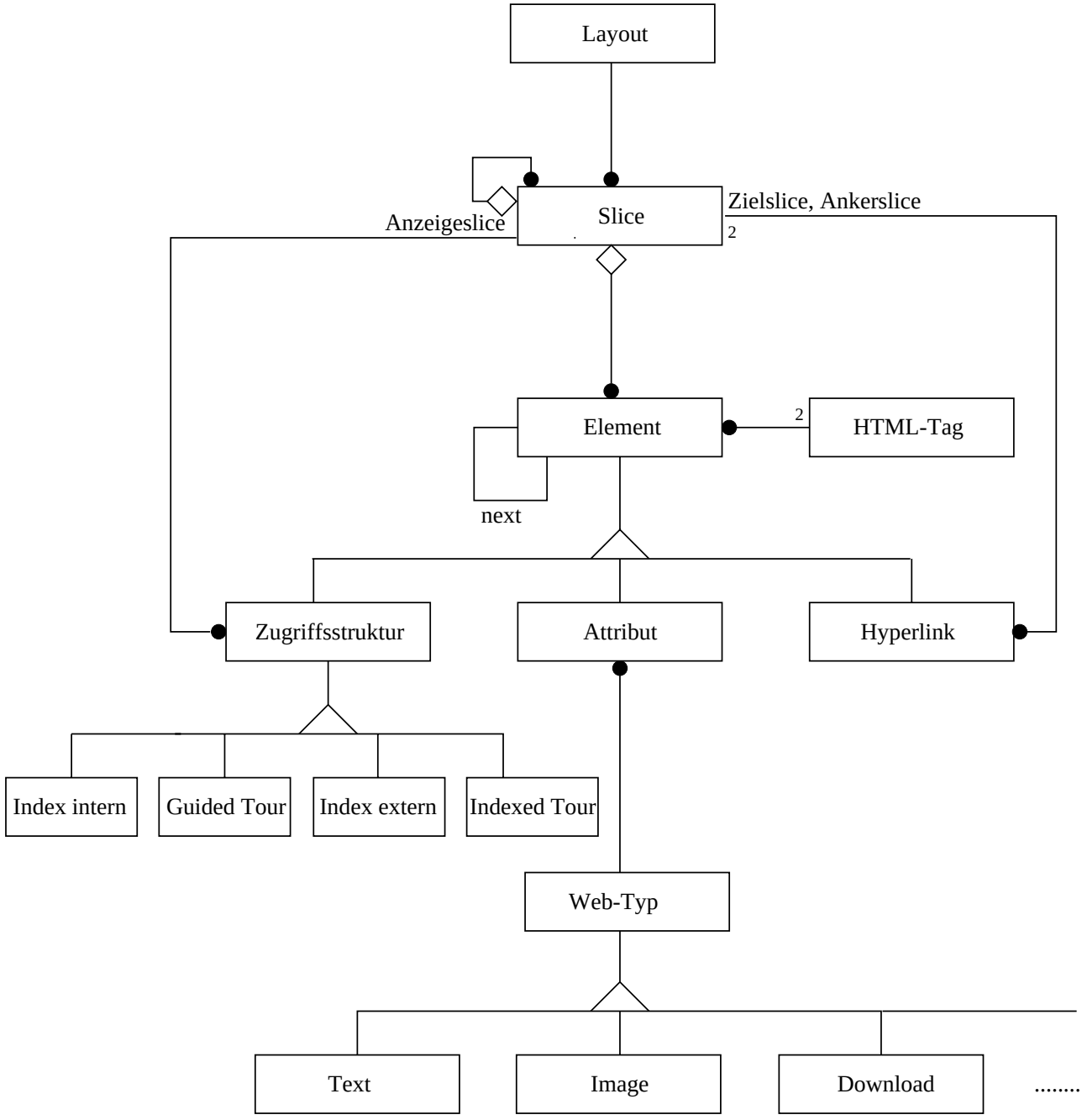


Abbildung 5.16: HDBM-Meta-Modell

Zugriffsstruktur	*HTML-Präsentation*
INDEX_INTERN	Die Tupel des referenzierten M-Slices werden direkt auf der HTML-Seite angezeigt
INDEX_EXTERN	Die Tupel des referenzierten M-Slices werden auf einer separaten HTML-Seite als Index angezeigt
GUIDED_TOUR	Die Tupel des referenzierten M-Slices werden als Guided Tour angezeigt
INDEXED_TOUR	Kombination aus INDEX_EXTERN und GUIDED_TOUR

Tabelle 5.2: Zugriffsstrukturen

5.6 Abbildung auf ein relationales Schema

Das folgende Kapitel beschäftigt sich mit Organisation und Speicherung der Meta-Daten in einer relationalen Datenbank und zeigt verschiedene Modellierungsvarianten auf.

5.6.1 Der Systemkatalog

Der Systemkatalog (Data Dictionary) ist der Teil einer Datenbank, in der sämtliche Informationen über den Aufbau und die Struktur der Datenbank abgelegt werden, die unabhängig von der konkreten Anwendung für alle Benutzer von Interesse oder Bedeutung sind. Er wird mit dem Entwurf einer Datenbank automatisch angelegt und spiegelt den aktuellen Status der Datenbank wieder. Die interne Verwaltung des Systemkatalogs wird durch Aktionen des Anwenders gesteuert. So hat beispielsweise das Anlegen einer neuen Tabelle den sofortigen Eintrag im Data Dictionary zur Folge. Die in dem Systemkatalog abgelegten Daten umfassen üblicherweise Informationen über Tabellen, Attribute, Fremdschlüssel, Indizes, Primärschlüssel und andere Typen von Constraints, und werden als *Meta-Daten* bezeichnet.

In relationalen Datenbanksystemen wird der Katalog selbst in Form von Relationen angelegt, die von Datenbankroutinen und autorisierten Anwendern genutzt werden können. Ein Beispiel einer stark vereinfachten Katalogstruktur für die Informationen über Basisrelationen ist in Tabelle 5.3 dargestellt [EN94].

Attribute	*Beschreibung*
TABNAME	Eindeutiger Name der Relation
COLNAME	Attributname
COLTYPE	Datentyp
MEMBER_OF_PK	Teil eines Primärschlüssels (Ja/Nein)
MEMBER_OF_FK	Teil eines Fremdschlüssels (Ja/Nein)
FK_RELATION	Fremdschlüsselrelation

Tabelle 5.3: Relation REL_AND_ATTR_CATALOG

Bei Oracle besteht der Systemkatalog aus über 100 Tabellen, die im *system*-Tablespace liegen und zum Schema *sys* gehören. Die Tabellen sind normalisiert und codiert. Aus diesem Grund ist das interne Data Dictionary für den Datenbankadministrator nur schwer zu interpretieren. Um dem Datenbankadministrator benötigte Informationen über die Datenbank in leicht interpretierbarer Form zur Verfügung zu stellen, wird eine Schicht von Views als externes Data Dictionary über das interne Data Dictionary gelegt.

Auch wenn sich Datenbank-Modelle und Hypermedia-Modelle in vielen Punkten unterscheiden sind die Einträge im Systemkatalog wichtig für die Generierung von HTML-Seiten aus einer Datenbank. Mit Hilfe des Systemkatalogs ist bereits eine Standard-Abbildung der Datenbank-Relationen auf HTML-Seitentypen möglich. Hierfür sind Informationen über Primärschlüssel, Fremdschlüssel und Namen und Typen der Attribute einer Relation relevant. Dabei gibt es unterschiedliche Varianten, wie die Tupel einer Relation auf HTML-Seiten abgebildet werden. Folgende drei verschiedenen Ansätze sollen dies verdeutlichen [FGP96]:

- **Direkte Abbildung der Tupel auf HTML-Seiten**: In diesem Fall wird jedes Tupel t einer Relation auf eine HTML-Seite $node(t)$ abgebildet. Die Primärschlüsselattribute ermöglichen die eindeutige Identifizierung des Knotens durch den URL. Jeder Knoten beinhaltet entweder alle oder eine Untermenge der Attribute des Tupels. Hyperlinks zwischen Knoten werden mit Hilfe definierter Fremdschlüssel generiert.

- **Abbildung einer Menge von gleich strukturierten Tupeln auf eine HTML-Seite:** In diesem Fall werden alle Tupel einer Datenbank-Tabelle, die einer bestimmten Bedingung genügen auf einer HTML-Seite präsentiert. Ein Beispiel hierfür wäre eine Veröffentlichungsliste.

- **Abbildung einer Menge von in Beziehung stehender Tupel auf eine HTML-Seite:** In diesem Fall werden die Tupel von verschiedenen, miteinander in Beziehung stehenden Relationen auf einer HTML-Seite angezeigt. Hierfür werden Fremdschlüssel genutzt.

Die im Meta-Schema definierten Tabellen und die darauf aufbauenden Werkzeuge nutzen zur Abbildung der Basis-Relationen auf ein Hypertext-Modell den Systemkatalog, erweitern diesen aber noch um weitere Informationen. Das relationale Meta-Schema wird im folgenden Kapitel genauer beschrieben.

5.6.2 Umsetzung in ein Relationenschema

Bei der Übertragung des im vergangenen Kapitel vorgestellten Meta-Modells auf ein relationales Schema erhält man acht Relationen, die im folgenden als Hypertext-Relationen bezeichnet werden. Zusammen mit den Relationen des Anwendungsbereichs, den sogenannten Basis-Relationen, bezeichnen wir das Datenbank-Schema als *Web-Datenbank-Schema.*

Definition 5.6.2.1
Ein Web-Datenbank-Schema ist eine finite Menge von Basis-Relationen $DB = R_1, \ldots R_n$ *mit* $n \in \mathbb{N}$, *und acht zusätzlichen Relationen: SLICE, ELEMENT, ATTRIBUT, ZUGRIFFSSTRUKTUR, HYPERLINK, ANKERSLICE, PRAESENTATION und WEBTYP. Die Relation SLICE beinhaltet logisch zusammengehörige Hypertext-Elemente, ATTRIBUT beinhaltet die Attribute der Hypertext-Knoten, HYPERLINK definiert die Beziehung zwischen Hypertext-Knoten, ZUGRIFFSSTRUKTUR definiert den Typ des Zugriffs auf eine Menge von DB-Tupeln. Eine Instanz des Web-Datenbank-Schemas W, auch Web-Datenbank genannt, ist eine Abbildung einer finiten Menge von Tupeln jeder Relation* $R_i \in DB$ $(i \in \{1, \ldots n\})$ *auf eine finite Menge von Knoten, eine finite Menge von Hypertext-Attributen und eine finite Menge von Links zwischen den Knoten.*

Modellierung der Slices

Jeder Eintrag in der Relation SLICE entspricht einem Slice-Typ. Dabei ist ein Slice-Typ aus verschiedenen Elementen aufgebaut und genau einer DB-Tabelle zugeordnet. Der Name des Slice-Typs ist eindeutig und dient zur Generierung der HTML-Seiten. Jeder Slice-Typ ist eindeutig einer Owner-Relation zugeordnet, die durch den Tabellennamen referenziert wird. Durch die mögliche Schachtelung entspricht nicht jeder Slice auf HTML-Seitenebene einem HTML-Seitentyp, sondern kann selbst Komponente eines anderen Slices sein. In der erweiterten RMM werden Slices, die einem HTML-Seitentyp entsprechen als Presentation Unit bezeichnet. Hintergrund und Titel eines Slice werden nur dann zur Generierung einer HTML-Seite ausgewertet, wenn es sich um eine Presentation Unit handelt.

Relation SLICE

Attribute	*Beschreibung*	*Datentyp*
SNAME	Bezeichnung des Slice-Typs	VARCHAR2
P#	Referenz auf Präsentation	NUMBER
TABNAME	Referenz auf Owner-Relation	VARCHAR2
PRESUNIT	ja/nein	VARCHAR2
TITEL	Titel der HTML-Seite	VARCHAR2

Modellierung der Präsentation eines Slices

In der Relation PRAESENTATION werden die wichtigsten Layout-Tags einer HTML-Seite gespeichert. Dazu gehören *Header*, *Footer* und der *Background*. Header und Footer ermöglichen es zu Beginn und Ende jedes Slices HTML-Tags einzufügen. Dabei spielt es keine Rolle ob dieser Slice eine *Presentation Unit* ist oder nicht. Der Hintergrund wird dagegen nur dann berücksichtigt, falls der Slice eine *Presentation Unit* ist. Ein Tupel der Relation PRAESENTATION kann verschiedenen Slice-Typen zugeordnet werden. Eine andere Modellierungsvariante wäre die Ablage von Cascading Style Sheets (Kapitel 5.8). Da diese zum jetzigen Zeitpunkt von gängigen Browsern noch nicht vollständig unterstützt werden, wurden sie im aktuellen Meta-Schema nicht berücksichtigt.

Relation PRAESENTATION

Attribute	*Beschreibung*	*Datentyp*
P#	Primärschlüssel	NUMBER
HEADER	HTML-Teil zu Beginn des Slices	VARCHAR2
FOOTER	HTML-Teil als Abschluß des Slices	VARCHAR2
BACKGROUND	Hintergrund der HTML-Seite	VARCHAR2

Modellierung der Elemente eines Slices

Jeder Slice baut sich aus verschiedenen Elementen auf, die entweder ein Attribut, ein Hyperlink, eine Zugriffsstruktur oder wiederum ein anderer Slice sein können. Geht man von einer sequentiellen Abfolge aus, so hat jedes Element genau eine *Position*. Das Einfügen von HTML-Tags vor und nach einem Element durch die Attribute HTMLSTART und HTMLENDE ermöglicht im Rahmen von HTML die beliebige Darstellung von Elementen. Beispielsweise kann durch die Tags `<B>` und `</B>` ein Element **fett** dargestellt werden. Eine genaue Positionierung von Elementen, beispielsweise durch die Angabe von Koordinaten für eine zweidimensionale Ausrichtung auf einer HTML-Seite, ist nicht möglich. Um dennoch eine Ausrichtung von Attributen auf einer HTML-Seite zu gewährleisten, teilen manche HTML-Editoren die HTML-Seite in feingranulare Tabellenelemente ein.

Relation ELEMENT

Attribute	*Beschreibung*	*Datentyp*
ELEM#	Primärschlüssel	NUMBER
SNAME	Referenz auf Slice-Typ	VARCHAR2
BEZEICHNUNG	Bezeichnung des Elements auf der HTML-Seite	VARCHAR2
POSITION	Reihenfolge der Elemente auf der HTML-Seite	NUMBER
HTMLSTART	HTML-Präsentation des Elements	VARCHAR2
HTMLENDE	HTML-Präsentation des Elements	VARCHAR2
SUBTABLE	Typ des Elements: Zugriffsstruktur, Attribut, Hyperlink	VARCHAR2

Die *Bezeichnung* des Elements dient zur Anzeige eines Attributs auf der HTML-Seite. Ein Beispiel hierfür wäre ' Name: Steiner', wobei 'Name' die Bezeichnung und 'Steiner' der Wert des Attributs ist. Damit ist gewährleistet, daß die Bezeichnung eines Datenbank-Attributs nicht gleich der Bezeichnung eines Web-Attributs sein muß. Die Bezeichnung eines Elements kann auch leer sein. Ein Beispiel hierfür wäre die Anzeige eines Bildes auf einer HTML-Seite. Der jeweilige Typ des Elements (Attribut, Hyperlink, Zugriffsstruktur) wird in dem Attribut SUBTABLE abgelegt.

Modellierung der Attribute von Relationen

Ein Web-Attribut ist eindeutig einem DB-Attribut der Owner-Relation des Slices zugeordnet (COLNAME). Jedem Attribut wird ein *Web-Typ* für die Darstellung im WWW zugewiesen. Die Bezeichnung des Web-Attributs auf der HTML-Seite wird der Relation ELEMENT entnommen. Sollen mehrere Attribute in Folge angezeigt werden (z.B: FORWISS München, Orleansstr. 34, 81667 München) wird kein Zeilenumbruch zwischen den Attributen (z.B. Bezeichnung, Strasse, Postleitzahl und Ort) definiert und als Trennzeichen ein Komma festgelegt.

Relation ATTRIBUT

Attribute	*Beschreibung*	*Datentyp*
ELEM#	Primärschlüssel des Elements	NUMBER
TYPNAME	Referenz auf den Web-Typ (z.B. email)	VARCHAR2
COLNAME	Referenz auf DB-Attribut der Owner-Relation	VARCHAR2
UMBRUCH	Zeilenumbruch vor Anzeige des folgenden Attributs (Ja/Nein)	VARCHAR2
TRENNZEICHEN	Trennzeichen vor Anzeige des folgenden Attributs	VARCHAR2

Modellierung der Web-Typen

Im WWW gibt es viele Datentypen, die von traditionellen Datenbanksystemen nicht unterstützt werden. Jedem Attribut eines Slices wird deshalb ein Web-Typ zugewiesen. Beispielsweise werden in der Datenbank die Web-Typen URL und Email als CHAR(*) definiert, im Web-Browser sollen diese aber mit entsprechenden HTML-Tags als ein Hyperlink und als Email dargestellt werden. Ähnliches gilt für in der Datenbank abgelegte BLOBS (z.B. Bilder oder Word-Dokumente). Das Attribut TYPNAME beinhaltet den Namen des Web-Typs, wie beispielsweise Email oder URL. Das Attribut HTMLTAG beschreibt die Behandlung des Web-Typs. Dabei wird als Platzhalter für das anzuzeigende Web-Attribut `hdbm_attr` verwendet. Für den Web-Typ URL würde das zugehörige HTMLTAG beispielsweise wie folgt aussehen: `<A HREF="hdbm_attr">hdbm_attr</A>`.

Relation WEB_TYP

Attribute	*Beschreibung*	Datentyp
TYPNAME	Primärschlüssel	VARCHAR2
HTMLTAG	HTML-Text (mailto ....)	VARCHAR2

Modellierung der Hyperlinks

Ein Hyperlink ist eine Referenz auf einen anderen Slice. Jeder Hyperlink wird durch einen *Ankerslice* und einen *Zielslice* definiert. Ankerslice und Zielslice müssen der gleichen Owner-Relation zugeordnet sein. Ein Beispiel hierfür ist der Ankerslice `Standort.Name` und der Zielslice `Standort.Info` in Abbildung 5.5. Der Anker definiert die Attribute, die auf einer HTML-Seite als Hyperlink angezeigt werden sollen und ist durch eine Referenz auf die Relation ANKERSLICE modelliert. Besteht ein Hyperlink beispielsweise aus den drei DB-Attributen TITEL, VNAME und NNAME einer Datenbanktabelle MITARBEITER, so würde eine Referenz auf eine Mitarbeiter-HTML-Seite wie folgt aussehen: Dr. Ute Berger. Dieser Hyperlink-Anker könnte beispielweise in der Projekt-Seite eines Forschungsverbundes eingebettet sein.

Ein Hyperlink-Anker, der in einem Slice-Typ S_1 (z.B. Projekt-Seite) eingebettet ist und auf einen anderen Slice-Typ S_2 (z.B. Mitarbeiter-Seite) verweist, wird auf relationaler Ebene durch einen Join zwischen den beiden beteiligten Owner-Relationen R_1 und R_2 berechnet, wobei A ein Attribut von R_1 und B ein Attribut von R_2 ist:

$$S_1 \longrightarrow S_2 = R_1 \bowtie_{R_1.A=R_2.B} R_2$$

Dabei werden aus R_2 nur die Attribute selektiert, die in dem Anker des Hyperlinks angezeigt werden sollen (z.B. Titel, Vorname, Nachname der Mitarbeiter-Relation).

Für die Abbildung eines Joins auf ein Meta-Schema gibt es verschiedene Möglichkeiten. Eine flexible Variante ist die Speicherung der zugehörigen SQL-Anfrage ohne Select-Klausel als Text-Attribut in der Relation HYPERLINK. Alternativ zu dieser Modellierung können Informationen zu Quell- und Ziel-Attribut des Joins separat abgelegt werden. Um auch Joins zwischen drei Relationen (bei n:m-Beziehungen) zu ermöglichen, wird ein weiteres Feld für eine Join-Relation vorgesehen. Quell- und Zielrelation sind durch die Owner-Relationen der beteiligten Slice-Typen gegeben. Quell- und Zielattribut der Join-Relation sind durch im DB-Schema definierte Fremdschlüssel gegeben. Im letzten Fall ist es nicht möglich einen Join über mehr als drei Relationen durchzuführen.

In HDBM wurden beide Möglichkeiten modelliert. In QUELLCOL, ZIELCOL, ZIEL, SORTATTR und JOINTAB definiert der Anwender alle zur Ausführung des Joins notwendigen Attribute. Möchte der Anwender eine komplexere SQL-Anfrage über verschiedene Relationen absetzen, kann diese im Feld QUERY spezifiziert werden. Die Attribute QUELLCOL, ZIELCOL, JOINTAB und SORTATTR bleiben in diesem Fall leer. In QUERY wird nur die From- und Where-Klausel abgelegt. Die Select-Klausel wird automatisch aus den Attributen des Ankers generiert.

Weiterhin wurden interne HYPERLINK-Listen durch die Relation HYPERLINK und

nicht durch die Relation ZUGRIFFSSTRUKTUR modelliert. Dies erhöht die Performance bei der Generierung einfacher Hyperlink-Listen.

Relation HYPERLINK

Attribute	*Beschreibung*	*Datentyp*
ELEM#	Primärschlüssel des Elements	NUMBER
ANKER	Bezeichnung des ANKERSLICE	VARCHAR2
ZIEL	Referenz auf die Relation SLICE	VARCHAR2
QUELLCOL	Informationen für den Join	VARCHAR2
ZIELCOL	Informationen für den Join	VARCHAR2
JOINTAB	Informationen für den Join	VARCHAR2
QUERY	Informationen für den Join	VARCHAR2
SORTATTR	Informationen für den Join	VARCHAR2

Modellierung der Hyperlink-Anker

Ein Ankerslice definiert die Attribute der Owner-Relation des Ziel-Slices, die als Hyperlink angezeigt werden sollen. In der RMM ist jeder Anker als ein M-Slice definiert. Da ein Hyperlink aber ausschließlich aus Attributen aufgebaut sein darf und demnach keine anderen Elemente, wie beispielsweise Zugriffsstrukturen, beinhaltet, wurden Anker als separate Relation modelliert. Ein Ankerslice entspricht demnach einem eingeschränkten Slice-Typ, der rekursiv keine anderen Slice-Typen beinhalten darf. Die Reihenfolge der Attribute wird durch das POSITION-Feld definiert. Die Owner-Relation des Anker-Slices ergibt sich aus der Owner-Relation des Ziel-Slices (siehe Relation HYPERLINK). Damit ist gewährleistet, daß Anker und Ziel die gleiche Owner-Relation haben. Beispielsweise muß ein Hyperlink Dr. Ute Berger auf einen Mitarbeiter verweisen.

Relation ANKERSLICE

Attribute	*Beschreibung*	*Datentyp*
NAME	Bezeichnung des Ankerslices	VARCHAR2
COLNAME	Name des Attributs der Ziel-Relation	VARCHAR2
POSITION	Reihenfolge der Attribute	NUMBER
TRENNZEICHEN	Trennzeichen zwischen den Attributen	VARCHAR2

Anhand der Abbildung 5.5 soll das Hyperlink-Konzept nochmal verdeutlicht werden. Auf der Mitarbeiter-Seite `Mitarbeiter.Info` ist ein Hyperlink definiert, der als Anker den Slice `Standort.Name` und als Ziel den Slice `Standort.Info` hat. Sowohl `Standort.Name` als auch `Standort.Info` haben die gleiche Owner-Relation STANDORT. Der Anker-Slice `Standort.Name` setzt sich aus den drei Attributen Standortname, Postleitzahl und Ort zusammen (siehe Abbildung 5.18). In der Relation Ankerslice werden deshalb drei Tupel eingetragen, die das jeweilige Attribut der Owner-Relation Standort referenzieren. Die Owner-Relation des Ankerslices wird nicht separat abgespeichert, sondern ist implizit die gleiche Owner-Relation des Ziel-Slices (STANDORT) der Relation HYPERLINK.

Modellierung der Zugriffsstrukturen

Die Zugriffsstruktur kennzeichnet die Art und Weise wie auf die Instanzen eines M-Slice zugegriffen wird. Zugriffsstrukturen werden dann definiert, wenn die Owner-Entities von Ziel- und Quell-Slice miteinander in einer n:m- oder 1:n-Beziehung stehen. Dabei unterscheidet man verschiedene Typen von Zugriffsstrukturen: GUIDED_TOUR, INDEX_INTERN, INDEX_EXTERN und INDEXED_TOUR. Bei einem Index wird eine Tupelliste mit den Instanzen der Owner-Relation des Anzeigeslices erzeugt. Die Tupel werden entweder auf der gleichen HTML-Seite oder auf einer separaten HTML-Seite angezeigt. Beispiel hierfür ist eine Liste von Veröffentlichungen. Eine Guided Tour leitet den Anwender mit Hilfe einer verketteten Liste von HTML-Seiten durch die verschiedenen Instanzen der Owner-Relation. Eine Indexed Guided Tour ist die Verbindung der beiden Zugriffsstrukturen INDEX_EXTERN und GUIDED_TOUR. Als Index auf die Guided Tour dient das erste Text-Attribut des HTML-Seitentyps des geführten Rundgangs. Das Attribut TEXT modelliert freien Text, der bei externen Zugriffsstrukturen als Link-Anker auf den Index verwendet wird (z.B. Mitarbeiterliste)

Um geschachtelte Slices zu modellieren, wurde INTERN als weitere Zugriffsstruktur definiert. INTERN und INTERN_INDEX unterscheiden sich im wesentlichen dadurch, daß bei INTERN_INDEX vor dem anzuzeigenden Element noch ein Listensymbol (`<LI>`) erzeugt wird. Bei dem Zugriffstyp INTERN ist dies nicht nötig, da es sich sicher um genau ein Tupel handelt.

Relation ZUGRIFFSSTRUKTUR

Attribute	*Beschreibung*	*Datentyp*
ELEM#	Primärschlüssel des Elements	NUMBER
ANZEIGESLICE	Referenz auf einen SLICE	VARCHAR2
ZTYP	INTERN, INDEX_EXTERN, GUIDED_TOUR, INDEX_INTERN, INDEXED_TOUR	VARCHAR2
QUELLCOL	Informationen für den Join	VARCHAR2
ZIELCOL	Informationen für den Join	VARCHAR2
JOINTAB	Informationen für den Join	VARCHAR2
QUERY	Informationen für den Join	VARCHAR2
SORTATTR	Attribut zur Sortierung der Treffer	VARCHAR2
TEXT	Freier Text, der als Link auf einen externen Index dient	VARCHAR2

Durch Abbildung 5.17 werden die Fremdschlüsselbeziehungen zwischen den HDBM-Meta-Relationen nochmal verdeutlicht.

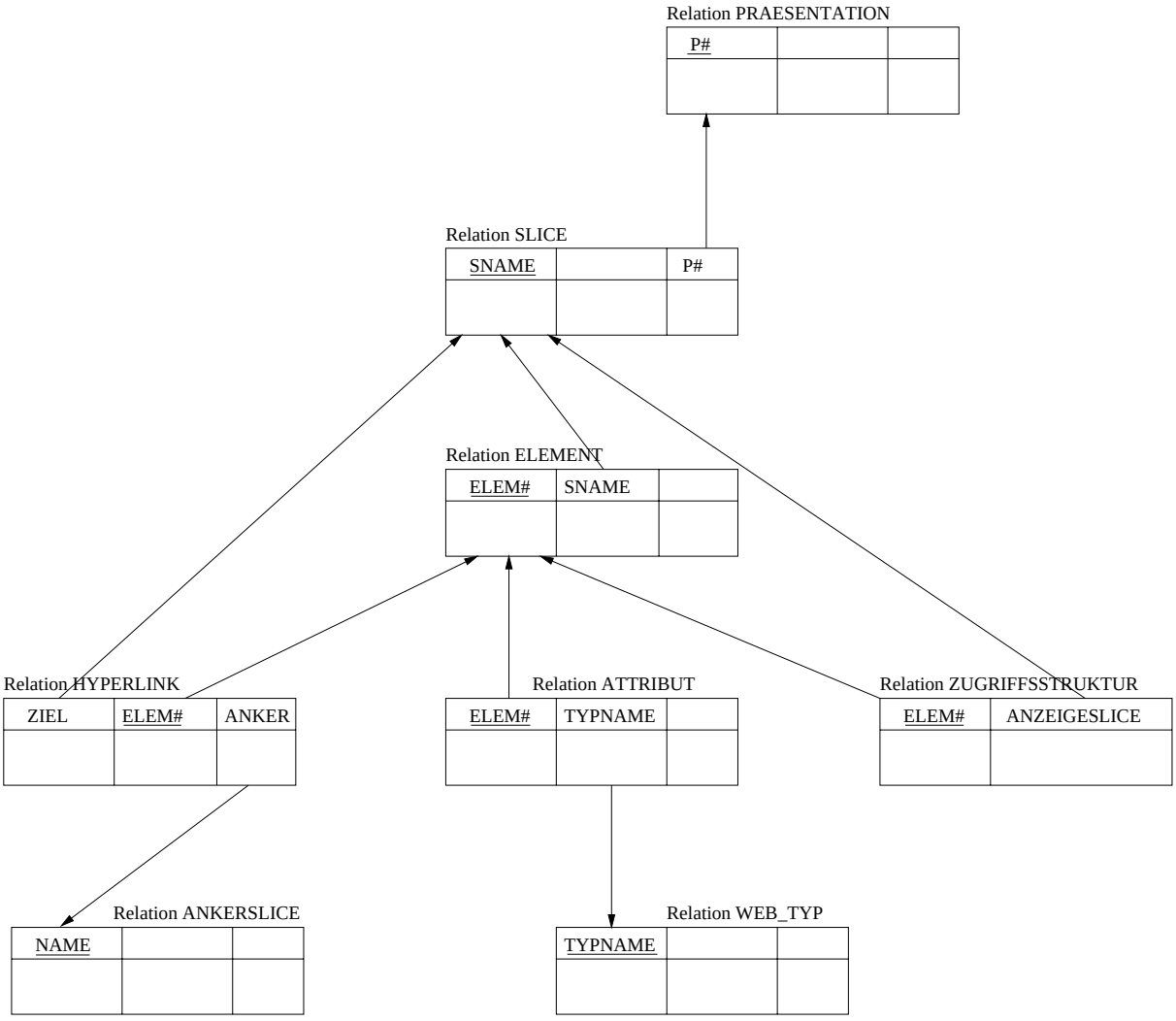


Abbildung 5.17: Fremdschlüssel-Beziehungen zwischen den Meta-Relationen

5.6.3 Getroffene Modellierungsentscheidungen

Bei der Modellierung des Meta-Schemas wurde versucht, auf der einen Seite vielfältige Gestaltungsmöglichkeiten für HTML-Seitentypen zu ermöglichen und auf der anderen Seite das Schema hinsichtlich häufig verwendeter Anfragen, wie beispielsweise der Selektion von Ankern und Attributen, zu optimieren. Im folgenden sollen die wichtigsten Modellierungsentscheidungen bei der Erstellung des relationalen Meta-Schemas erläutert und begründet werden.

Die zentrale Relation des Meta-Schemas bildet die Relation SLICE. Durch sie werden Eigenschaften modelliert, die für alle Elemente eines Slice-Typs gelten. Ein Beispiel hierfür ist die Zuordnung eines Slice-Typs zu einer Owner-Relation. Slice-Typen werden abhängig davon, ob sie einem HTML-Seitentyp entsprechen oder nur eine Komponente eines anderen Slice-Typs bilden, in Presentation Units und Component Slices unterschieden. Dieser Sachverhalt wurde durch das Attribut PRESUNIT der Relation SLICE modelliert. Die Modellierung einer Vererbungshierarchie auf relationaler Ebene und die hierdurch bedingte schlechtere Performance durch die Auswertung eines weiteren Joins bei der Generierung der HTML-Seiten, erschien nicht sinnvoll. Sowohl Presentation Units als auch Component Slices bilden ähnliche Eigenschaften und Beziehungen (Abbildung 7.5) aus.

Durch die einheitliche Modellierung aller Elemente eines Slice-Typs in der Relation ELEMENT ist die durchgängige Intra-Dokument-Struktur, unabhängig von dem jeweiligen Typ des Elements (M-Slice, Attribut, Hyperlink oder Zugriffsstruktur) gewährleistet. Dies betrifft insbesondere die Reihenfolge der Elemente, die Bezeichnung der Elemente und die Darstellung durch HTML-Tags. Für die Abbildung einer Vererbungshierarchie auf relationale Ebene existieren verschiedene Modellierungsvarianten [EN94]. In HDBM wurde der Primärschlüssel der Relation ELEMENT als Primärschlüssel der drei Relationen ATTRIBUT, HYPERLINK und ZUGRIFFSSTRUKTUR übernommen. Zur Optimierung der Suche nach dem jeweiligen Typ eines Elements wurde in der Relation ELEMENT ein Attribut SUBTABLE ergänzt, das auf die jeweilige Relation verweist.

Jedes Web-Attribut einer HTML-Seite entspricht einem DB-Attribut einer Basis-Relation. Die Web-Typen werden in einer separaten Relation zusammen mit den für die Behandlung des Web-Typs notwendigen HTML-Tags abgelegt. Die getrennte Verwaltung von Web-Attributen und zugehörigen Web-Typen gewährleistet die einfache Definition neuer Web-Typen und die leichte Skalierbarkeit des Meta-Schemas.

In der RMM entspricht der Anker eines Hyperlinks einem vollwertigen M-Slice. Ein Anker darf nur aus Attributen, nicht aber wiederum aus Hyperlinks oder Zugriffsstrukturen aufgebaut sein. Um eine performantere Generierung von Hyperlinks zu gewährleisten, wurde deshalb der Anker eines Hyperlinks durch eine separate Relation ANKERSLICE modelliert. Weiterhin wurden interne Hyperlink-Listen ebenfalls über die Relation HYPERLINK und nicht über die Relation ZUGRIFFSSTRUKTUR modelliert. Dies ermöglicht die effiziente Generierung einfacher Index-Listen.

Der Zugriff auf eine Menge von DB-Tupeln erfolgt über eine der vier Zugriffsstrukturen: Index Intern, Index Extern, Guided Tour und Indexed Guided Tour. Alle Zugriffs-

strukturen wurden durch eine Relation modelliert. Die Darstellung erfolgt abhängig von dem Attribut ZTYP. Für die Modellierung von geschachtelten Slices wurde ein spezieller Zugriffstyp INTERN eingeführt. Dies ermöglicht die einheitliche Modellierung von geschachtelten Slice-Typen und Zugriffsstrukturen in einer Relation.

5.6.4 Definition von Constraints

Eine der grundlegenden Anforderungen an eine Datenbank ist die Korrektheit der dort für die Anwendungswelt modellierten Daten. Zur Spezifikation der semantischen Korrektheit werden typischerweise Integritätsbedingungen verwendet, die zur Laufzeit durch das Datenbanksystem, beispielsweise mit Hilfe von *Triggern* oder *check-Klauseln*, geprüft werden und im Falle einer Verletzung zur Zurücksetzung der betroffenen Transaktion führen.

Die Abbildung von Slice-Typen auf Relationen, Attribute und Beziehungen unterliegt gewissen Regeln. Ein Slice-Typ und dessen Attribute sollen beispielsweise automatisch gelöscht werden, falls die zugehörige Owner-Relation gelöscht wird. Weiterhin sollten Anker-Slice und Ziel-Slice eines Hyperlinks die gleiche Owner-Relation haben. Die Konsistenz innerhalb des Meta-Schemas wird durch Primär- und Fremdschlüssel gewährleistet. Ein Slice-Typ darf beispielsweise nicht gelöscht werden, wenn Hyperlinks oder Zugriffsstrukturen auf ihn verweisen. Hyperlinks ins Leere (Dangling Links) sind so nicht möglich.

Zwischen dem Meta-Schema und dem Data Dictionary der Datenbank existieren ebenfalls funktionale Abhängigkeiten. Beispielsweise darf ein Slice-Typ nur dann angelegt werden, wenn die zugehörige Owner-Relation im Datenbankschema existiert. Das gleiche gilt für Web-Attribute und die zugehörigen DB-Attribute. Die Definition eines Fremdschlüssels auf eine Tabelle des Data Dictionary ist in vielen Datenbanksystemen nur beschränkt möglich. Konsistenzbedingungen werden in diesem Fall mit Hilfe von Triggern, die durch den INSERT- oder UPDATE-Befehl ausgelöst werden, überprüft.

Neben den Constraints, die auf Datenbankebene zur dynamischen Generierung von HTML-Seiten erfüllt sein müssen, werden im World Wide Web auch Constraints auf HTML-Seitenebene unterschieden. Bei der dynamischen Generierung von HTML-Seiten aus einer Datenbank werden diese Constraints automatisch durch die Gewährleistung der referentiellen Integrität des Datenbanksystems überwacht.

Hypertext-Applikationen im World Wide Web beinhalten üblicherweise einen hohen Grad an Redundanz. Zum einen sind viele Teile von Informationen über mehrere HTML-Seiten repliziert. Zum anderen können einzelne Seiten häufig durch unterschiedliche Pfade erreicht werden. Nach [MMM98] werden deshalb in Hypertext-Applikationen *Link Constraints* und *Inclusion Constraints* unterschieden.

Ein *Link Constraint* ist ein Prädikat, das mit einem Link assoziiert wird, falls ein Wert der HTML-Quellseite dem Wert einer HTML-Zielseite entspricht. Im Falle der Verbund-Homepage (siehe Abbildung 5.9) wäre dies beispielsweise das Attribut *Name* der Mitarbeiter-Seite und das Attribut Mitarbeiter-*Name* der Projekt-Seite:

Projekt⋄MitarbListe.Name = Mitarbeiter.Name

Definition 5.6.4.1
Gegeben seien zwei HTML-Seitentypen P_1 und P_2, die über einen Hyperlink $ToP2$ miteinander verbunden sind. Ein Link Constraint ist jeder Ausdruck der Form: $A = B$, wobei A ein einwertiges Attribut von P_1 und B ein einwertiges Attribut von P_2 ist. Gegeben seien zwei Instanzen der HTML-Seitentypen. Wir sagen ein Link hält genau dann, wenn für jedes Tupelpaar: $t_1 \in P_1$, $t_2 \in P_2$ gilt: das Attribut ToP_2 von t_1 ist gleich dem URL von t_2, dann und nur dann, wenn das Attribut A von t_1 gleich dem Attribut B von t_2 ist.

Neben Link Constraints werden im Web *Inclusion Constraints* unterschieden. Ein Inclusion Constraint gibt Aufschluß darüber, welcher Navigationspfad in welchem anderen Navigationspfad beinhaltet ist. Betrachtet man beispielsweise wieder das Application-Diagramm der Verbund-Homepage so sieht man, daß die *Mitarbeiter*-Seite entweder über die Seite *MitarbeiterListe* oder die Seite *Projekt* erreicht werden kann. Dabei gilt:

Projekt$\diamond$MitarbListe.ToMitarb $\subseteq$ MitarbeiterListe.ToMitarb

Definition 5.6.4.2
Gegeben seien ein Seiten-Schema P und zwei Link-Attribute L_1, L_2 zu P in P_1 und P_2. Ein Inclusion Constraint ist ein Ausdruck der Form $P_1.L_1 \subseteq P_2.L_2$. Gegeben seien zwei Instanzen p_1 und p_2 von den Seitentypen P_1 und P_2. Wir sagen der Constraint hält falls gilt: für jedes Tupel $t_1 \in p_1$, gibt es ein Tupel $t_2 \in p_2$, so daß der Wert von L_1 in t_1 gleich dem Wert von L_2 in t_2 ist.

Die Gewährleistung dieser Constraints auf HTML-Seitenebene wird bei der Wartung materialisierter HTML-Seiten relevant (siehe Kapitel 7.2).

5.7 Abbildung zwischen Hypertext- und Datenbankschema

Zur Umsetzung der konzeptuellen Modelle auf relationale Ebene müssen Umsetzungsregeln definiert werden. Dabei gelten für die Abbildung des ER-Modells auf ein relationales Modell Standardregeln, die eine systematische und weitgehend verlustfreie Abbildung erlauben. Bei der Abbildung des Hypertext-Modells werden die verschiedenen Slice-Typen auf Instanzen des Meta-Schemas abgebildet. Im folgenden sollen die Abbildung von Entities auf ein relationales Schema und die Abbildung von M-Slices auf Instanzen des Meta-Schemas beschrieben werden.

5.7.1 Abbildung des ER-Modells auf das relationale Modell

Bei der Abbildung des ER-Modells auf ein relationales Modell werden Standardregeln eingesetzt [Spe98].

Abbildung von Entity-Typen auf Relationen:
Jeder Entity-Typ E mit Attributen A_i aus Domäne $D_i (1 \leq i \leq k)$ wird abgebildet auf eine k-stellige Relation $E(A_1 : D_1, A_2 : D_2, \ldots A_k : D_k)$. Falls außerdem E *isa* F besteht oder E ein schwacher Entity-Typ, existenzabhängig von F ist, werden die Schlüsselattribute von F in E hinzugenommen.

1:1-Beziehung zwischen nur zwei Entity-Typen E und F:
Die Primärschlüssel der Relation E werden in die Relation F oder die Primärschlüssel der Relation F in die Relation E aufgenommen.

1:n-Beziehung zwischen nur zwei Entity-Typen E und F:
Die Primärschlüssel der Relation E werden in die Relation F als Fremdschlüssel aufgenommen.

n:m-Beziehung und alle 3er, 4er, etc. Beziehungen:
Eine Relationship R zwischen Entity-Typen $E_1, \ldots E_n$ wird abgebildet auf die Relation R, deren Attribute aus den Primärschlüsseln der E_i besteht.

Angewendet auf das Beispiel eines Forschungsverbundes, würde das relationale Schema wie folgt aussehen. Zur Unterscheidung zwischen Primär- und Fremdschlüssel werden Primärschlüssel **fett** und unterstrichen und Fremdschlüssel nur unterstrichen dargestellt:

1. VERBUND (**VERBNAME**, BEZEICHNUNG, LOGO, BESCHREIBUNG, GRDATUM, URL)
2. PROJEKT (**PTITEL**, BEZEICHNUNG, START, ENDE, BESCHREIBUNG, URL)
3. MITARBEITER (**M#**, VERBUND, BILD, VNAME, NNAME, TITEL, TELNR, EMAIL, SNAME, BEREICH)
4. STANDORT (**SNAME**, STRASSE, PLZ, ORT, PF, ANREISE)
5. PROJMITARB (M#, PTITEL)

5.7.2 Befüllung des Meta-Schemas

Um die Vielfalt der Informationen über logische und physische Strukturen einer Datenbank, die im erweiterten Data Dictionary eines Datenbanksystems gespeichert sind, leichter handhabbar zu machen, soll die Befüllung des Meta-Schemas durch eine graphische Benutzerschnittstelle erfolgen. Diese ermöglicht es dem Datenbankadministrator mit Hilfe von Verweisstrukturen entlang verschiedener Abhängigkeiten zwischen Datenbank-Objekten und Hypertext-Objekten durch das Data Dictionary zu browsen. Die Umsetzung und Beschreibung der graphischen Eingabekomponente zur Befüllung des Meta-Schemas ist Inhalt einer anderen am FORWISS durchgeführten Forschungsarbeit. Die

folgenden Abbildungsregeln sollen auf der Basis der im Kapitel 5.3.3 vorgestellten Programmspezifikation die Abbildung der Basis-Komponenten des erweiterten RMDM auf ein relationales Meta-Schema skizzieren.

Abbildung von Slice-Typen auf Instanzen der Relation SLICE:
Jeder Slice-Typ S mit Owner-Entity E wird abgebildet auf eine Instanz t der Relation SLICE wobei das Attribut SNAME von t dem Namen des Slices S und das Attribut TABNAME von t dem Namen der Relation E entspricht. Um die Eindeutigkeit des Namens eines Slice-Typs zu erzielen, wird dieser aus dem Namen der Owner-Relation und der Bezeichnung des Slice-Typs konstruiert. Slice-Typ und Owner-Entity werden der Programmspezifikation eines M-Slices entnommen: ⟨m-slice⟩ :== ⟨owner-entity⟩.⟨slice⟩

Abbildung der Elemente eines Slice-Typs auf Instanzen der Relation ELEMENT:
Jedes Element des Slice-Typs S wird abgebildet auf eine Instanz t der Relation ELEMENT wobei das Attribut ELEM# von t ein eindeutiger Primärschlüssel ist, das Attribut POSITION von t die Position des Elements innerhalb des Slices und das Attribut SNAME von t dem Namen des zugehörigen Slices S entspricht. Weiterhin kann durch den Anwender eine BEZEICHNUNG des Elements auf der HTML-Seite ergänzt werden. Derartige Layout-Informationen wurden im konzeptuellen Design nicht berücksichtigt. Jedes Element kann entweder ein Attribut, ein Hyperlink oder eine Zugriffsstruktur sein. Ist ein Element ein M-Slice, so wird dies durch die Zugriffsstruktur INTERN ausgedrückt.

Abbildung von Attributen eines Slices auf Instanzen der Relation ATTRIBUT:
Jedes Attribut A eines Slice-Typs S und eines Owner-Entities E wird abgebildet auf eine Instanz t der Relation ATTRIBUT wobei das Attribut ELEM# von t dem eindeutigen Schlüssel des zugehörigen Elements und das Attribut COLNAME eine Referenz auf das Attribut A der Owner-Relation E enthält. Der Name des Attributs wird der Programmspezifikation entnommen.

Abbildung von Hyperlinks eines Slices auf Instanzen der Relation HYPERLINK:
Jeder Hyperlink H mit Anker A, Ziel Z und Relationship R wird abgebildet auf eine Instanz t der Relation HYPERLINK wobei das Attribut ANKER von t eine Referenz auf den Ankerslice A, das Attribut ZIEL eine Referenz auf den M-Slice Z, und ELEM# eine Referenz auf das zugehörige Element beinhaltet. Die Bezeichnung der Relationship und deren Abbildung auf das relationale Modell gibt Aufschluß über Quell-Relation, Ziel-Relation und die an dem Join beteiligten Attribute. Ist der Inhalt einer Zugriffsstruktur ein Hyperlink, so wird dieser ebenfalls auf die Relation HYPERLINK abgebildet.

- ⟨hyperlink⟩ :== [relation] →*⟨anchor⟩ ⇒ ⟨destination⟩
- ⟨access_structure⟩ :== [⟨relation⟩] → index intern (⟨hyperlink⟩)

Abbildung eines Ankerslices auf Instanzen der Relation ANKERSLICE:
Jeder Ankerslice S eines Hyperlinks mit Attributen $A_1, \ldots A_n$ mit $n \in \mathbb{N}$ wird abgebildet auf n Instanzen $t_1, \ldots t_n$ der Relation ANKERSLICE wobei das Attribut NAME von t_i ($i \in 1, \ldots n$) dem Namen von S und das Attribut COLNAME eine Referenz auf das Attribut A_i der Owner-Relation E enthält.

Abbildung von Zugriffsstrukturen auf Instanzen der Relation ZUGRIFFSSTRUKTUR:
Jede Zugriffsstruktur Z mit Inhalt I und Relationship R wird abgebildet auf eine Instanz t der Relation ZUGRIFFSSTRUKTUR wobei das Attribut ZTYP von t dem Zugriffstyp von Z und das Attribut ANZEIGESLICE eine Referenz auf den M-Slice I ist. Die Bezeichnung der Relationship und deren Abbildung auf das relationale Modell geben Aufschluß über Quell-Relation, Ziel-Relation und die an dem Join beteiligten Attribute:

⟨access_structure⟩ :== [⟨relation⟩] → ⟨access_typ⟩ (⟨m-slice⟩)

Am Beispiel der Modellierung eines Forschungsverbundes sieht die Modellierung beispielsweise wie folgt aus. Auf die Definition der Präsentation und nicht belegter Attribute wird aus Platzgründen verzichtet. Weiterhin wurden nur die Elemente des `Mitarbeiter.Info` Slices genauer erläutert.

Relation SLICE

SNAME	*P#*	*TABNAME*	*PRESUNIT*	*TITEL*
MitarbInfo		MITARBEITER	JA	Mitarbeiter
StandortInfo		STANDORT	JA	Standort
VerbundInfo		VERBUND	JA	Verbund
ProjektTitel		PROJEKT	NEIN	
ProjektInfo		PROJEKT	JA	Forschungsprojekt

Relation ELEMENT

ELEM#	*SNAME*	*BEZEICHNUNG*	*POSITION*	*SUBTABLE*	...
1	MitarbInfo		1	Attribut	
2	MitarbInfo	Vorname	2	Attribut	
3	MitarbInfo	Nachname	3	Attribut	
4	MitarbInfo	Titel	4	Attribut	
5	MitarbInfo	Email	5	Attribut	
6	MitarbInfo	Telefon	6	Attribut	
7	MitarbInfo	Arbeitsgebiete	7	Attribut	
8	MitarbInfo	Standort	8	Hyperlink	
9	MitarbInfo	Verbund	9	Hyperlink	
10	MitarbInfo	Projekte	10	Zugriffsstruktur	
11	ProjektTitel		1	Hyperlink	
12	ProjektTitel		2	Attribut	

Relation ATTRIBUT

ELEM#	*TYPNAME*	*COLNAME*	...
1	IMAGE	BILD	
2	TEXT	VNAME	
3	TEXT	NNAME	
4	TEXT	TITEL	
5	EMAIL	EMAIL	
6	TEXT	TELNR	
7	TEXT	BEREICH	
12	TEXT	BEZEICHNUNG	

Relation ANKERSLICE

NAME	*COLNAME*	*POSITION*	...
StandortName	SNAME	1	
StandortName	PLZ	2	
StandortName	ORT	3	
VerbundName	VERBNAME	1	
ProjektName	PTITEL	1	

Relation HYPERLINK

ELEM#	*ANKER*	*ZIEL*	*QUELLCOL*	*ZIELCOL*	...
8	StandortName	StandortInfo	SNAME	SNAME	
9	VerbundName	VerbundInfo	VERBUND	VERBNAME	
11	ProjektName	ProjektInfo	M#	M#	

Relation ZUGRIFFSSTRUKTUR

ELEM#	*ANZEIGESLICE*	*ZTYP*	*QUELLCOL*	*ZIELCOL*	*JOINTAB*
10	ProjektTitel	INDEX_INTERN	M#	PTITEL	PROJMITARB

Die Instanz eines Mitarbeiter-Seitentyps auf HTML-Seitenebene wird durch Abbildung 5.18 veranschaulicht.

Vorname: Gabriele

Nachname: Höfling

Titel: Dr.

Email: hoefling@forwiss.tu-muenchen.de

Telefon: +49-89-48095-217

Arbeitsgebiete: Data Warehouses, Objektorientierte Datenbanksysteme, Dynamische Schemaevolution

Standort: FORWISS München, 81667 München

Verbund: FORWISS

Projekte:

- ITEX
  Information Technology for the Textile Industry
- DIADEM
  Diagnosedaten-Management
- MoodBase
  Objektorientiertes Datenbankmanagementsystem

Abbildung 5.18: HTML-Seite einer Mitarbeiterin

5.8 Layout und Präsentation

Die Definition des Layouts ist im Meta-Modell durch die Relation LAYOUT und die Attribute HTMLSTART und HTMLENDE der Relation ELEMENT möglich. Hierdurch können HTML-Titel, HTML-Footer und Hintergrund eines HTML-Seitentyps definiert und einzelne HTML-Tags für Elemente bestimmt werden. Die Verwendung von HTML-Tags zwischen den Elementen einer HTML-Seite ermöglicht die Verteilung und Positionierung von Objekten innerhalb einer HTML-Tabelle. In der Relation ELEMENT werden weiterhin die Bezeichnung und Reihenfolge der Attribute eines Seitentyps abgelegt.

Eine andere Variante zur Definition des Layouts von HTML-Seitentypen ist die Verwendung von *Style Sheets*. Ein Style Sheet ist eine Menge von Regeln, über die eine Abbildung von Markup-Elementen auf Layout-Elemente definiert wird. Style Sheets ermöglichen so die Trennung von Layout und Inhalt einer HTML-Seite und codieren die

Präsentation in wiederverwendbare, anpaßbare Komponenten. Zur Zeit existieren zwei weit verbreitete Standards: Zum einen die für HTML entworfenen *Cascading Style Sheets* (CSS) [BLLJ98] und zum anderen die *Document Style Semantics and Specification Language* (DSSSL), für SGML.

Folgendes Beispiel soll die Funktionsweise von *Cascading Style Sheets* verdeutlichen:

```
<STYLE TYPE="text/css">
<!--
BODY
{
  background-image: url(hintergrund.gif);
}

STRONG
{
  color: red;
  background-color: lightyellow;
  font-weight: bold;
  font-style: normal;
}

// -->
</STYLE>
```

In ein HTML-Dokument integriert, definiert dieses Style Sheet ein Hintergrundbild und ein neues Markup-Tag STRONG. Im einleitenden `<style>`-Tag ist der Typ der Formatdefinition angegeben. Damit Web-Browser, die keine Style-Sheets unterstützen, die Angaben nicht als anzuzeigenden Text interpretieren wurden Kommentare verwendet. Es werden interne von externen Style Sheets unterschieden. Interne Style Sheets werden in den Header des gewünschten HTML-Dokuments integriert. Sollen viele HTML-Dateien das gleiche Format haben, kann die Style Sheet-Angabe in eine separate Textdatei ausgelagert und über einen Link referenziert werden.

Angewendet auf das Meta-Modell wäre es möglich zu jedem HTML-Seitentyp ein externes Style-Sheet zu definieren, durch das die Präsentation der einzelnen HTML-Tags bestimmt wird. Leider ist der Einsatz von Cascading Style Sheets abhängig von dem verwendeten Web-Browser [MM99]. Erst Netscape 4.x interpretiert fast den vollen Sprachumfang von CSS-Version 1.0 und einen Teil der CSS-Version 2.0. Die Netscape Version 3.x unterstützt dagegen keine Style Steets. Der MS Internet Explorer unterstützt die CSS-Version 1.0 bereits seit der Produktversion 3.0.

Eine andere Alternative zu der reinen Layout-Beschreibungssprache HTML ist die Verwendung von XML zusammen mit der *XML Style Sheet Language* (XSL) [BLLJ98]. XML ermöglicht die semantische Strukturierung von Dokumenten mit Hilfe der im Datenbank-

Schema definierten Attribute und Relationen (siehe Kapitel 2.4.3). Mit XSL können für die einzelnen Datenbank-Attribute verschiedene Layouts definiert werden.

Leider werden sowohl CSS als auch XSL zum jetzigen Zeitpunkt von gängigen Browsern noch nicht vollständig unterstützt. Auch wenn dies für die nächsten Browser-Versionen zu erwarten ist, wurden Style Sheets bei der Modellierung des aktuellen Meta-Modells deshalb noch nicht berücksichtigt.

5.9 Zusammenfassung

HDBM basiert auf verschiedenen konzeptuellen und logischen Datenmodellen aus dem Datenbank- und Hypermedia-Bereich, die für den Entwurf datenbankbasierter Web-Sites kombiniert und erweitert wurden. Der Entwurfsprozeß von HDBM ist aus fünf Designphasen aufgebaut: Anforderungsanalyse, Konzeptuelles Design, Logisches DB-Design, Logisches Hypertext-Design und Physisches Design. Das konzeptuelle Design umfaßt ER-Design, Navigational-Design, Slice-Design und Application-Design und basiert auf den Modellierungskonzepten des ER-Modells und des erweiterten RMM-Datenmodells.

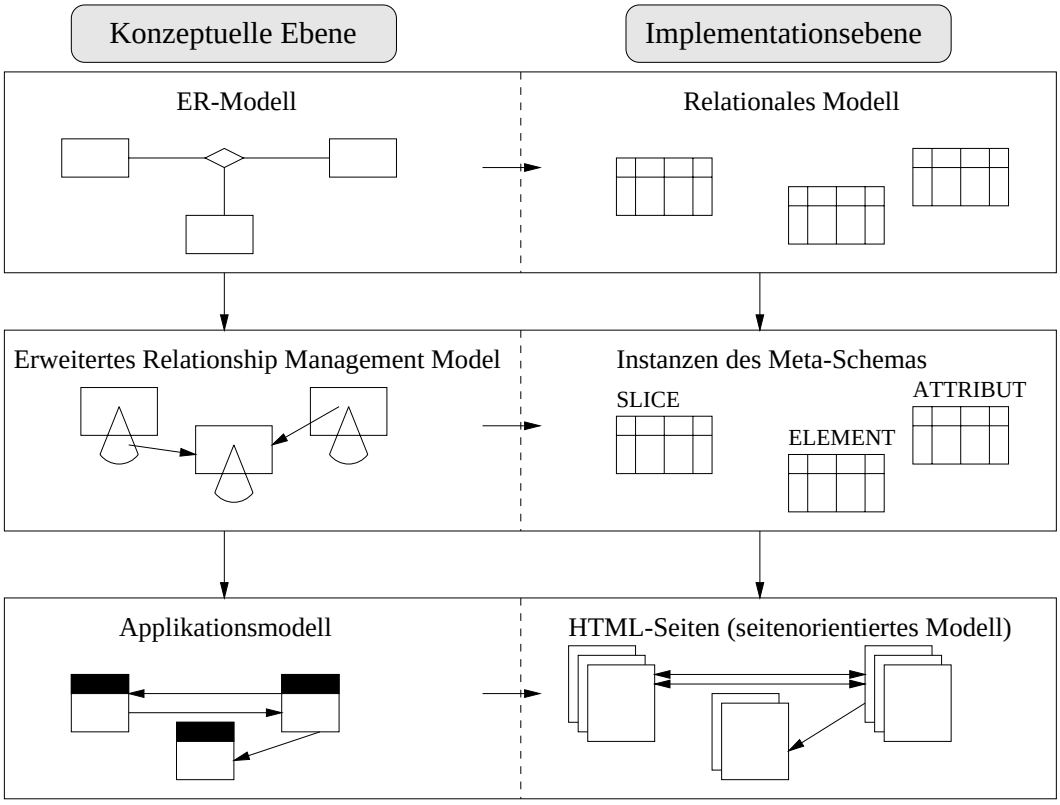


Abbildung 5.19: Abstraktionsebenen und Datenmodelle der HDBM

Dies ermöglicht sowohl die konzeptuelle Modellierung der Anwendungsdomäne als auch die Modellierung von Hypertext-Struktur und Navigation einer Web-Site. Im Verlauf des logischen DB-Designs wird die Anwendungsdomäne auf ein relationales Datenmodell abgebildet. Zur logischen Modellierung von Hypertext-Struktur, Navigation und Layout in

einer relationalen Datenbank wurde ein Meta-Schema entworfen, das als Erweiterung des Systemkatalogs der Datenbank dient. Im Verlauf des logischen Hypertext-Design werden die Struktur und das Layout der Web-Anwendung auf Instanzen des Meta-Schemas abgebildet. Die Relationen des Meta-Schemas zusammen mit den Relationen des Anwendungsbereichs bilden das HDBM-Web-Datenbank-Schema.

Die Erfassung von Hypertext-Struktur, Layout und Navigation zusammen mit der Anwendungsdomäne innerhalb einer Datenbank gewährleistet die einfache Entwicklung, Wartung und Skalierbarkeit datenbankbasierter Web-Sites. Abbildung 5.19 zeigt abschließend die verschiedenen Abstraktionsebenen und korrespondierenden Datenmodelle der HDBM.

Kapitel 6

Der Web-Site-Generator

Im folgenden Kapitel wird die Systemarchitektur des prototypisch implementierten Web-Site-Generators erläutert. Dieser realisiert die dynamische Generierung von HTML-Seiten auf der Basis des im letzten Kapitel beschriebenen Meta-Schemas.

6.1 Systemarchitektur

Bei datenbankgestützten Internet-Anwendungen sind neben der reinen Datenspeicherung innerhalb eines Datenbanksystems auch die Aufbereitung und Präsentation der Informationen in einem Web-Browser wichtig. Dadurch ergeben sich drei verschiedene Modellebenen:

- **Datenspeicherungsebene**: Diese Ebene beschreibt, wie die Inhalte von HTML-Seiten gespeichert und in einer Datenbank organisiert werden.
- **Interaktionsebene**: Diese Ebene beschreibt die Struktur der Informationen und der Hypertext-Anwendung und legt Interaktions- und Navigationsmöglichkeiten fest, die dem Benutzer zur Verfügung stehen.
- **Präsentationsebene**: Diese Ebene beschreibt das konkrete Aussehen der Daten und das Layout der Hypertext-Anwendung.

Abhängig von der gewählten Internet-Kopplung des Datenbanksystems und der logischen Modellierung der Hypertext-Anwendung wird die Funktionalität der einzelnen Ebenen physisch auf verschiedenen Einheiten implementiert. Bei einer separaten Speicherungsebene existiert eine klare Trennung zwischen der Speicherung der Anwendungsdaten innerhalb einer Datenbank und den Ebenen der Präsentation und Interaktion. Die Präsentations- und Interaktionsebene werden durch Templates, Skripten oder Programme realisiert, die außerhalb der Datenbank abgelegt werden (siehe auch Kapitel 5.4.1).

Eine andere Möglichkeit ist die Integration aller konzeptueller Ebenen in einem Datenbanksystem. Das DBMS verwaltet nicht nur die Anwenderdaten sondern auch die

Strukturierung des Hypertextes und die Navigation zwischen den HTML-Seiten. Gleichzeitig regelt das DBMS mit Hilfe von Methoden oder Stored Procedures die Aufbereitung der Informationen für den Benutzer und generiert den HTML-Code, der anschließend im Web-Browser angezeigt wird. Diese Variante bringt viele Vorteile, wie die einfache Wartbarkeit, Skalierbarkeit und Konsistenz der Hypertext-Anwendung mit sich.

In HDBM wurde deshalb die zweite Variante mit den Oracle-Systemkomponenten Oracle 8.0.5 und Web Application Server 3 gewählt. Neben den genannten Vorteilen der integrierten Verwaltung, bietet das Oracle-System eine der fortschrittlichsten auf dem Markt befindlichen Web-Anbindungen von Datenbanksystemen. Transaktionen über mehrere Seiten, sitzungsorientiertes Arbeiten und eine komplexe Benutzerauthentifizierung auf verschiedenen Ebenen (Web-Server, Datenbanksystem) garantieren die in Kapitel 2.2.1 diskutierten Systemanforderungen an eine Datenbankkopplung. Abbildung 6.1 zeigt die Basisarchitektur des Systems. Das Datenbanksystem umfaßt neben Datenbank und Data Dictionary der Datenbank auch die HDBM Meta-Daten. Erfassung und Wartung der Meta-Daten geschehen online über eine Benutzerschnittstelle im Web-Browser. Auf der Basis der Meta-Daten, dem Data Dictionary und den Daten des Anwendungsbereichs generiert ein hierfür entwickeltes Tool, der `webgenerator`, dynamische HTML-Seiten aus der Datenbank. Dabei stehen für die Implementierung des `webgenerator` zwei verschiedene Varianten zur Auswahl. Die Meta-Daten können entweder zur Laufzeit ausgewertet und für die dynamische Generierung der HTML-Dokumente eingesetzt oder bereits im Vorfeld ausgewertet und zur dynamischen Generierung von PL/SQL-Prozeduren verwendet werden, die selbst nicht mehr auf Meta-Daten zugreifen. In HDBM wurden beide Alternativen prototypisch implementiert. Eine genaue Erläuterung der Oracle-Systemarchitektur und der verschiedenen Implementierungsvarianten folgt in den nächsten Kapiteln.

Der HDBM-Web-Site-Generator wurde in folgender Systemumgebung entwickelt:

- **Hardware:** Sun Ultra 1, 143 MHz, 0,5MB ECache, 64 MB Speicher, 4GB Platte
- **Betriebssystem:** Sun Solaris 2.5
- **Datenbanksystem:** Oracle 8.0.5
- **Anbindungssoftware:** Oracle Application Server 3.0

6.2 Der Oracle Web Application Server 3.0

Im folgenden wird die Kernfunktionalität des Oracle Web Application Server 3.0 (OAS) vorgestellt, soweit dies zum Verständnis der folgenden Kapitel erforderlich ist. Für eine genauere Beschreibung sei auf die umfangreiche Fachliteratur verwiesen [Ora96], [HB98a].

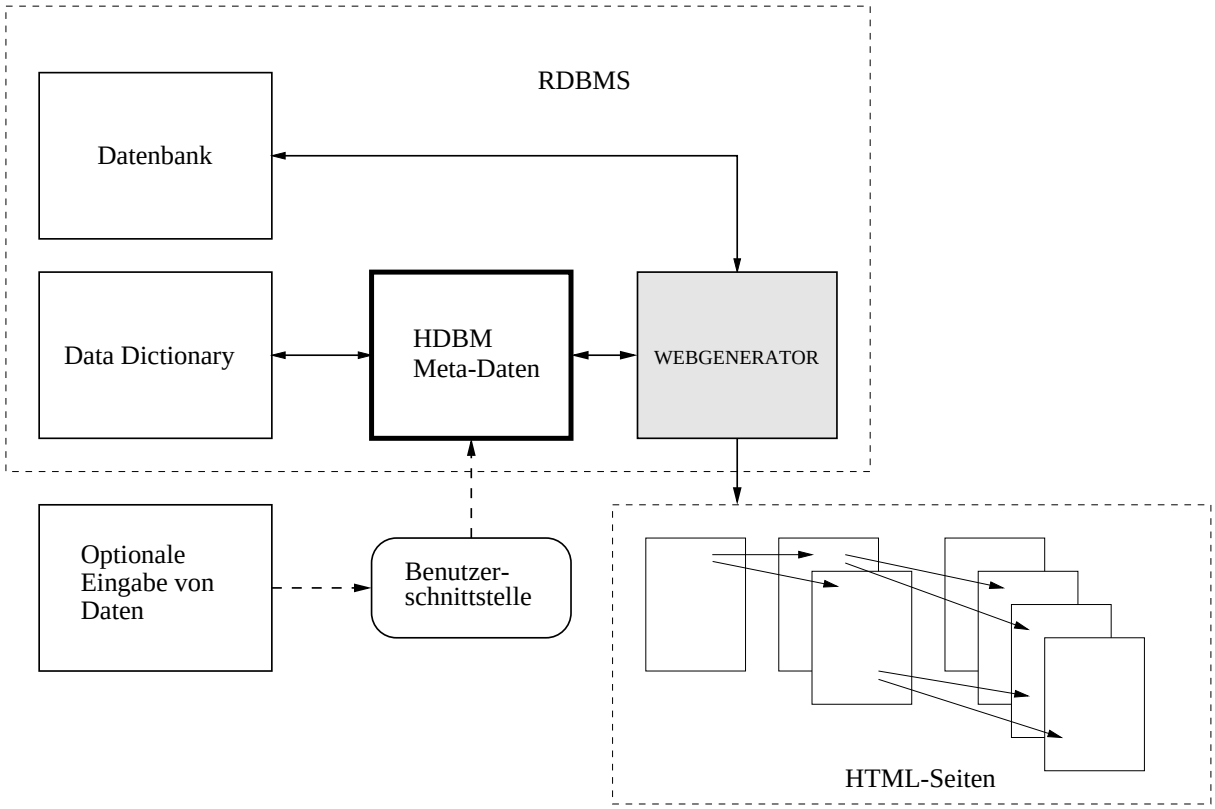


Abbildung 6.1: Dynamische Generierung von HTML-Seiten

6.2.1 Architektur

Der Oracle Web Application Server 3.0 besteht aus folgenden Komponenten, die durch Abbildung 6.2 veranschaulicht werden:

- Web Listener
- Dispatcher
- Web Request Broker (WRB)
- Cartridges

Zusammen mit dem OAS 3.0 wird ein Oracle-HTTP-Listener ausgeliefert. HTTP-Listener und Dispatcher bilden zusammen einen Prozeß. Dieser hat die Aufgabe, eingehende Anforderungen entgegenzunehmen und zu verteilen. Dabei kann es sich um die Anforderung statischer HTML-Dokumente, den Aufruf von CGI-Programmen oder den Verbindungswunsch zu einer Cartridge (beispielsweise PL/SQL-Cartridge) handeln. Nur im letzten Fall wird die Anforderung vom Web-Listener an den Dispatcherteil weitergeleitet. Der OAS 3.0 ist so ausgelegt, daß parallel viele Listener und Dispatcher betrieben werden können, so daß auch die Bearbeitung vieler Anfragen mit geringen Antwortzeiten möglich ist.

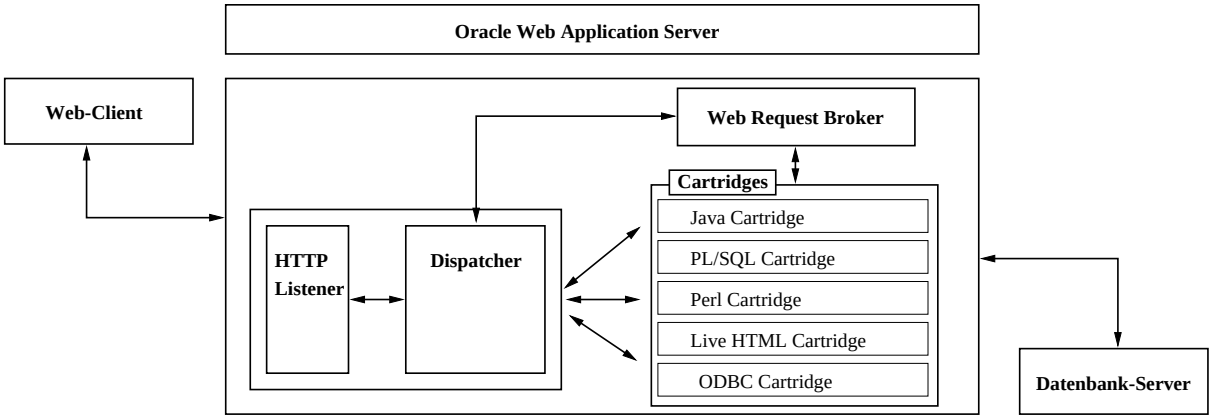


Abbildung 6.2: Architektur des Oracle Web Application Server 3.0

Der Dispatcher bildet eine CORBA-Schnittstelle zwischen dem Web-Listener, dem Web Request Broker und einer WRB Cartridge und verwaltet die zu einer Cartridge angesteuerten Requests. Erhält der Web-Listener den Verbindungswunsch zu einer WRB-Cartridge, so wird dieser Request an den Dispatcher weitergeleitet. Dieser schickt den Request zu der in dem URL angesprochenen Cartridge weiter. Ist gerade keine Cartridge-Instanz aufnahmebereit, verlangt der Dipatcher vom Web Request Broker die Erzeugung einer neuen Instanz. Nach dem erfolgreichem Start der Instanz wird der Dispatcher vom WRB benachrichtigt und leitet den Request an die Instanz weiter. Die eindeutige Zuordnung jedes Web-Clients zu einer Cartridge-Instanz gewährleistet sitzungsorientiertes Arbeiten im Web.

Die dritte Komponente bildet der Corba 2.0-kompatible Web-Request-Broker (WRB). Seine Aufgabe ist es, die Kontrolle über das Gesamtsystem zu gewährleisten. Hierzu gehören beispielsweise die Lastverteilung, die Verwaltung von globalen Adressen von Ressourcen und die Verwaltung der Objektreferenzen für verfügbare Cartridges. Diese werden vom Dispatcher zum Aufruf entsprechender Funktionen genutzt. Weiterhin bietet der WRB eine ganze Reihe von Diensten, wozu beispielsweise der Transaction Service, Logger und Authentification Server gehört.

Zusammen mit dem WRB wird ein umfangreicher Satz von Cartridges ausgeliefert. Hierzu gehören die PL/SQL Cartridge, Java Cartridge, PERL Cartridge, ODBC Cartridge und LiveHTML Cartridge. Eine Cartridge besteht zum einen aus Code zur Ausführung der Applikations-Logik und zum anderen aus Konfigurations-Meta-Daten zur Definition des eigenen Laufzeitverhaltens. Die PL/SQL-Cartridge besteht beispielsweise aus Code, der die Anbindung an eine Oracle-Datenbank und Ausführung der Stored Procedures in der Datenbank ermöglicht. Die Meta-Daten beinhalten Informationen zu welcher Datenbank die Verbindung stattfinden soll und welcher Benutzername/Paßwort beim Anmelde-Vorgang verwendet werden soll.

Damit eine PL/SQL Cartridge auf die Datenbank zugreifen kann, sind folgende zwei Definitionen erforderlich:

- Der *Database Access Descriptor* (DAD) beschreibt, mit welchen Benutzerangaben (Login und Paßwort) auf eine Datenbank zugegriffen werden soll. Dabei muß für jede Datenbank mindestens ein DAD angelegt werden.
- Der *PL/SQL-Agent* legt die globalen Angaben für den Zugriff fest. Hierzu gehören, mit welchem DAD auf eine Datenbank zugegriffen werden soll, ob ein Zugriffsschutz vorhanden ist, welcher Port benutzt werden soll u.a.

Durch die Trennung von DAD und PL/SQL-Agent ist es möglich, mehrere Agenten mit dem gleichen DAD zu betreiben. Der Name des gewünschten PL/SQL-Agenten und der PL/SQL-Prozedur wird in dem URL an den HTTP-Listener übergeben (siehe auch Kapitel 6.2.2).

Ein Bestandteil der PL/SQL-Cartridge sind eine Reihe von PL/SQL-Paketen, die in der Datenbank installiert werden müssen. Hierzu gehört das *HTP-Paket* und das *OWA-Paket*. Das HTP-Paket enthält einen Satz von Prozeduren, die HTML-Tags generieren. So legt beispielsweise die Prozedur `htp.headerOpen` den Tag `<HEADER>` an. Daneben existiert auch noch ein HTF-Paket, bei dem alle Prozeduren des HTP-Pakets auf Funktionen abgebildet werden. Das OWA-Paket besteht aus einer Reihe von PL/SQL Einzelpaketen, in denen nützliche Prozeduren für den Aufbau von HTML-Seiten bereitgestellt werden (z.B. Cookies).

6.2.2 Aufbau von PL/SQL-URLs

Die Adressierung von HTML-Dokumenten im WWW erfolgt über URLs (**U**niform **R**esource **L**ocator). Der URL beschreibt das Protokoll, mit dem der Zielserver erreicht wird, das Zielsystem (oder den Servernamen), auf dem das Dokument liegt, den Verzeichnispfad zum Dokument sowie dessen Dateinamen. Das Format eines URL ist dabei vereinfacht wie folgt definiert:

```
http://hostname:port/path/to/the/resource
```

- `http://`: Bezeichnet das Hypertext-Transfer-Protokoll
- `hostname`: Bezeichnet den Domain-Namen des WWW-Servers
- `port`: Bezeichnet die Port-Nummer des WWW-Servers
- `path`: Bezeichnet den Pfad eines Dokuments oder ein auf dem Server ausführbares Programm.

Bei der dynamischen Generierung von HTML-Seiten aus einer Datenbank, wird in dem URL der Name des Skriptes/Programmes aufgerufen, das die Verbindung zur Datenbank

herstellt. Vor Start eines Skripts belegen Web-Server einige Umgebungsvariablen mit Werten, die von Programmen und Skripten genutzt werden können. Die wichtigsten drei Umgebungsvariablen für die Parameterübergabe sind *QUERY_STRING*, *PATH_INFO* und *SCRIPT_NAME*.

Parameter, die nach dem `?` in dem URL folgen, werden in der vom Web-Server belegten Umgebungsvariablen *QUERY_STRING* übergeben. Die Stringübergabe kann entweder implizit durch das Abschicken eines Formulars (Methode `get`) oder explizit durch das Anhängen der Parameter an den URL erfolgen. Bei Verwendung der Methode `post` erfolgt die Stringübergabe über die Standardeingabe. Parameter werden durch *name=value-Paare*, die mit `&` getrennt sind, übertragen [Ste95].

```
/cgi-bin/lookup?author=Poe&title=The%20Raven
```

Die vom Web-Server belegte Umgebungsvariable *PATH_INFO* enthält den Teil des URL nach der Identifikation des Skripts.

```
/cgi-bin/lookup/Edgar/Allen/Poe
```

Der Name unter dem das Skript/Programm in dem URL angefordert wird, ist in der Umgebungsvariablen *SCRIPT_NAME* abgelegt.

Bei der Übergabe von Eingabeparametern an PL/SQL-Prozeduren werden die drei vorgestellten Umgebungsvariablen, wie in der Tabelle 6.1 dargestellt, belegt.

CGI-Variable	*Wert*
SCRIPT_NAME	/plsql-Agent/plsql
PATH_INFO	package.procedure
QUERY_STRING	name1=value1&name2=value2

Tabelle 6.1: Aufruf von PL/SQL-Prozeduren in Hyperlinks

Der Basis-URL zum Zugriff auf eine PL/SQL-Prozedur sieht damit wie folgt aus:

```
http://hostname:port/owa_webdb/plsql/package.procedure?name1=value1
```

wobei in diesem speziellen Fall `owa_webdb` der Name des PL/SQL-Agenten ist.

6.3 Generierung von dynamischen HTML-Seiten

Bei der dynamischen Generierung von HTML-Seiten werden die HTML-Seiten erst zum Abfragezeitpunkt aus der Datenbank aufgebaut. Dies bringt viele Vorteile, wie beispielsweise die referenzielle Integrität der berechneten Hyperlinks, mit sich. Jedes Element einer HTML-Seite kann abhängig vom Datenbankinhalt, Benutzerprofil und Navigationskontext zur Laufzeit angezeigt oder weggelassen werden. Für jeden HTML-Seitentyp werden

üblicherweise in einem HTML-Template die genauen Datenbankabfragen und das Layout des Anfrageergebnisses definiert. In HDBM wurden hierfür Oracle-PL/SQL-Prozeduren verwendet. Durch den Einsatz eines Meta-Schemas ist die automatische Erstellung einer Hypertext-Applikation ohne HTML- und SQL-Kenntnisse und ohne Implementierungsaufwand möglich. Dabei existieren prinzipiell zwei Implementierungsvarianten zur Auswertung der Meta-Daten:

- **Auswertung zur Laufzeit:** Eine PL/SQL-Prozedur greift zur Laufzeit auf die Meta-Daten zu und generiert die vom Anwender abgefragte Instanz eines Slice-Typs. Der Name des Slice-Typs und der Primärschlüssel der Owner-Relation werden als Parameter übergeben.
- **Auswertung im Vorfeld:** Mit Hilfe der Meta-Daten wird für jeden Slice-Typ im Vorfeld genau eine PL/SQL-Prozedur generiert und in der DB gespeichert. Die vom Anwender aufgerufenen PL/SQL-Prozeduren greifen selbst nicht auf Meta-Daten zu. Als Eingabeparameter erhalten sie den Primärschlüssel der Owner-Relation zur Identifizierung der Seiteninstanz. Eine Änderung der Meta-Daten führt zur neuen Generierung der betroffenen PL/SQL-Programme.

Nachteil der ersten Variante ist vor allem die schlechtere Performance. Dies ist durch die Auswertung der Meta-Daten zur Laufzeit bedingt. Da in der zweiten Variante die Meta-Daten bereits im Vorfeld ausgewertet und zur Generierung von PL/SQL-Prozeduren genutzt werden, bringt dies Performance-Verbesserungen mit sich. Nachteil ist, daß sich im Meta-Schema durchgeführte Änderungen nicht automatisch auf die materialisierten PL/SQL-Programme propagieren. Beide Ansätze sollen in den nächsten Kapiteln vorgestellt und diskutiert werden.

6.3.1 Auswertung der Meta-Daten zur Laufzeit

Abhängig von dem als Parameter übergebenen Slice-Typ und dem Primärschlüssel einer Instanz der Owner-Relation generieren PL/SQL-Prozeduren dynamische HTML-Seiten. Ein großer Vorteil dieser Variante liegt in der einfachen Konfigurierbarkeit der Hypertext-Oberfläche. Durch die Auswertung der Meta-Daten zur Laufzeit werden Änderungen der Meta-Daten, die in der Datenbank durchgeführt wurden, sofort bei der Generierung des HTML-Seitentyps sichtbar. Dieser Ansatz wurde in HDBM gewählt und implementiert (siehe Abbildung 6.3).

Die Implementierung des HTML-Seiten-Generators setzt sich im wesentlichen aus zwei Prozeduren und vier Funktionen [Jan00] zusammen. Alle Prozeduren und Funktionen sind in einem PL/SQL-Paket `webgenerator` implementiert. Über einen URL können ausschließlich die beiden Prozeduren `zeige_slice` und `zeige_zugriffsstruktur` aufgerufen werden. Die Prozedur `zeige_slice` ermöglicht die Darstellung eines DB-Tupels auf einer HTML-Seite. Sie erhält als Eingabeparameter den Slice-Typ und den Primärschlüssel einer Instanz der Owner-Relation. In HDBM wird vereinfacht von einem eindeutigen

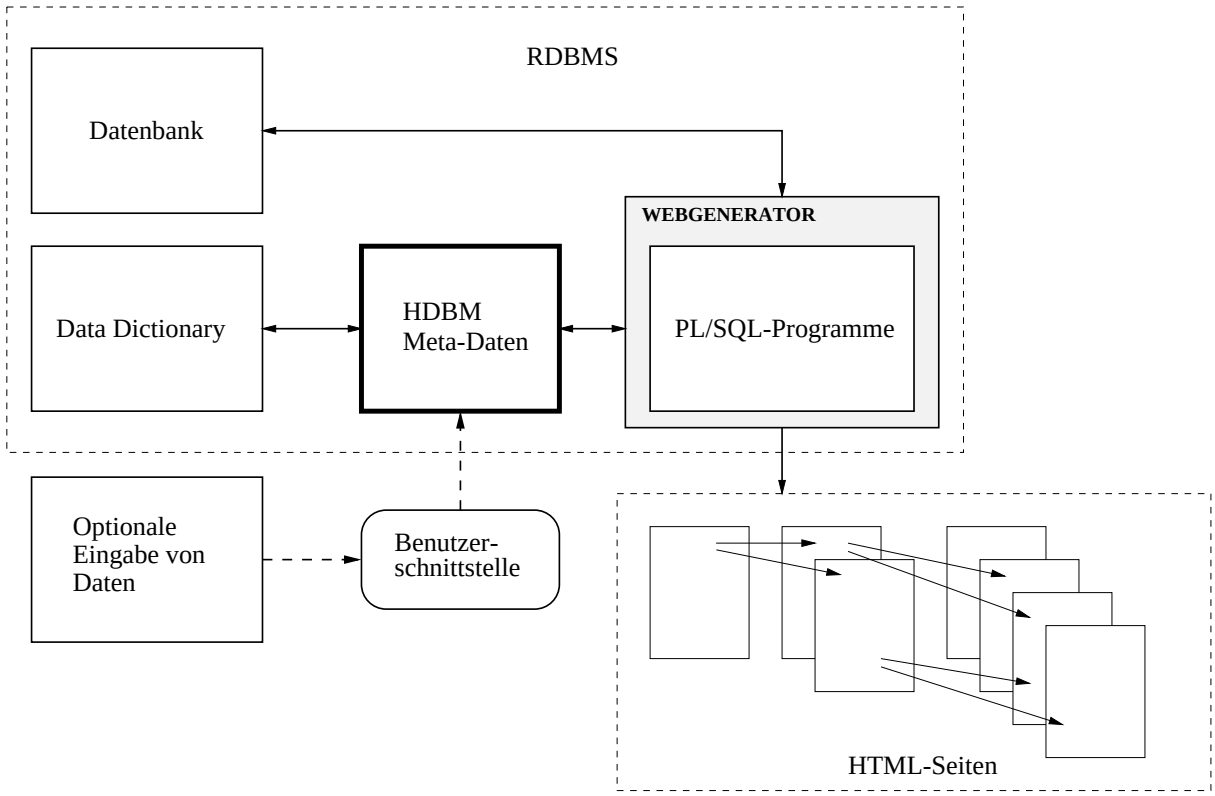


Abbildung 6.3: Auswertung der Meta-Daten zur Laufzeit

Primärschlüssel ausgegangen, der genau aus einem Attribut besteht. Prinzipiell ist aber durch eine Erweiterung des Programms auch die Behandlung zusammengesetzter Primärschlüssel möglich, wie im Verlauf des Kapitels noch gezeigt wird.

Der Aufruf der Prozedur `zeige_slice` und die Übergabe der Parameter erfolgen innerhalb des URL. Folgendes Beispiel zeigt den URL zur Generierung einer Mitarbeiter-HTML-Seite:

```
.../owa_webdb/plsql/webgenerator.zeige_slice?slice=MitarbInfo&keywert=5
```

wobei der Name des gewünschten Slice-Typs `MitarbInfo` und der Primärschlüssel einer Instanz der Owner-Relation `5` ist. Die Prozedur `zeige_zufriffsstruktur` zeigt eine Menge von gleich strukturierten DB-Tupeln als Liste auf einer HTML-Seite an (Index Extern).

Die Prozedur erhält als Eingabeparameter die eindeutige Element-Id (ELEM#), die Owner-Relation der Quell-HTML-Seite (TABNAME), das Schlüsselattribut (KEY) und den Wert des Schlüssels. Alle anderen Informationen können aus der Relation ZUGRIFFSSTRUKTUR des Meta-Schemas entnommen werden. Folgendes Beispiel soll dies verdeutlichen:

```
.../owa_webdb/plsql/webgenerator.zeige_zugriffsstruktur?
              elem_in=13&tabelle_in=MITARBEITER&keyattr=M#&keywert=5
```

Die Generierung von Guided Tours und Indexed Guided Tours wurde im Prototyp nicht implementiert.

Die beiden Prozeduren `zeige_slice` und `zeige_zugriffsstruktur` nutzen vier Funktionen zur Generierung der HTML-Seiten bzw. der Listenelemente: `hole_element`, `hole_attribut`, `hole_hyperlink` und `hole_zugriffsstruktur`. Zusammengefaßt lassen sich die dynamisch generierten HDBM-Hyperlinks in der erweiterten Backus-Naur-Form (EBNF) (siehe Anhang A) wie folgt darstellen:

⟨hdbm-url⟩ ::= **http://**⟨host⟩/⟨hdbm-path⟩

⟨host⟩ ::= (Internet-Adr)[:⟨port⟩]

⟨port⟩ ::= ⟨number⟩

⟨number⟩ ::= **0** | **1** | ... | **32767**

⟨hdbm-path⟩ ::= ⟨script-name⟩/⟨query_info⟩/

⟨script-name⟩ ::= **/owa_webdb/plsql/**

⟨query-info⟩ ::= **webgenerator.zeige_slice?**⟨query-string1⟩ |
webgenerator.zeige_zugriffsstruktur?⟨query-string2⟩

⟨query-string1⟩ ::= **slice=**(Wert)**&keywert=**(Wert)

⟨query-string2⟩ ::= **elem_in=**(Wert)**&tabelle_in=**(Wert)**&**
keyattr=(Wert)**&keywert=**(Wert)

Die Funktionsweise der Prozeduren und Funktionen des Pakets `webgenerator` sollen im folgenden kurz vorgestellt werden.

Paket:	`webgenerator`
Prozedur:	`zeige_slice`
INPUT:	`slice`: Name eines Slice-Typs (SNAME), `keywert`: Wert des Primärschlüssels einer Instanz
Beschreibung:	Zeigt ein Datenbank-Tupel auf einer HTML-Seite an. Selektiert zu dem jeweiligen Slice-Typ HEADER, FOOTER und BACKGROUND aus der Relation PRAESENTATION und legt ein entsprechendes HTML-Dokument an. Startet anschließend die Funktion `hole_element`.
Verwendete Meta-Relationen:	SLICE, PRAESENTATION
Prozedur:	`zeige_zugriffsstruktur`
INPUT:	`elem_in`: Element-Id (ELEM#) `tabelle_in`: Name der Owner-Relation (TABNAME) `keyattr`: Primärschlüssel-Attribut `keywert`: Wert des Primärschlüssels
Beschreibung:	Zeigt eine Menge von Tupeln auf einer HTML-Seite an. Selektiert mit Hilfe der Element-Id das zugehörige Tupel aus der Relation ZUGRIFFSSTRUKTUR. Öffnet in dem HTML-Dokument eine HTML-Liste. Führt Join zwischen Quell- und Zielrelation durch und startet für jedes Ergebnistupel die Funktion `hole_element`. Das Ergebnis wird als HTML-Liste ausgegeben.
Verwendete Meta-Relationen:	ZUGRIFFSSTRUKTUR, SLICE
Funktion:	`hole_element`
INPUT:	`slice`: Name des Slice-Typs (SNAME) `keywert`: Wert des Primärschlüssels
OUTPUT:	HTML-Tabelle mit Elementen eines Slice-Typs
Beschreibung:	Selektiert den Namen der Owner-Relation (TABNAME) und das Schlüssel-Attribut aus dem Data Dictionary. Selektiert alle Elemente des übergebenen Slice-Typs und sortiert diese nach der Position (POSITION). Holt ein Element nach dem anderen. Öffnet eine HTML-Tabelle im HTML-Dokument, schreibt die Bezeichnung des Elements (BEZEICHNUNG) mit Doppelpunkt an den Beginn der Zeile und startet für jedes Element abhängig davon, ob es sich um ein Attribut, Hyperlink oder eine Zugriffsstruktur handelt (SUBTABLE) eine der Funktionen `hole_attribut`, `hole_hyperlink` oder `hole_zugriffsstruktur`. Schließt die HTML-Tabelle.
Verwendete Meta-Relationen:	ELEMENT, SLICE

Funktion:	`hole_attribut`
INPUT:	`elem_in`: Element-Id (ELEM#) `tabelle_in`: Name der Owner-Relation (TABNAME) `keyattr`: Primärschlüssel-Attribut `keywert`: Wert des Primärschlüssels
OUTPUT:	Attributwert und Web-Typ
Beschreibung:	Selektiert den Attributnamen (COLNAME) und zugehörigen Attributwert aus der Owner-Relation. Selektiert Web-Typ und gibt Attibutwert mit Web-Typ zurück
Verwendete Meta-Relationen:	ATTRIBUT, WEB_TYP

Funktion:	`hole_hyperlink`
INPUT:	`elem_in`: Element-Id (ELEM#) `tabelle_in`: Name der Owner-Relation (TABNAME) `keyattr`: Primärschlüssel-Attribut `keywert`: Wert des Primärschlüssels
OUTPUT:	Hyperlink oder Hyperlinkliste
Beschreibung:	Selektiert Anker- und Zielslice (ANKER, ZIEL) des Hyperlinks. Holt nach der Position (POSITION) geordnet alle Attribut-Namen (COLNAME) des Ankerslices aus der Relation ANKERSLICE. Führt Join zwischen Quell- und Zielrelation durch und selektiert Schlüssel zur URL-Generierung und die Werte der Ankerattribute aus der Zielrelation. Generiert einen Hyperlink oder eine Hyperlink-Liste aus dem Slice-Typ und dem Primärschlüssel. Zeigt Ankerattribute als Hyperlink an.
Verwendete Meta-Relationen:	HYPERLINK, ANKERSLICE

Funktion:	`hole_zugriffsstruktur`
INPUT:	`elem_in`: Element-Id (ELEM#) `tabelle_in`: Name der Owner-Relation (TABNAME) `keyattr`: Attribut zur URL-Generierung `keywert`: Wert des Primärschlüssels
OUTPUT:	Hyperlink falls externe Zugriffsstruktur, interne Zugriffsstruktur sonst
Beschreibung:	Überprüft Zugriffstyp (ZTYP): falls es sich um internen Index handelt wird Join zwischen Quell- und Zielrelation durchgeführt und Primärschlüssel selektiert. Aufruf der Funktion `hole_element` für jeden selektierten Primärschlüssel. Falls es sich um eine externe Zugriffsstruktur handelt wird der URL mit Funktion `zeige_zugriffsstruktur` generiert.
Verwendete Meta-Relationen:	SLICE, ZUGRIFFSSTRUKTUR

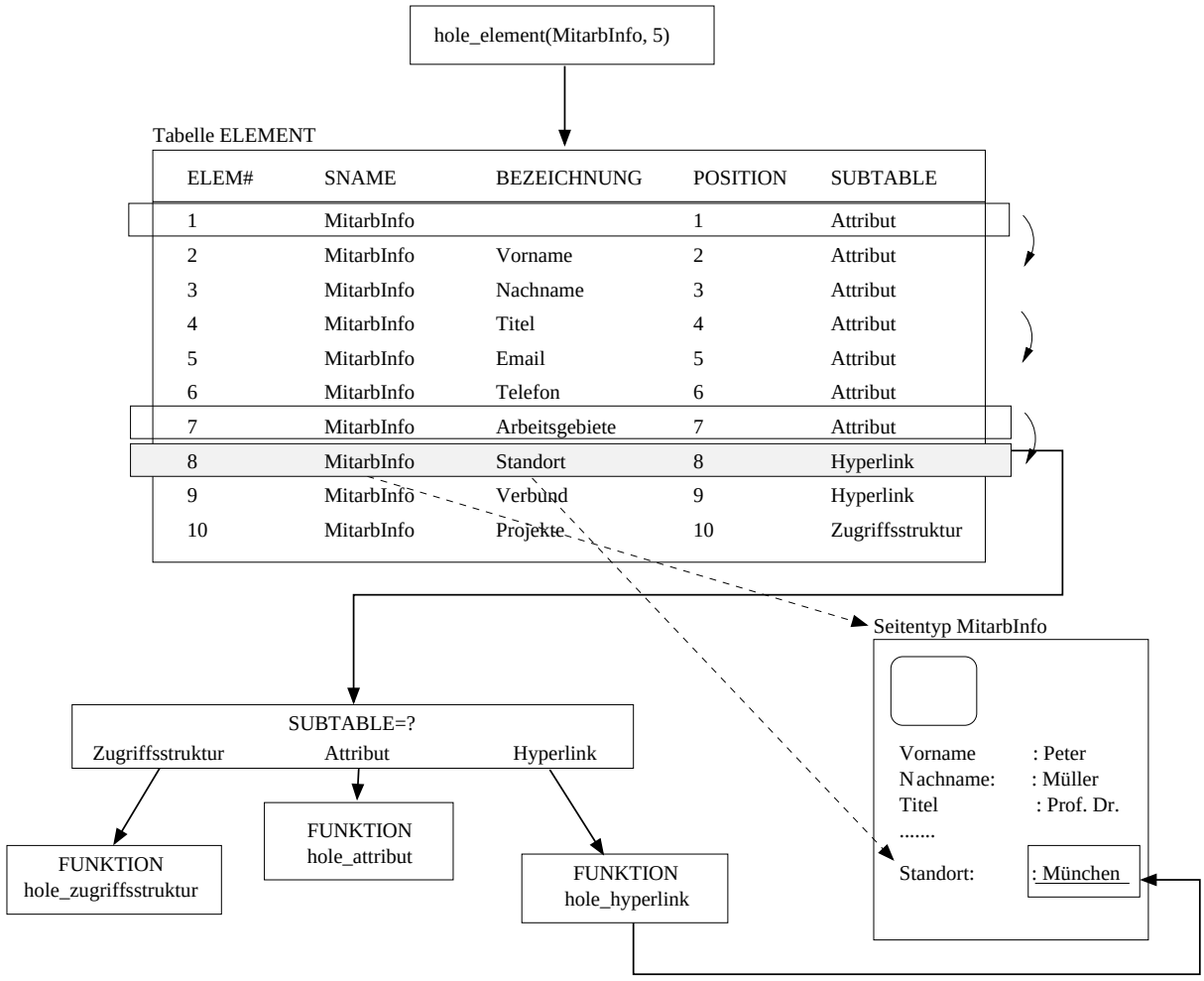


Abbildung 6.4: Funktion: hole_element

Die Funktion `hole_element` selektiert ein Element nach dem anderen eines HTML-Seitentyps (siehe Abbildung 6.4). Anhand des Attributs SUBTABLE wird in Abhängigkeit davon, ob es sich um ein Attribut, einen Hyperlink oder eine Zugriffsstruktur handelt eine der drei Funktionen `hole_attribut`, `hole_hyperlink` und `hole_zugriffsstruktur` aufgerufen. Die jeweilige Funktion gibt das zugehörige Element auf der HTML-Seite aus. Ein geschachtelter Slice wird über die Relation ZUGRIFFSSTRUKTUR und den Zugriffstyp INTERN realisiert.

In HDBM wird von einem eindeutigen, invarianten Identifikator aller Instanzen ausgegangen. Dies trifft allerdings für legacy-Datenbanken nicht immer zu. Bei der Übergabe von zusammengesetzten Primärschlüsseln ergibt sich folgendes Problem: da der Slice-Typ selbst als Parameter übergeben wird, ist die Anzahl der zur Identifikation eines DB-Tupels benötigten Schlüsselattribute nicht festgelegt. Dies führt dazu, daß die Anzahl der Eingabeparameter der Prozedur `zeige_slice` nicht feststeht. In PL/SQL kann dieses Problem beispielsweise durch die Deklaration einer Prozedur mit einem Eingabeparameter vom Typ `ident_arr` des `owa_util`-Paketes gelöst werden, wobei `ident_arr` eine PL/SQL-Tabelle mit einer beliebigen Anzahl von Werten (maximal 30) ist:

```
type ident_arr is table of varchar2(30) index by binary_integer
```

Die Werte müssen dabei von dem Basis-Typ VARCHAR2 sein, können aber anschließend in einen anderen Datentyp konvertiert werden. Die Deklaration der Prozedur `zeige_slice` sieht in diesem Fall wie folgt aus:

```
CREATE PROCEDURE zeige_slice (slice IN VARCHAR2, keywert IN
owa_util.ident_arr)
IS
key_counter INTEGER;
BEGIN
...
END;
```

Auf die einzelnen Schlüsselwerte wird mit Hilfe von `keywert(key_counter)` zugegriffen. Die hierzu gehörende URL für eine Mitarbeiter-Seite wäre beispielsweise:

```
.../webgenerator.zeige_slice?slice=MitarbInfo&keywert=Peter&keywert=Baumann
```

wobei Vorname und Nachname des Mitarbeiters den zusammengesetzten Primärschlüssel bilden. Für die Zuordnung von Schlüsselwert und Attributbezeichnung gibt es zwei verschiedene Varianten. Bei der ersten Variante wird von einer festen Reihenfolge der Primärschlüssel-Werte ausgegangen (wie im obigen Beispiel). Diese ergibt sich aus der Reihenfolge der Definition der Schlüsselattribute im Datenbank-Schema. In Oracle können die Primärschlüssel-Attribute einer Relation beispielsweise durch folgende select-Anfrage aus dem Data Dictionary selektiert werden:

```
select C.COLUMN_NAME
from ALL_CONS_COLUMNS C, ALL_CONSTRAINTS A
where A.TABLE_NAME='MITARBEITER' and
      A.CONSTRAINTS_TYPE='P' and
      C.CONSTRAINT_NAME=A.CONSTRAINT_NAME
```

Eine andere Möglichkeit ist die explizite Erfassung der Schlüsselattribute in einem weiteren Eingabeparameter:

```
CREATE PROCEDURE zeige_slice (slice IN VARCHAR2, keyattr IN
owa_util.ident_arr, keywert IN owa_util.ident_arr)
IS
keyattr_counter INTEGER;
keywert_counter INTEGER;
BEGIN
...
END;
```

Die hierzu gehörende URL für eine Mitarbeiter-HTML-Seite sieht in diesem Fall wie folgt aus:

```
.../webgenerator.zeige_slice?slice=MitarbInfo&keyattr=VNAME&keyattr=NNAME&
                              keywert=Peter&keywert=Baumann
```

In HDBM wurde vereinfacht von einem eindeutigen Schlüsselattribut in jeder Basis-Relation ausgegangen.

6.3.2 Auswertung der Meta-Daten im Vorfeld

Bei der Auswertung der Meta-Daten im Vorfeld wird für jeden Slice-Typ eine PL/SQL-Prozedur generiert (siehe Abbildung 6.5). Die PL/SQL-Prozeduren greifen selbst nicht mehr auf Meta-Daten zu sondern setzen direkt Anfragen an die Basis-Relationen ab.

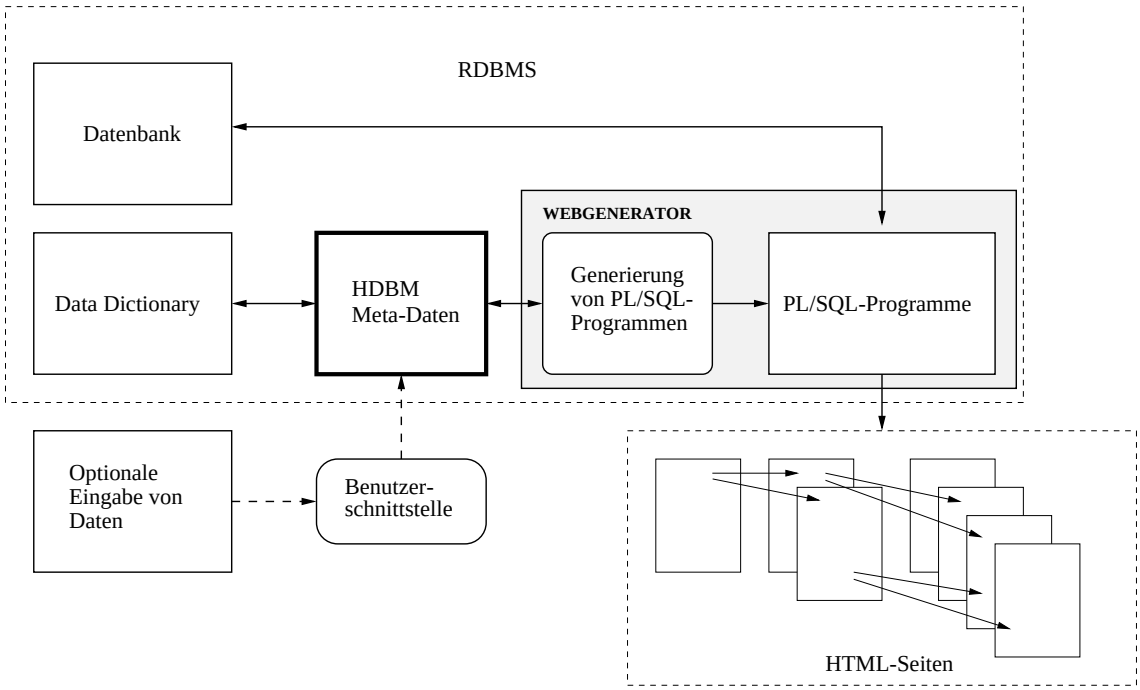


Abbildung 6.5: Auswertung der Meta-Daten im Vorfeld

Als Eingabeparameter erhalten die PL/SQL-Prozeduren die Primärschlüssel-Attribute der zum Slice-Typ gehörenden Owner-Relation. Zur eindeutigen Zuordnung eines Slice-Typs zu einer PL/SQL-Prozedur, wird die jeweilige Prozedur nach dem Slice-Typ benannt. Die Deklaration der PL/SQL-Prozedur zur dynamischen Generierung der Mitarbeiter-HTML-Seiten sieht beispielsweise wie folgt aus:

```
CREATE PROCEDURE mitarbinfo(M#_key IN INTEGER)
IS
BEGIN ... END;
```

Entsprechend sieht die URL zur Generierung einer Mitarbeiter-Seite wie folgt aus:

```
.../owa_webdb/plsql/webgenerator.mitarbinfo?M#_key=5
```

Da zu jedem Slice-Typ, eine Prozedur existiert, können bei dieser Variante auch zusammengesetzte Primärschlüssel problemlos behandelt werden. Owner-Relation und Slice-Typ stehen bereits im Vorfeld fest.

Das UTL_FILE_Paket ermöglicht PL/SQL-Programmen einfache Lese- und Schreibe-Operationen auf Betriebssystem-Text-Files durchzuführen. Es erlaubt eingeschränkte Versionen von OS stream file I/O mit den bekannten Operationen `open`, `put`, `get` und `close`. Damit kann der im letzten Kapitel vorgestellte Algorithmus zur Auswertung der Meta-Daten zur Laufzeit weitgehend für die Generierung von PL/SQL-Prozeduren verwendet werden. Als Eingabeparameter erhält die Prozedur `zeige_slice` einen Slice-Typ, zu dem eine PL/SQL-Prozedur generiert werden soll. Anschließend werden alle HTML-Tags und SQL-Anfragen an Basis-Relationen in ein BS-File geschrieben. Zugriffsstrukturen werden durch Schleifen realisiert, in denen die Prozedur zur Generierung des Zielslices aufgerufen wird.

6.4 Performance-Messungen

Beide vorgestellten Ansätze haben verschiedene Vor- und Nachteile und eignen sich deshalb für unterschiedliche Anwendungsgebiete. Während die Auswertung der Meta-Daten zur Laufzeit ein größtmögliches Maß an Flexibilität gewährleistet, zeichnet sich der zweite Ansatz durch eine bessere Performance aus. Die Auswertung der Meta-Daten zur Laufzeit eignet sich deshalb vor allem für die iterative Entwicklung des Informationssystems. Die einfache Konfigurierbarkeit der Hypertext-Oberfläche ermöglicht ein schnelles Prototyping und die einfache Anpassung der Hypertext-Oberfläche an unterschiedliche Benutzeranforderungen. Zudem spielt in dieser Phase die Anfragezeit keine entscheidende Rolle. Nach der Fertigstellung der Oberfläche erscheint die Generierung von PL/SQL-Prozeduren durch die Auswertung der Meta-Daten sinnvoll. Die Generierung verschiedener PL/SQL-Prozeduren ermöglicht die Feingestaltung des Layouts der HTML-Seitentypen, was bei der ersten Implementierungsvariante nicht möglich ist. In diesem Zusammenhang sind folgende Fragestellungen hinsichtlich der tatsächlichen Performance beider Varianten interessant:

- Wie hoch ist der Performance-Unterschied von beiden Implementierungsvarianten? Wieviel kostet die Auswertung der Meta-Daten zur Laufzeit?
- Inwieweit verschlechtert sich die Performance bei der Eröhung des Datenvolumens?
- Wieviel kostet der Aufbau einzelner Slice-Elemente (Attribute, Hyperlink, geschachtelte Slices, Zugriffsstrukturen)?

Im folgenden Kapitel wird zunächst das Szenario zur Messung erläutert. Anschließend werden die Meßergebnisse vorgestellt.

6.4.1 Das Szenario für die Messung

Um beide alternativen Implementierungvarianten hinsichtlich der Performance besser vergleichen zu können, wurden verschiedene Performance-Messungen durchgeführt. Bei der Messung der Ausführungszeit von PL/SQL-Prozeduren bietet sowohl der Oracle Application Server 3.0 als auch der Oracle-Datenbankserver verschiedene Alternativen. Der Oracle Application Server ermöglicht die Auswertung der Web-Server-Aktivitäten mit Hilfe eines Log-Analyzers, der die durch einen Logger protokollierten Daten analysiert. Der konfigurierbare Logger speichert unterschiedliche Attribute, wie beispielsweise IP-Adressen der Clients, URL, Query-Stream und CPU-Load in einem Extended Log File Format (XLF)[1]. Die Aktivitäten des Web-Servers können entweder in einer Oracle Datenbank in hierzu gehörenden Log-Tabellen oder im File-System protokolliert werden. Der Log-Analyzer wertet diese Daten aus und liefert Ergebnisse zu der Benutzung einer Web-Site. Hierzu gehört beispielsweise die totale Anzahl der übertragenen Bytes, die am häufigsten abgefragten URLs oder die häufigsten Besucher einer Web-Site. Für Messungen der exakten Ausführungszeit von PL/SQL-Prozeduren eignet sich diese Technik allerdings weniger. Die Messungen wurden deshalb direkt in der Oracle-Datenbank durchgeführt. Eine einfache und schnelle Möglichkeit Antwortzeiten eines Befehls in Oracle zu messen, ist über die Eingabe von `set timing on` im Oracle Server Manager oder in SQL*Plus möglich. Es werden hundertstel Sekunden angezeigt. Folgendes Beispiel soll diese Vorgehensweise verdeutlichen:

```
SVRMGR> connect webdb
Password:
Connected.
SVRMGR> set timing on
Timing                          ON
SVRMGR> execute mitarbinfo(6);
Statement processed.
Parse             0.32 (Elapsed)     0.00 (CPU)
Execute/Fetch     0.23 (Elapsed)     0.00 (CPU)
Total             0.55               0.00
```

Alle PL/SQL-Prozeduren zu beiden Implementierungsalternativen wurden als Stored Procedures in einer Datenbank installiert, mit unterschiedlichen Parametern ausgeführt und mit Hilfe des timing-Befehls im Server Manager gemessen. Um repräsentative Messungen zu erhalten wurden pro Befehl mehrere Messungen durchgeführt und der Mittelwert zum Vergleich der Befehle herangezogen.

Als Datenmodell diente ein vereinfachtes abayfor-Schema, das mit dem aktuellen Datenbestand der abayfor-Datenbank befüllt wurde. Das genaue Datenbankschema ist aus Anhang C ersichtlich. Die Relation MITARBEITER umfaßt 383 Tupel, die Relation

[1] XLF ist eine Obermenge des Common Logfile Format (CLF)

VERBUND umfaßt 12 Tupel, die Relation PROJEKT umfaßt 200 Tupel und die Relation STANDORT umfaßt 250 Tupel. Um das Verhalten bei einem größeren Datenbestand zu veranschaulichen wurde eine weitere Datenbank aufgebaut, in der die Datenmenge von abayfor verfünffacht wurde. Die beiden Datenbanken wurden mit `webdb1` (abayfor Datenbestand) und `webdb2` (fünffacher abayfor Datenbestand) bezeichnet.

Als Hypertext-Schema diente das in Abbildung 5.9 dargestellte Application-Diagramm, das geringfügig hinsichtlich der internen und externen Zugriffsstrukturen für die Performance-Messungen verändert wurde. Das Hypertext-Schema setzt sich im wesentlichen aus den vier Seitentypen `Verbund`, `Mitarbeiter`, `Projekt` und `Standort` zusammen. Die Screen-Shots zu den verschiedenen Seitentypen sind in Anhang D abgebildet.

Da die Entwicklungsmaschine für Messungen nicht dediziert werden konnte, diente als Plattform ein PC Pentium 200 MHz MMX mit 64 MB RAM und dem Betriebssystem Windows NT. Der PC war vollkommen isoliert und nicht in ein Netzwerk eingebunden.

Im Rahmen von HDBM wurden die Meta-Daten zur Laufzeit ausgewertet. Die Implementierung der hier vorgestellten zweiten Alternative zur Auswertung der Meta-Daten im Vorfeld erfolgte nur prototypisch. Um dennoch Aussagen über die Performance und die Vor- und Nachteile beider alternativen Implementierungsvarianten machen zu können, wurde die Implementierung der zweiten Alternative simuliert, indem für das Beispiel der Forschungsverbünde von Hand zugehörige PL/SQL-Prozeduren geschrieben wurden. Dabei ist davon auszugehen, daß von Hand geschriebene PL/SQL-Prozeduren performanter als dynamisch generierte PL/SQL-Prozeduren sind. Zum einen können viele SQL-Anfragen zur Generierung eines Seitentyps zu einer SQL-Anfrage optimiert werden. Zum anderen sind keine Prozedur-Aufrufe für jeden Slice-Typ sondern nur für die Presentation Units nötig. Es ist deshalb zu erwarten, daß die tatsächliche Performance der mit Hilfe eines Programms generierten PL/SQL-Prozeduren nur im Idealfall den gemessenen Zeiten der von Hand geschriebenen PL/SQL-Prozeduren entspricht.

Um Aussagen über die Performance treffen zu können wurden Messungen hinsichtlich verschiedener Parameter durchgeführt:

- **Abhängigkeit von dem Seitentyp:** Die Performance wurde in Abhängigkeit von den vier abayfor-Seitentypen: `Standort`, `Mitarbeiter`, `Projekt` und `Verbund` bei durchschnittlichen Seiteninstanzen für beide Implementierungsvarianten gemessen.

- **Abhängigkeit von den Elementen:** Die Performance wurde in Abhängigkeit der innerhalb eines Seitentyps angezeigten Elemente für die erste Implementierungsvariante gemessen.

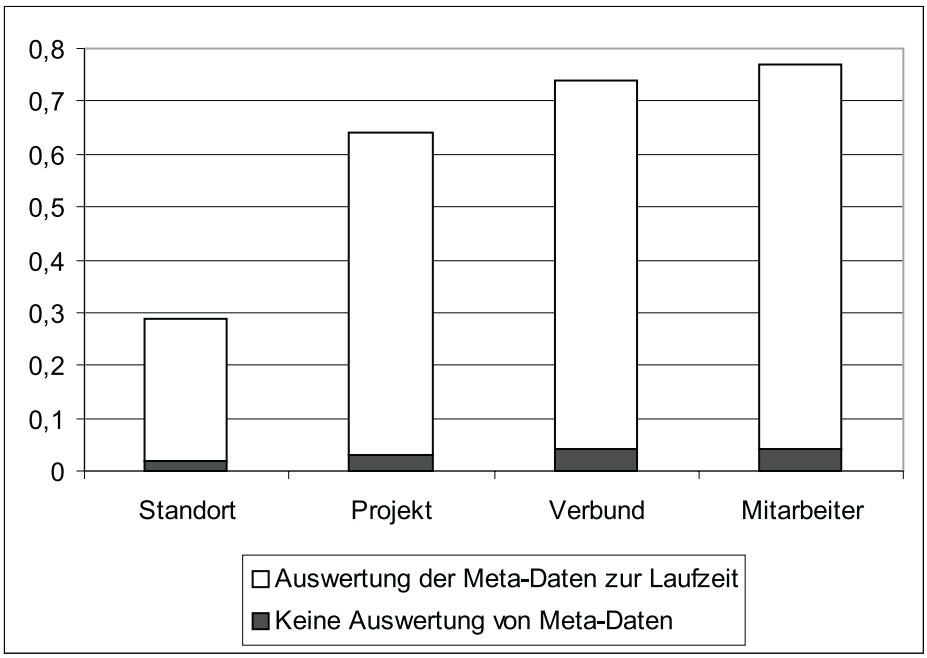
- **Abhängigkeit von der Anzahl der Listenelemente:** Die Performance eines Seitentyps wurde in Abhängigkeit von der Tupelanzahl einer internen Index-Liste gemessen. Die gleichen Messungen wurden für zwei verschiedene Datenbankgrößen durchgeführt.

- **Abhängigkeit von der Größe des Datenvolumens:** Die Performance wurde in Abhängigkeit von der Datenbankgröße für beide Implementierungsvarianten für die verschiedenen Seitentypen gemessen.

6.4.2 Meßergebnisse

Abhängigkeit von dem Seitentyp

Die Auswertung der Meta-Daten zur Laufzeit führt im Vergleich zur zweiten Implementierungsvariante zu längeren Antwortzeiten im Web-Browser. Der Unterschied zu im Vorfeld ausgewerteten Meta-Daten ist dabei abhängig von dem Aufbau des jeweiligen Seitentyps. Ist ein Seitentyp ausschließlich aus Attributen einer Relation aufgebaut, erfolgt der Seitenaufbau relativ schnell. Beim Aufbau von Hyperlinks, geschachtelten Slices oder Zugriffsstrukturen nimmt die Anfragezeit zu. Dies liegt zum einen daran, daß deutlich mehr Meta-Daten ausgewertet werden müssen sobald ein Join zwischen mehreren Relationen erforderlich ist. Zum anderen werden teilweise viele SQL-Anfragen separat durchgeführt, die zu einer SQL-Anfrage optimiert werden könnten. Um diesen Unterschied zu verdeutlichen, wurde die Antwortzeit in Sekunden beim Aufbau der vier Seitentypen `Standort`, `Mitarbeiter`, `Projekt` und `Verbund` für beide Implementierungsvarianten verglichen. Dabei wurden von jedem Seitentyp, die in Anhang D abgebildeten Instanzen der Seitentypen gewählt. Das Ergebnis wird durch Abbildung 6.6 veranschaulicht.



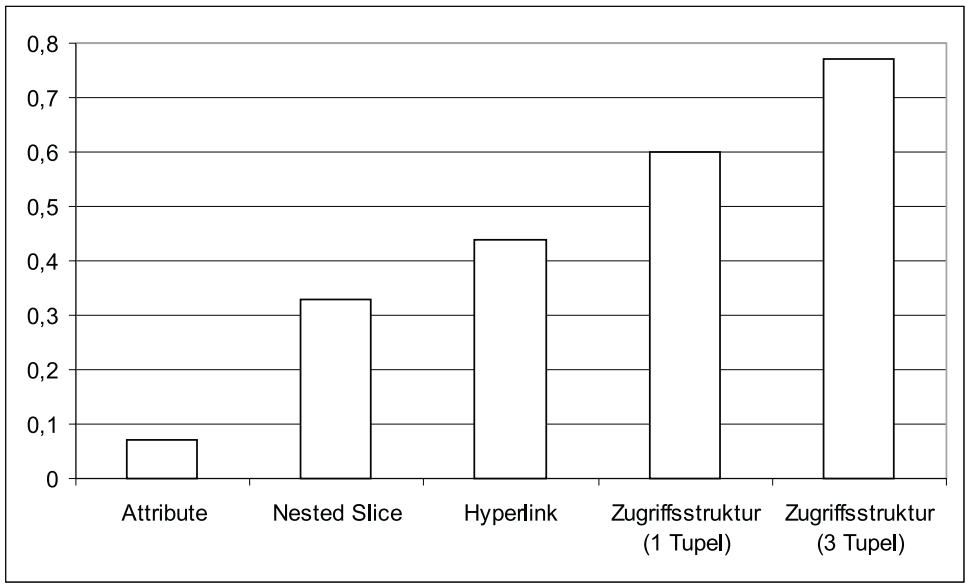
Antwortzeit in Sekunden	Standort	Projekt	Verbund	Mitarbeiter
Keine Auswertung der Meta-Daten	0,02	0,03	0,04	0,04
Auswertung der Meta-Daten zur Laufzeit	0,29	0,64	0,74	0,77

Abbildung 6.6: Performance unterschiedlicher Seitentypen im Vergleich

Da die Standort-HTML-Seite fast nur aus Attributen der STANDORT-Relation aufgebaut ist, erfolgt die Ausführung der zugehörigen PL/SQL-Prozeduren schnell. Der Aufbau der Mitarbeiter-Seite dauert dagegen erheblich länger. Dies ist durch den geschachtelten Slice der STANDORT-Relation, den Hyperlink auf einen Verbund und die Projektliste zu erklären.

Abhängigkeit von den Elementen eines Seitentyps

Da die verschiedenen Seitentypen unterschiedliche Elemente und Listen beinhalten, ist eine genauere Aussage über den Aufwand für die Generierung einzelner Elemente durch die Messung der Antwortzeit kaum möglich. Um die unterschiedlichen Antwortzeiten bei dem Aufbau von Attributen, Hyperlinks und Zugriffsstrukturen dennoch zeigen zu können, wurden verschiedene Versionen der Mitarbeiter-Seite für die gleiche Seiteninstanz gemessen. Die Mitarbeiter-Seite eignet sich hierfür sehr gut, da sie neben Attributen sowohl einen Hyperlink, einen geschachtelten Slice als auch eine Zugriffsstruktur beinhaltet. Es wurden fünf verschiedene Versionen des Mitarbeiter-Seitentyps unterschieden: Seitentyp, der ausschließlich sieben Attribute enthält (Name, Vorname ...), Seitentyp mit Attributen und einem nested Slice (Standort), Seitentyp mit einem weiteren Hyperlink auf den Verbund, Seitentyp mit einer weiteren Zugriffsstruktur (Projekte). Die Messergebnisse für die fünf verschiedenen Versionen des Mitarbeiter-HTML-Seitentyps werden durch Abbildung 6.7 veranschaulicht.

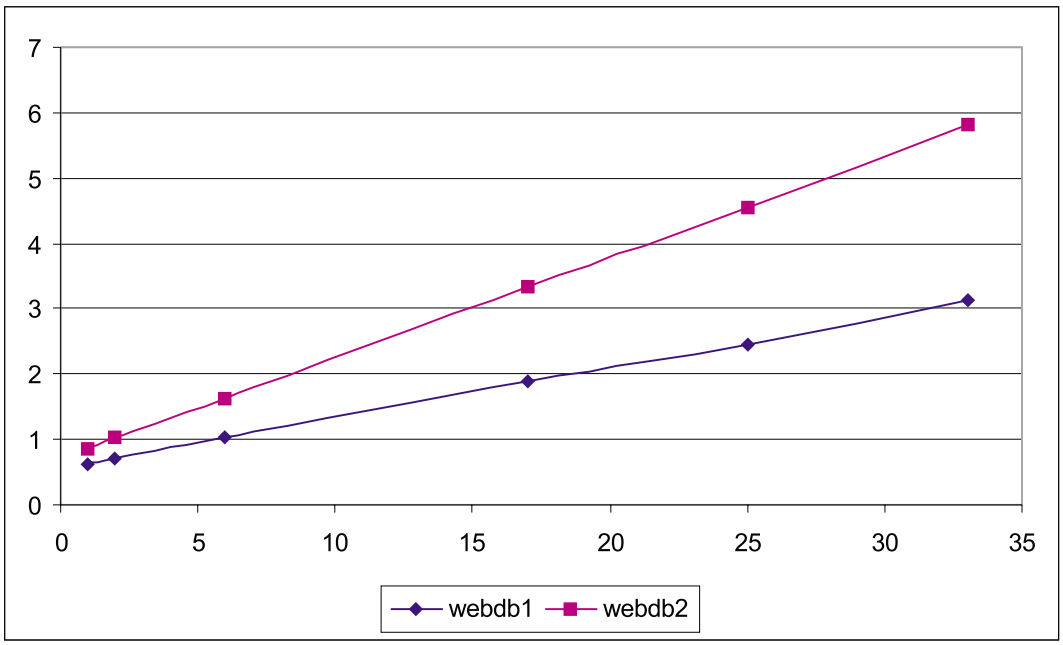


	Attribute	Nested Slice	Hyperlink	Zugriffsstruktur (ein Tupel)	Zugriffsstruktur (drei Tupel)
Antwortzeit in Sekunden	0,07	0,33	0,44	0,6	0,77

Abbildung 6.7: Performance des Mitarbeiter-Seitentyps mit verschiedenen Elementen

Abhängigkeit von der Anzahl der Listenelemente

Für jedes Listenelement müssen Meta-Daten ausgewertet werden. Dies begründet, daß die Anfragezeit bei der Generierung von längeren Listen wächst. Die Antwortzeit für den Aufbau einer HTML-Seite ist demnach abhängig davon, wieviele Listenelemente die Instanz beinhaltet. Abbildung 6.8 zeigt das lineare Wachstum der Kurve in Abhängigkeit von der Anzahl der Listenelemente. Die Steigung der Kurve hängt zum einen von der Komplexität der Listenelemente, zum anderen vom Datenvolumen der Datenbank ab. Sind die einzelnen Listenelemente einfach aufgebaut und beinhalten beispielsweise nur ein Attribut, dauert der Aufbau einer Liste weniger lange als bei komplexen, ineinander geschachtelten Elementen. Da für jedes Listenelement das zugehörige DB-Tupel durch SQL-Anfragen aus der Datenbank selektiert werden muß, ist die Steigung der Kurve ebenfalls vom Datenvolumen der Datenbank abhängig. Für die Messung wurde die Zugriffszeit auf verschiedene Instanzen der Verbund-Seite gemessen. Abhängig von dem jeweiligen Forschungsverbund wird auf der Verbund-Seite eine Liste zwischen einem und 33 Standorten generiert. Erhöht man den Datenbestand nimmt die Steigung der Kurve zu.



Antwortzeit in Sekunden	1 Tupel	2 Tupel	6 Tupel	17 Tupel	25 Tupel	33 Tupel
webdb1	0,62	0,7	1,03	1,9	2,46	3,12
webdb2	0,87	1,03	1,62	3,34	4,54	5,81

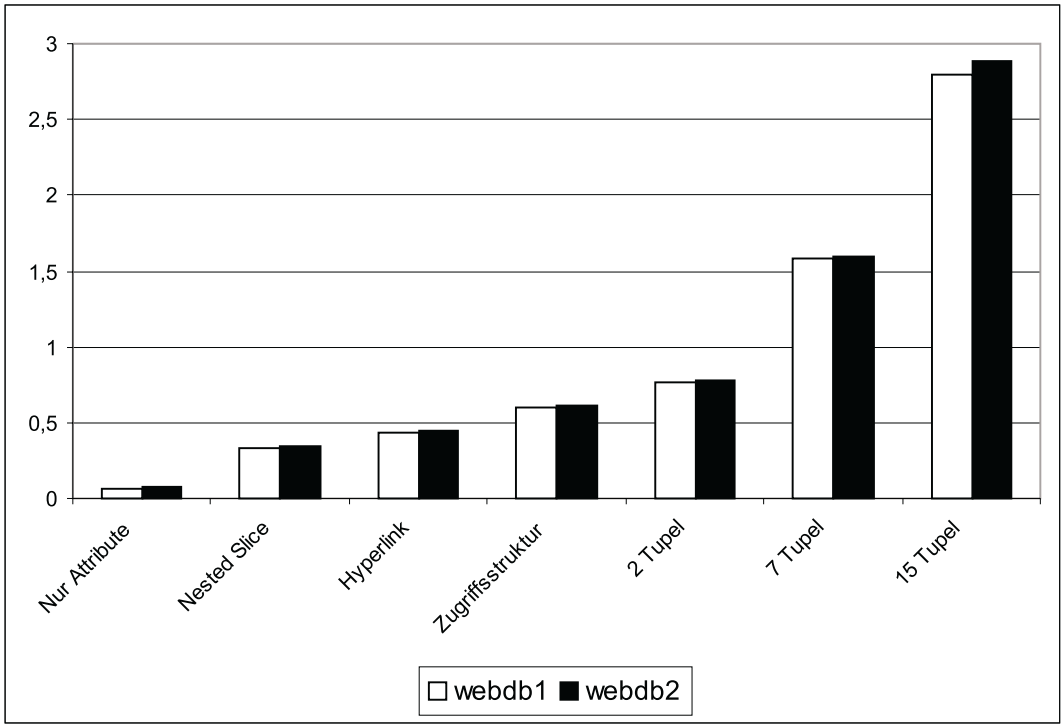
Abbildung 6.8: Performance beim Aufbau von Listen

Die Messung zeigt, daß für die Generierung langer Listen noch Optimierungsbedarf besteht. Bei der augenblicklichen Implementierung werden für jedes Tupel einer Liste die gleichen Abfragen an das Meta-Schema abgesetzt obwohl alle Listenelemente die gleiche

Struktur haben. Weiterhin erfolgt die Selektion der innerhalb eines Listenelements angezeigten Attribute durch einzelne SQL-Anfragen, die zu einer Anfrage optimiert werden könnten. Diese Implementierung ist u.a. durch die Einschränkungen von dynamischem SQL bei Oracle bedingt. Es ist beispielsweise nicht möglich Anzahl und Datentyp der innerhalb der Select-Klausel selektierten Attribute erst zur Laufzeit auszuwerten. Das DBMS_DESCRIBE-Paket (verfügbar ab Oracle 8.0) soll hierfür eine Lösung bieten.

Abhängigkeit von der Größe des Datenvolumens

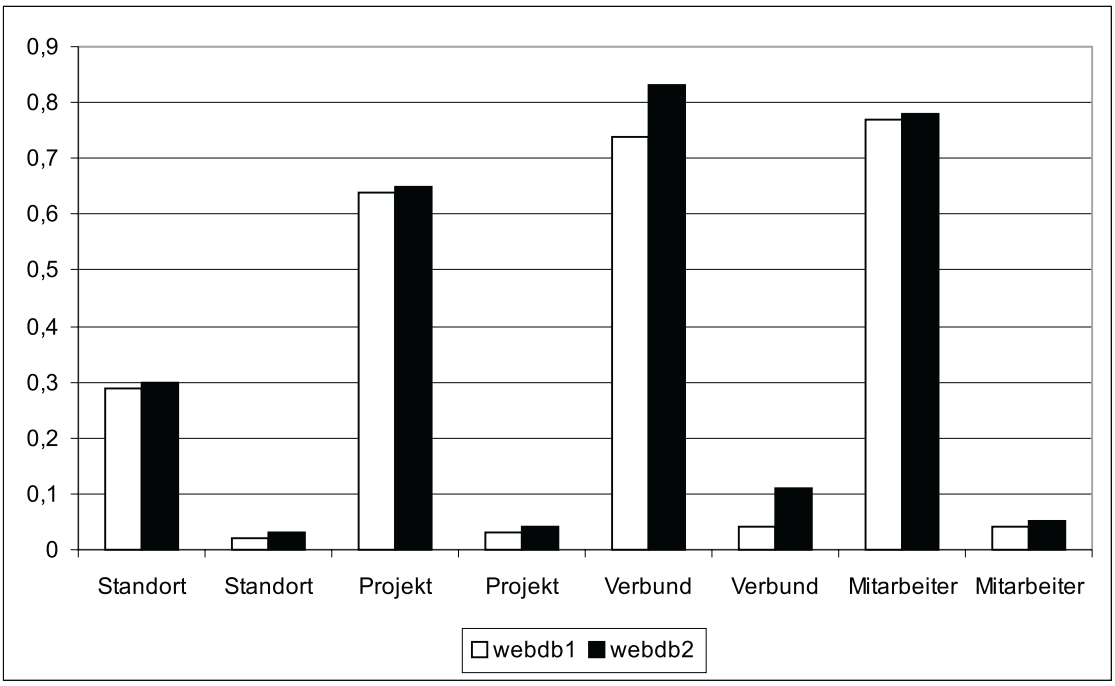
Meta-Daten beschreiben nur die Abbildung von Basis-Relationen auf eine Hypertext-Applikation. Die Auswertung der Meta-Daten erfolgt deshalb weitgehend unabhängig von dem Datenvolumen der Basis-Relationen. Eine Ausnahme bildet der Aufbau von langen Listen. Um dies zu zeigen wurden zwei unterschiedliche Datenbanken aufgebaut. Anschließend wurde der Aufbau von sieben verschiedenen Versionen des Mitarbeiter-Seitentyps in den beiden unterschiedlichen Datenbanken gemessen. Abbildung 6.9 zeigt, daß der Unterschied bei den Antwortzeiten in den zwei verschieden großen Datenbanken für alle Mitarbeiter-Versionen bei 0,01 Sekunden liegt. Erst bei der zunehmenden Tupel-Anzahl der Zugriffsstrukur wird der Unterschied bei der Anfragebearbeitung größer.



Antwortzeit in Sekunden	Nur Attribute	Nested Slice	Hyperlink	Zugriffs-struktur	2 Tupel	7 Tupel	15 Tupel
webdb1	0,07	0,33	0,44	0,6	0,77	1,58	2,79
webdb2	0,08	0,34	0,45	0,61	0,78	1,60	2,88

Abbildung 6.9: Performance des Mitarbeiter-Seitentyps für zwei Datenbankgrößen

Abbildung 6.10 zeigt einen Vergleich der Antwortzeiten bei der Generierung der verschiedenen Seitentypen mit und ohne Meta-Daten. Dabei wurden für jeden Seitentyp die Antwortzeiten in den beiden Datenbanken `webdb1` und `webdb2` gemessen. Als Instanzen der Seitentypen dienten die in Anhang D abgebildeten HTML-Seiten. Das Balkendiagramm zeigt, daß die Auswertung der Meta-Daten in einer größeren Datenbank ungefähr die gleiche Zeit beansprucht wie in einer kleineren Datenbank. Die Differenz der Antwortzeit für die kleine und die größere abayfor-Datenbank ist bei beiden Implementierungsvarianten nahezu gleich. Die aufgebauten Datenbankgrößen sind zwar repräsentativ für das abayfor-online-Projekt und andere Projekt- oder branchenspezifische Datenbanken im Web aber zu klein, um Aussagen für sehr große Datenbestände treffen zu können.



Anfragezeit in Sekunden	Standort	Projekt	Verbund	Mitarbeiter
Auswertung im Vorfeld, webdb1	0,02	0,03	0,04	0,04
Auswertung im Vorfeld, webdb2	0,03	0,04	0,11	0,05
Auswertung zur Laufzeit, webdb1	0,29	0,64	0,74	0,77
Auswertung zur Laufzeit, webdb2	0,3	0,65	0,83	0,78

Abbildung 6.10: Performance verschiedener Seitentypen für zwei Datenbankgrößen

6.5 Zusammenfassung

Die Umsetzung einer mächtigen Hypermedia-Modellierungsmethode auf ein Meta-Modell ermöglicht das schnelle Prototyping einer datenbankbasierten Web-Site und gewährleistet die einfache Wartbarkeit und Skalierbarkeit der Hypertext-Anwendung. Der Anwendungsentwickler kann sich vollständig auf Design-Aspekte konzentrieren und diese über komfortable Benutzeroberflächen oder mit Hilfe von CASE-Tools in der Datenbank definieren.

Im vergangenen Kapitel wurden zwei Implementierungsvarianten zur dynamischen Generierung datenbankbasierter Web-Sites auf der Basis eines Meta-Schemas vorgestellt.

Die Auswertung der Meta-Daten zur Laufzeit ermöglicht die hohe Konfigurierbarkeit der Hypertext-Oberfläche. Owner-Relation, Attribute und Layout einer HTML-Seite werden mit Hilfe von dynamischem SQL erst zur Laufzeit aus der Datenbank gelesen. Änderungen der Meta-Daten übertragen sich damit sofort auf die Hypertext-Oberfläche. Nachteil dieser Implementierungsvariante ist die etwas schlechtere Performance, die durch die Auswertung der Meta-Daten zur Laufzeit bedingt ist. Vor allem bei der dynamischen Generierung langer, komplexer Listen wird dies deutlich.

Eine andere Variante ist die Auswertung der Meta-Daten im Vorfeld. Die Meta-Daten werden von einem Progamm ausgewertet und zur Generierung von PL/SQL-Programmen verwendet. Jedem Slice-Typ wird eine Prozedur zugeordnet, die für die Generierung dieses Slice-Typs verantwortlich ist. Diese Implementierungsvariante wurde nur prototypisch implementiert, indem für jede Presentation Unit eine PL/SQL-Prozedur von Hand erzeugt wurde. Die Performance dieser Alternative ist besser da keine Meta-Daten zur Laufzeit ausgewertet werden müssen. Weiterhin können die so generierten PL/SQL-Prozeduren nachträglich von Hand verändert werden. Im Meta-Schema durchgeführte Änderungen übertragen sich allerdings erst auf die Hypertext-Oberfläche, wenn die zugehörige Prozedur aus dem DB-Schema neu erzeugt wird.

Die hohe Konfigurierbarkeit der Hypertext-Oberfläche und der relativ geringe Performance-Unterschied bei der Generierung durchschnittlicher HTML-Seitentypen spricht für die erste Implementierungsvariante. Die Antwortzeit bei der Generierung langer Listen muß allerdings noch optimiert werden.

Auf der Basis des HDBM-Meta-Schemas sind verschiedene Anwendungen möglich, die in den nächsten Kapiteln genauer beschrieben werden.

Kapitel 7

Anwendungsgebiete des Meta-Modells

Die logische Modellierung der Hypertext-Struktur innerhalb eines Meta-Schemas eröffnet vielfältige Anwendungsgebiete zur Nutzung und Erweiterung der Meta-Daten. Hierzu gehören erweiterte Anfragemöglichkeiten und inkrementelle Wartung materialisierter Web-Sites. Im folgenden Kapitel sollen Grundlagen und Lösungsansätze für die verschiedenen Anwendungsgebiete auf der Basis eines erweiterten HDBM-Meta-Modells beschrieben werden.

7.1 Ad-hoc-Anfragebearbeitung

Bei datenbankbasierten Informationssystemen im Web lassen sich verschiedene Techniken zur Formulierung von Datenbankanfragen mit Hilfe eines Web-Browsers unterscheiden: *Vordefinierte Anfragen*, *Navigieren durch die Datenbank* und *Ad-hoc-Anfragen*.

Im ersten Fall werden vordefinierte Datenbank-Anfragen durch HTML-Eingabeformulare abgesetzt. Dabei erfolgt die Anfragebearbeitung üblicherweise in drei Stufen: um eine Anfrage zu formulieren, trägt der Anwender zunächst in vordefinierte Eingabefelder eines HTML-Formulars die Selektionskriterien ein (siehe Abbildung 7.1). Hierfür werden keinerlei SQL-Kenntnisse benötigt. Nach Abschicken der Anfrage werden die Ergebnisse in Kurzform in einer Trefferliste angezeigt. Durch Auswahl des gewünschten Datenbankeintrags erfolgt die detaillierte Anzeige des Treffers, wobei die Ergebnispräsentation speziell auf die Struktur des Anfrageergebnisses abgestimmt ist.

Im zweiten Fall setzt der Anwender keine Datenbankanfrage durch ein HTML-Formular ab, sondern navigiert durch das Verfolgen von berechneten Hyperlinks, die intern auf Datenbankanfragen umgesetzt werden, durch die Datenbank. Auch bei dieser Variante sind die möglichen Navigationspfade und die damit verbundenen Anfragen an den Datenbestand vordefiniert.

Die letzte Variante ermöglicht die Formulierung beliebiger Datenbankanfragen an das

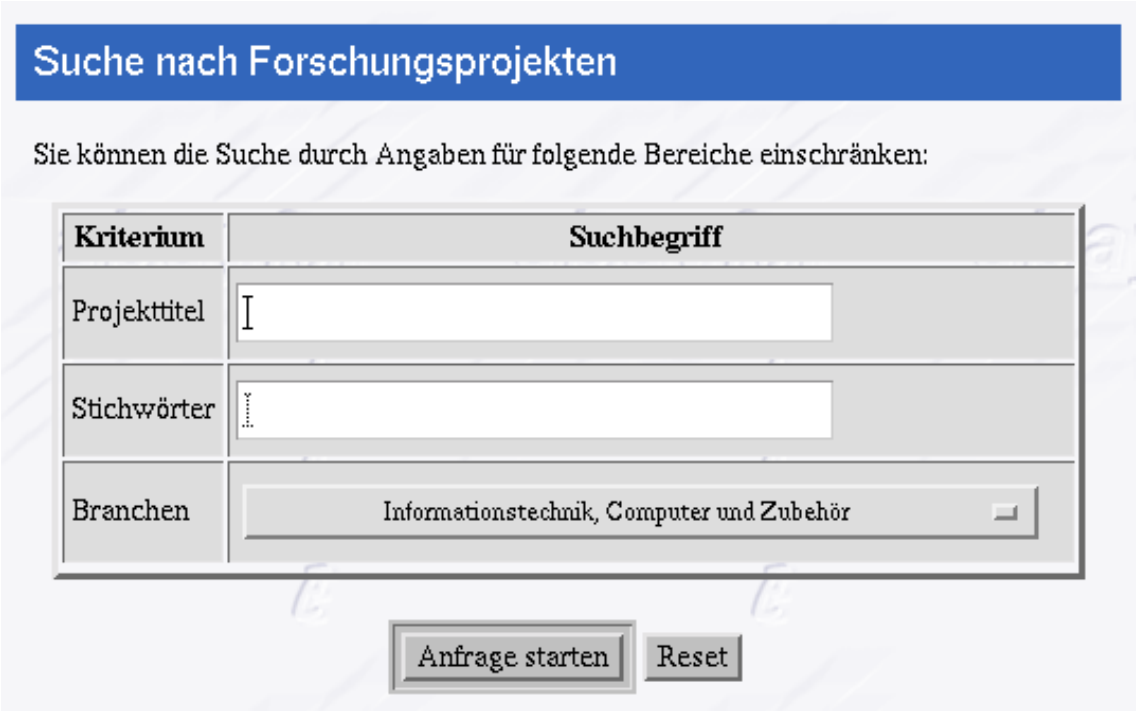
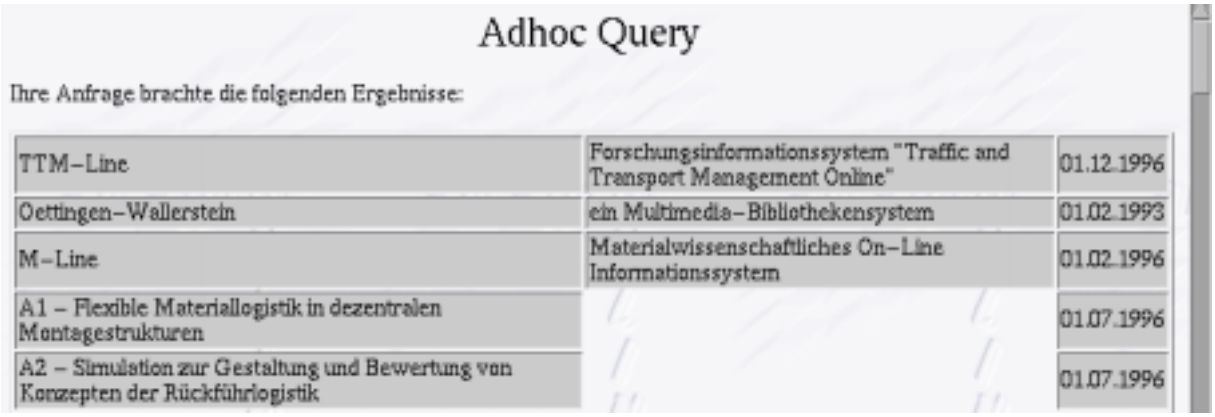


Abbildung 7.1: Vordefinierte Datenbankanfrage

Informationssystem (siehe Abbildung 7.2). Die Problematik bei *Ad-hoc-Anfragen* liegt darin, daß die dynamisch bestimmte Struktur des Anfrageergebnisses zur Laufzeit in eine sinnvolle Hypermedia-Darstellung umgewandelt werden muß. In herkömmlichen datenbankbasierten Web-Anwendungen erfolgt die Darstellung der Ergebnisse im relationalen Fall deshalb durch HTML-Tabellen (siehe Abbildung 7.3). Wünschenswert wäre die Präsentation der Anfrageergebnisse aber auf der Basis der bestehenden Strukturierung der jeweiligen Web-Site.

Abbildung 7.2: Ad-hoc-Datenbankanfrage

Für die Formulierung beliebiger Ad-hoc-Anfragen an eine datenbankbasierte Hypertext-Anwendung wurden in der Vergangenheit verschiedene Forschungsarbeiten durchgeführt. Im ARANEUS-Projekt wird auf der Basis des ADM-Datenmodells eine navigationale Algebra definiert [MMM98], mit deren Hilfe Anfragen über eine Web-Site abgesetzt und auf ein relationales Datenmodell abgebildet werden können. Ein anderer Ansatz ist die Verwendung einer erweiterten SQL-Anfragesyntax, in der die Hypertext-Struktur und



Adhoc Query

Ihre Anfrage brachte die folgenden Ergebnisse:

TTM–Line	Forschungsinformationssystem "Traffic and Transport Management Online"	01.12.1996
Oettingen–Wallerstein	ein Multimedia–Bibliothekensystem	01.02.1993
M–Line	Materialwissenschaftliches On–Line Informationssystem	01.02.1996
A1 – Flexible Materiallogistik in dezentralen Montagestrukturen		01.07.1996
A2 – Simulation zur Gestaltung und Bewertung von Konzepten der Rückführlogistik		01.07.1996

Abbildung 7.3: Ergebnis einer Ad-hoc-Datenbankanfrage

Darstellung des Anfrageergebnisses integriert werden. Ein Beispiel hierfür ist die auf der RMM basierende Anfragesprache von Geert-Jan Houben und Paul De Bra [HB98b], [HB99]:

```
select (H,K) asslice X, (I,J,K) asslice W,
links (head(R),X), (X,W)
from R;
```

Durch die Anfrage werden zwei neue Slices X und W definiert, wobei X die Attribute H und K der Relation R, und W die Attribute I, J und K der Relation R beinhaltet. Der Zugriff auf den neuen Slice X erfolgt über den Head Slice (siehe Kapitel 7.1.1) von R, der Zugriff auf den neuen Slice W erfolgt über den ebenfalls neu definierten Slice X.

In der HDBM soll SQL zur Formulierung von Ad-hoc-Anfragen eingesetzt werden. Die logische Modellierung der Hypertext-Anwendung in einem Meta-Modell gestattet die Auswertung der Meta-Daten zum Anfragezeitpunkt. Das folgende Kapitel beschreibt mögliche Wege zur Ad-hoc-Anfragebearbeitung mit Hilfe eines erweiterten Meta-Modells.

7.1.1 Erweiterung des Meta-Modells

Slices gruppieren zusammengehörige Attribute einer Relation und ermöglichen es, die Präsentation eines Datenbank-Tupels auf verschiedene HTML-Seiten zu untergliedern. Um zwischen Slices navigieren zu können, werden zwei Navigationsarten unterschieden: die *Inter-Record-Navigation* und *Intra-Record-Navigation*.

Im ersten Fall erfolgt die Navigation zwischen unterschiedlichen Datenbank-Tupeln mit Hilfe einer der in der RMM definierten Zugriffsstrukturen (Index, Guided Tour, Indexed Guided Tour). Ein Beispiel hierfür wäre die Navigation von der Projekt-Seite eines Forschungsverbundes auf die zugehörigen Mitarbeiter-Seiten.

Die Navigation zwischen Slices ein und desselben Datenbank-Tupels bezeichnet man als *Intra-Record-Navigation*. Die Darstellung eines Tupels wird in diesem Fall durch verschiedene Slices realisiert, die miteinander durch Hyperlinks verbunden sind. Ein Beispiel

hierfür wäre die Aufteilung der Mitarbeiter-Seite in eine Leitseite und weitere HTML-Seiten, beispielsweise mit dem Lebenslauf oder den Forschungsinteressen des Mitarbeiters. Werden einer Relation verschiedene Slice-Typen zugeordnet, die über Intra-Record-Navigation miteinander verknüpft sind, wird ein Einstiegspunkt benötigt.

Dieser Sachverhalt wird in der RMM durch das Konzept der *Head Slices* ausgedrückt. Ein Head Slice bildet demnach die Wurzel eines Baumes von der aus auf andere Slices, die der gleichen Owner-Relation angehören, navigiert werden kann. Das Konzept der Head Slices ist für die Umsetzung der Intra-Record-Navigation im Rahmen der Ad-hoc-Anfragebearbeitung relevant. Die im Meta-Modell zu einem Head Slice abgelegten Informationen, wie beispielsweise Hintergrund, Header, Footer, dienen gleichzeitig zur *Default-Darstellung* der Owner-Relation, falls kein anderer Slice-Typ spezifiziert wird. Die Modellierung der Head Slices erfolgt auf relationaler Ebene durch dir Ergänzung eines weiteren Attributs in der Meta-Relation SLICE (siehe Tabelle 7.1)

Attribute	*Beschreibung*
SNAME	Titel des Slice-Typs
P#	Referenz auf Präsentation
TABNAME	Referenz auf Owner-Relation
HEADSLICE	Ja/Nein
PRESUNIT	Eigenständiger Seitentyp (ja/nein)
TITEL	Titel der HTML-Seite

Tabelle 7.1: Relation SLICE

7.1.2 Ad-hoc-Anfragen

Die Modellierung der Hypertext-Struktur in einer Datenbank ermöglicht die Auswertung und Berücksichtigung dieser Informationen zum Anfragezeitpunkt.

Gehen wir von folgender SQL-Anfrage aus:

```
SELECT <ATTRIBUTE>
FROM <RELATIONEN>
WHERE <BEDINGUNG>
```

dann lassen sich auf der Basis der im Meta-Modell abgelegten Hypertext-Struktur nach [HB99] folgende drei Anfragetypen unterscheiden:

Ein Slice, eine Relation

Definition 7.1.2.1
Eine Datenbankanfrage heißt Ein Slice/Eine Relation-Anfrage wenn die Select-Klausel nur Attribute aus einer Relation R beinhaltet und alle selektierten Attribute die Teilmenge von einem Slice-Typ S mit der Owner-Relation R bilden.

Im einfachsten Fall gehören alle selektierten Attribute zu genau einem Slice S und einer Relation R. Die zu dem jeweiligen Slice-Typ im Meta-Modell abgelegten Informationen dienen zur Ergebnispräsentation. Das Ergebnis einer Ein Slice/Eine Relation-Anfrage wird durch folgende Regeln zusammengesetzt:

- Der Slice-Typ, der die selektierten Attribute als Teilmenge beinhaltet wird selektiert und zur Ergebnispräsentation verwendet. Falls dies mehrere Slice-Typen sind, wird der Head Slice bevorzugt behandelt.
- Liefert die Ad-hoc-Anfrage mehr als einen Treffer, erfolgt die Anzeige der Treffermenge mit Hilfe einer Hyperlink-Liste (Index). Das erste Attribut der Select-Klausel dient als Anker und verweist auf die Detail-Seiten. Bei genau einem Treffer, wird der Treffer direkt angezeigt.
- Als Ergebnis wird der selektierte Ausschnitt eines Slice-Typs auf der Detail-Seite angezeigt. Die Anzahl der Attribute und deren Reihenfolge ist durch die Select-Klausel spezifiziert. Web-Typ, Bezeichnung und Layout der Attribute werden dem Meta-Modell entnommen. Ist zu dem Slice-Typ kein Seitenlayout im Meta-Schema erfaßt, wird das Layout des Head-Slices verwendet.

Eine andere Variante ist die Anzeige aller Attribute und Referenzen des gewählten Slice-Typs anstelle nur der in der Select-Klausel spezifizierten Attribute.

Viele Slices, eine Relation

Beinhaltet die Select-Klausel Attribute von verschiedenen Slices, die jedoch alle der gleichen Owner-Relation zugeordnet sind, spricht man von einer *Viele Slices/Eine Relation-Anfrage*. Die betroffenen Slice-Typen sind über Intra-Record-Navigation miteinander verknüpft.

Definition 7.1.2.2
Eine Datenbankanfrage heißt Viele Slices/Eine Relation-Anfrage wenn die Select-Klausel Attribute aus verschiedenen Slices beinhaltet, die alle der gleichen Owner-Relation zugeordnet sind.

Im Unterschied zu einer Ein Slice/Eine Relation-Anfrage sind für die Ergebnispräsentation der Anfrage viele Slice-Typen betroffen, von denen einer (Head Slice) als Einstiegspunkt dienen muß. Der zentrale Einstiegspunkt gewährleistet die Navigation zu den anderen Slices. Das Ergebnis einer Viele Slices/Eine Relation-Anfrage wird durch folgende Regeln zusammengesetzt:

- Alle Slice-Typen, die die selektierten Attribute als Teilmenge beinhalten einschließlich des Head Slices werden selektiert. Aus diesen Slice-Typen wird der Head Slice der Owner-Relation ausgewählt, der als Einstiegspunkt dient.

- Liefert die Anfrage mehr als einen Treffer, erfolgt die Anzeige der Treffermenge mit Hilfe einer Hyperlink-Liste (Index). Bei genau einem Treffer, wird der Head Slice direkt angezeigt.

- Die Ergebnispräsentation erfolgt durch verschiedene Detail-Seiten wobei jeweils der selektierte Ausschnitt der betroffenen Slice-Typen angezeigt wird. Dabei muß gewährleistet werden, daß jeder Slice-Typ vom Head Slice direkt oder indirekt erreichbar ist. Die Anzahl der Attribute und deren Reihenfolge ist durch die Select-Klausel spezifiziert. Web-Typ, Bezeichnung und Layout der Attribute werden dem Meta-Schema entnommen.

Eine andere Variante ist die Anzeige aller in der select-Anfrage spezifizierten Attribute auf einer HTML-Seite. Der Zugriff auf eine Menge von Treffern erfolgt dabei wieder über eine Hyperlink-Liste wobei das erste Attribut als Ankerattribut verwendet wird. Als Layout wird das durch den Head Slice der Relation spezifizierte Layout der Owner-Relation zugrunde gelegt. Bezeichnung und Web-Typ der Attribute werden ebenfalls dem Meta-Schema entnommen. Existiert zu einem selektierten Attribut kein derartiger Eintrag, so wird standardmäßig der Attributname des relationalen Schemas als Bezeichnung angenommen. Die Reihenfolge der in der select-Anfrage spezifizierten Attribute wird beibehalten.

Viele Relationen

Beinhaltet die Select-Klausel verschiedene Attribute, die unterschiedlichen Relationen zugeordnet sind, wird dies als *Viele Relationen-Anfrage* bezeichnet. Da jedem Slice nur genau eine Owner-Relation zugeordnet werden darf, gehören die Attribute zu verschiedenen Slice-Typen.

Definition 7.1.2.3
Eine Datenbankanfrage heißt Viele Relationen-Anfrage wenn die From-Klausel verschiedene Relationen enthält und die Select-Klausel Attribute aus verschiedenen Relationen beinhaltet.

Normalerweise spezifiziert der Anwender in der From-Klausel einer SQL-Anfrage verschiedene Relationen, die durch Joins in der Where-Klausel miteinander in Beziehung gesetzt werden. Daneben gibt es auch Viele-Relationen-Anfragen ohne Where-Klausel. Die Ergebnispräsentation sollte deshalb möglichst universal sein und garantieren, daß alle selektierten Slice-Typen von einer Einstiegsseite aus erreicht werden können. Das Ergebnis einer Viele-Relationen-Anfrage läßt sich entsprechend durch folgende Regeln aufbauen:

- Zu jeder in der From-Klausel aufgeführten Relation, von der Attribute selektiert werden, wird ein geeigneter HTML-Seitentyp aufgebaut, der die in der Select-Klausel spezifizierten Attribute beinhaltet. Dabei werden die Regeln für die Ein Slice/Eine Relation- und Viele Slices/Eine Relation-Anfragen verwendet.

- Die Inter-Record-Navigation zwischen den Seitentypen erfolgt auf der Basis der in der From-Klausel spezifizierten Relationen. Dabei erfolgt der Einstieg in die Ergebnispräsentation über den HTML-Seitentyp der ersten in der From-Klausel genannten Relation. Von hier wird auf den Head Slice der nächsten Relation mit Hilfe eines Index zugegriffen.

Folgendes Beispiel soll dies verdeutlichen:

```
select   STANDORT.SNAME, STANDORT.STRASSE, STANDORT.PLZ,
         MITARBEITER.NNAME, MITARBEITER.TITEL, MITARBEITER.EMAIL
from     STANDORT, MITARBEITER
where    STANDORT.SNAME=MITARBEITER.SNAME
```

Eine SQL-Anfrage verbindet die Instanzen zweier Tabellen über Joins zwischen den Tabellen. Der Zugriff auf die Instanzen anderer Tabellen erfolgt über einen Index. Zuerst wird der Slice-Typ der ersten Relation STANDORT aufgebaut. Auf der Standort-Seite wird ein interner Index auf den Slice von MITARBEITER ergänzt. Die Abbildung 7.4 soll diesen Vorgang verdeutlichen.

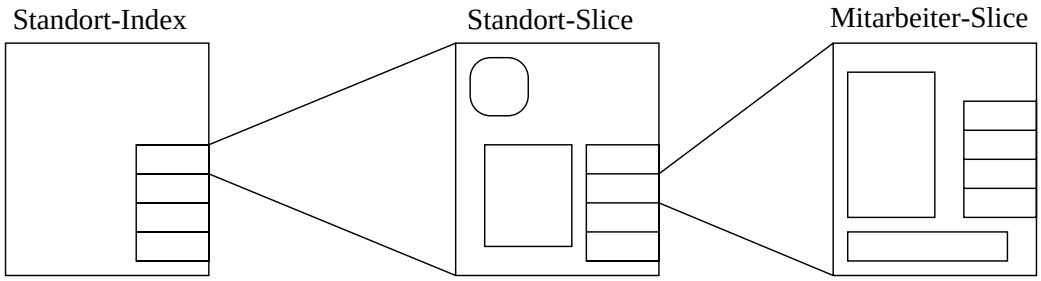


Abbildung 7.4: Generierung von Anfrageergebnissen

Durch die vorgestellte Auswertung von Ad-hoc-Anfragen lassen sich für einfache SQL-Anfragen sinnvolle Ergebnispräsentaionen in HTML generieren. Um die verschiedenen Darstellungsvarianten gezielt durch den Anwender bestimmen zu lassen, ist jedoch eine Erweiterung von SQL zur Spezifikation von Struktur und Navigation des Anfrageergebnisses notwendig.

7.2 Materialisierte HTML-Seiten

Das folgende Kapitel beschreibt Grundlagen und Lösungsansätze zur inkrementellen Wartung von materialisierten HTML-Seiten auf der Basis eines erweiterten Meta-Modells.

Bei datenbankbasierten Web-Anwendungen werden die Inhalte der Datenbank ausgespoolt, in HTML-Seiten eingebettet und in einem Web-Browser angezeigt. Die Kopplung der Datenbank an das Web ist dabei lose oder fest. Man spricht in diesem Zusammenhang auch von einer offline-Seitengenerierung oder online-Seitengenerierung:

- **offline-Seitengenerierung:** Das Datenbanksystem dient nur zur Datenverwaltung und überspielt die Daten bzw. die Updates in regelmäßigen Abständen oder iterativ von der Datenbank in flache HTML-Dateien. Dabei können entweder die gesamten Datenbankinhalte oder nur bestimmte Teile ausgespoolt werden.
- **online-Seitengenerierung:** Das WWW dient als Benutzerschnittstelle zu einem Datenbanksystem. Die gewählte Middleware extrahiert und analysiert die Benutzereingaben, greift auf die gewünschte Datenbank zu und erzeugt dynamisch die HTML-Ergebnisseiten.

Beide Alternativen haben verschiedene Vor -und Nachteile, die durch die Tabelle 7.2 verdeutlicht werden.

	offline	online
Wartung	−	+
Retrieval	−	+
Performance	+	−
URL/Bookmark	+	−
Plattformunabhängig	+	−

Tabelle 7.2: Vor- und Nachteile materialisierter Hypertext-Views

Die offline-Seitengenerierung eignet sich vor allem für die Erzeugung großer Web-Sites, deren Daten nicht häufig geändert werden. Dies ergibt sich aus der statischen Ablage des Datenbankinhaltes in HTML-Dateien. Der Vorteil dieser Variante besteht zum einen in der erhöhten Performance bei den Seitenzugriffen, da das Laden der HTML-Seiten direkt über den Web-Server ohne Datenbankzugriff erfolgt. Zum anderen sind offline-generierte HTML-Seiten im Gegensatz zu dynamisch generierten HTML-Seiten plattformunabhängig. Damit ist die lokale Ablage der HTML-Seiten auf jedem beliebigen Web-Server möglich.

Die Nachteile offline-generierter HTML-Seiten liegen in der schweren Wartbarkeit und den mangelnden Retrievalmöglichkeiten. Hierin liegen die Vorteile online generierter HTML-Seiten. Änderungen auf dem Datenbestand werden sofort sichtbar, da alle HTML-Seiten mit Hilfe von SQL-Anfragen an die Datenbank zur Laufzeit erzeugt werden. Beide

Alternativen finden in der Praxis für unterschiedliche Web-Applikationen Ihre Anwendung und sollten deshalb von der Anbindungssoftware geeignet unterstützt werden.

Neben der Generierung dynamischer HTML-Seiten aus der Datenbank (siehe Kapitel 6.1), ermöglicht das vorgestellte Meta-Modell auch die offline-Generierung materialisierter HTML-Seiten, die innerhalb oder außerhalb der Datenbank physisch abgelegt werden. Eine Problematik bei der Materialisierung von Hypertext-Views liegt in der schweren Wartung und den hierdurch bedingten Konsistenzverletzungen zwischen Basis-Relationen und Views. Im folgenden wird ein Ansatz zur Generierung und Pflege materialisierter Hypertext-Views mit Hilfe des in der HDBM entwickelten Meta-Modells vorgestellt.

7.2.1 Erweiterung des Meta-Modells

Für die Generierung materialisierter HTML-Seiten muß das Meta-Modell bezüglich der zwei folgenden Punkte erweitert werden.

Dies betrifft zum einen die Adressierung von HTML-Dokumenten im Internet mit Hilfe von Uniform Resource Locators. URLs beinhalten die vollständige Adresse eines HTML-Dokuments einschließlich Hostname (oder DNS-Alias), Pfad und Dateiname. Sie dienen zur eindeutigen Identifizierung von Datenquellen im Internet. Die tatsächliche Ablage der materialisierten HTML-Seiten erfolgt im *Document Root* des Web-Servers.

Im Falle des abayfor-Projektes wäre beispielsweise die *Basis Domain*:

```
abayfor.de
```

Das tatsächliche Document Root des Web-Servers lautet:

```
/home/sunabayfor/proj/abayfor/.html-data/
```

Der hierzu relative Verzeichnispfad eines HTML-Dokuments wäre beispielsweise

```
/forum/geschäftsstelle.html
```

Diese Informationen werden zur Generierung der URLs benötigt und im Meta-Modell abgelegt. Sowohl die Basis Domain als auch das Document Root des Web-Servers werden durch eine separate Relation DOMAIN modelliert. Dabei wird vereinfacht von dem Standard-Port für HTTP-Server 80 ausgegangen. Daneben wird der relative Verzeichnispfad jeder Presentation Unit in der Relation SLICE abgelegt. Zu jeder Domain können verschiedene Slice-Typen definiert werden.

Zur Wartung der materialisierten HTML-Seiten ist zum anderen die Unterscheidung von *Presentation Units* und *Component Slices* wichtig. Jede Presentation Unit entspricht einer HTML-Seite, die rekursiv aus verschiedenen Slice-Typen, den sogenannten Component Slices aufgebaut ist. Beispielsweise wäre der Mitarbeiter-Slice `Mitarb.Info` eine Presentation Unit. Die Anzeige des Standort-Namens innerhalb der Mitarbeiter-Seite wird im Gegensatz hierzu durch einen Component Slice realisiert.

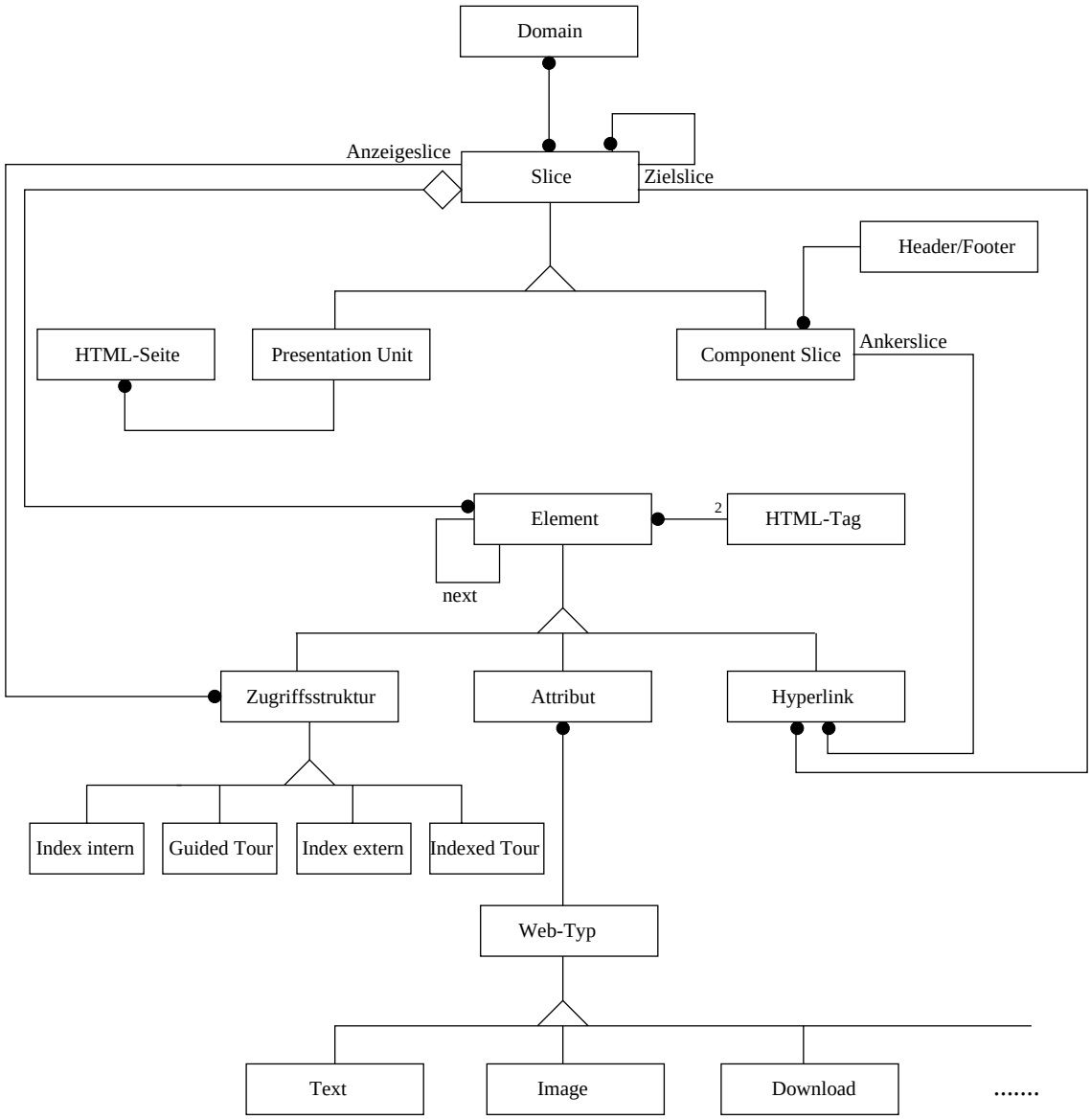


Abbildung 7.5: Erweitertes Meta-Modell

Abbildung 7.5 zeigt das erweiterte Meta-Modell. Die Klasse SLICE bildet die SUPERKLASSE zweier SUBKLASSEN: PRESENTATION UNIT und COMPONENT SLICE. Der Zielslice eines Hyperlinks ist eine Presentation Unit während der Ankerslice ein Component Slice ist. Jede externe Zugriffsstruktur verweist auf eine Presentation Unit. Interne Zugriffsstrukturen verweisen nur auf Component Slices.

Auf relationaler Ebene erfolgt die Ergänzung bzw. die Erweiterung folgender Relationen:

Relation SLICE

Attribute	*Beschreibung*
SNAME	Titel des Slice-Typs
P#	Referenz auf Präsentation
REL_PFAD	Relativer Verzeichnispfad
PRESUNIT	Ja/Nein
TABNAME	Referenz auf Owner-Relation
HEADSLICE	Ja/Nein
TITEL	Titel der HTML-Seite

Relation DOMAIN

Attribute	*Beschreibung*
DOMAIN	Basis-Domain des Web-Servers
DOC_ROOT	Document Root des Web-Servers

Relation SLICE_DOM

Attribute	*Beschreibung*
DOMAIN	Referenz auf Basis-Domain
SNAME	Referenz auf Slice-Typ

In den folgenden Kapiteln wird die Generierung der HTML-Seiten und ein Algorithmus für die inkrementelle Wartung vorgestellt.

7.2.2 Generierung und Ablage der Hypertext-Views

Die offline-Generierung von HTML-Seiten aus den Inhalten einer Datenbank unterscheidet sich von der online-Generierung vorwiegend in der Generierung der URLs und der Ablage der HTML-Dateien. Beide Punkte sollen in diesem Kapitel näher besprochen werden.

Bei der Materialisierung von Hypertext-Views gibt es verschiedene Möglichkeiten für die Ablage der HTML-Seiten (siehe Abbildung 7.6):

- Ablage im File-System. Die materialisierten HTML-Seiten werden innerhalb eines Verzeichnisbaumes im Dateisystem abgelegt. Zu jeder Änderung in der Datenbank müssen die zugehörigen HTML-Seiten neu generiert und überschrieben werden.
- Ablage in der Datenbank als BLOB. Die HTML-Seiten werden innerhalb des Datenbanksystems als BLOB abgelegt und gewartet (siehe in [DHW97]). Der Zugriff auf die HTML-Seite erfordert einen Datenbankzugriff.

Im folgenden soll von der ersten Variante ausgegangen werden. Die Generierung der URLs erfolgt mit Hilfe der im Meta-Modell zu jedem Slice-Typ abgelegten Meta-Daten (siehe Abbildung 7.7). Ein URL hat dabei folgende Form:

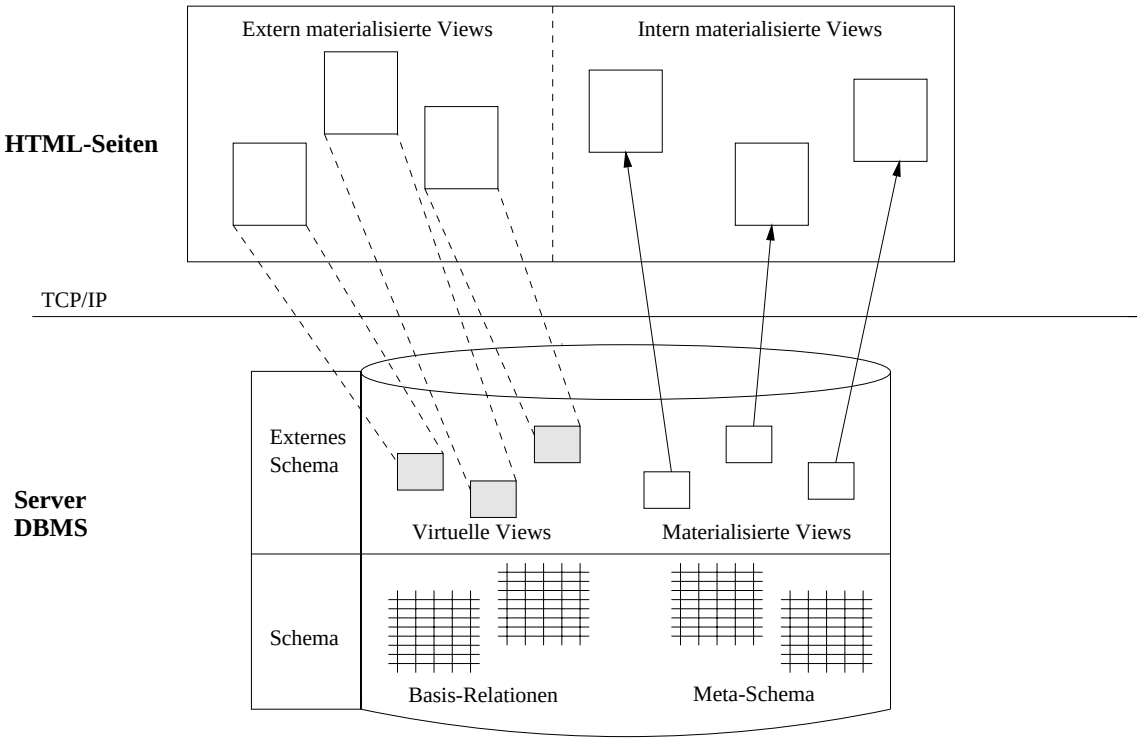


Abbildung 7.6: Unterschiedliche Varianten zur Materialisierung von HTML-Seiten

`http://hostname/relativer_Pfad/slice-name_schlüssel.html`

Wobei:

- `hostname`: bezeichnet den in der Relation DOMAIN abgelegten Domain-Namen des WWW-Servers.
- `relativer_Pfad`: bezeichnet den relativen Vezeichnispfad zum Document-Root des Web-Servers.
- `slice-name`: bezeichnet den Slice-Namen des jeweiligen Slice-Typs.
- `schlüssel.html`: bezeichnet den Wert des Primärschlüssels einer Instanz der Owner-Relation.

Die Generierung der HTML-Seiten erfolgt ähnlich der Generierung dynamischer HTML-Seiten. Der Unterschied liegt in der Ablage der HTML-Seiten im File-System, dem anderen Aufbau der URLs und in der Beschränkung auf ausgewählte Presentation Units. Jeder DOMAIN sind durch das Meta-Modell verschiedene Presentation Units zugeordnet. Nur diese werden bei der Generierung berücksichtigt. Damit ist es möglich einen Teilausschnitt einer Web-Site auszuspoolen. Im Meta-Schema definierte externe Zugriffsstrukturen (Index Extern, Guided Tour, Indexed Guided Tour) werden bei der Generierung nur dann berücksichtigt, wenn der Zielslice ebenfalls dieser Domain zugeordnet ist. Für die Generierung von HTML-Seiten wird eine neue Funktion `gen_slice` eingeführt, die als Eingabe eine Basis-Domain erhält und als Ausgabe eine Menge von HTML-Seiten generiert.

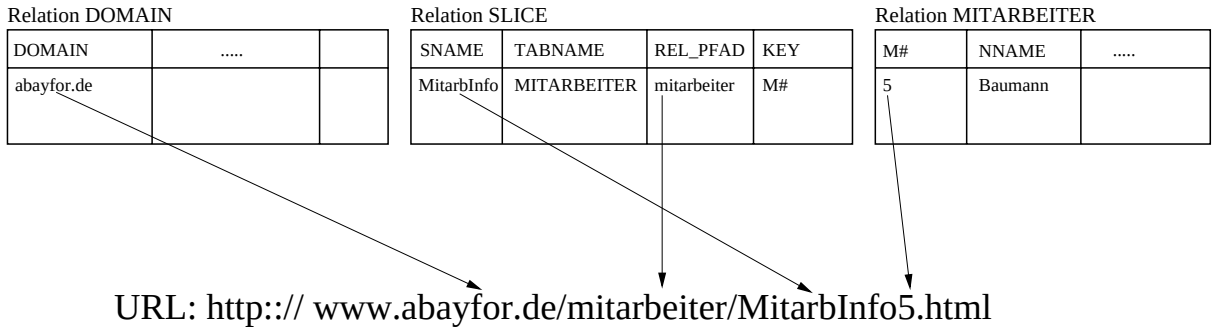


Abbildung 7.7: Beispiel für die Generierung einer URL

Ein interessanter Ansatz bei der offline-Generierung von HTML-Seiten aus einer Datenbank ist die Berücksichtigung von Meta-Daten in der HTML-Seite. Hierdurch wird die reiche innere Struktur der Daten mit Hilfe von Meta-Tags in das Dokument integriert. Dies ist beispielsweise mit XML [MAM+98] möglich.

7.2.3 Inkrementelle Wartung von Hypertext-Views

Eine Problematik bei materialisierten Hypertext-Views liegt in der erschwerten Wartung der HTML-Seiten. Jeder HTML-Seitentyp entspricht dabei in der HDBM einer Presentation Unit. Ein HTML-Seitentyp kann aus den Attributen verschiedener Relationen aufgebaut sein (siehe Abbildung 7.8). Der Seitentyp ist von verschiedenen Relationen abhängig. Ändert man einzelne Attribute eines Tupels einer Basis-Relation, so müssen die zugehörigen materialisierten HTML-Seiten ebenfalls aktualisiert werden. Dabei können entweder alle Instanzen der von dieser Relation abhängigen HTML-Seitentypen geändert werden oder nur inkrementell die betroffenen HTML-Seiten neu generiert werden. Im letzten Fall spricht man von *inkrementeller Wartung*.

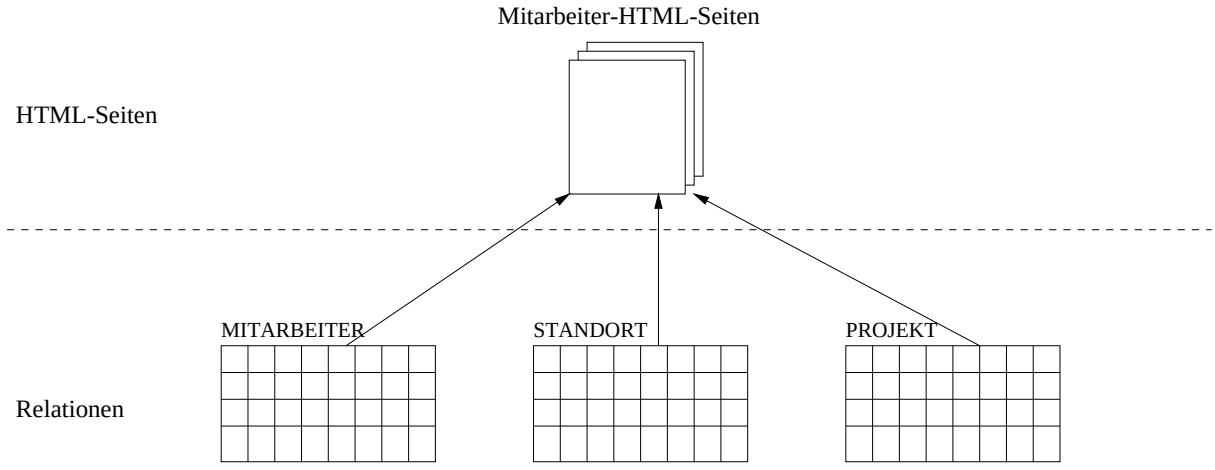


Abbildung 7.8: Abhängigkeit zwischen Relationen und HTML-Seiten

Ändert man beispielsweise die Adresse eines Mitarbeiters in der Relation Mitarbeiter, so müssen nicht alle HTML-Mitarbeiterseiten neu generiert werden, sondern nur alle

HTML-Seiten, die Informationen zu diesem Mitarbeiter beinhalten.

Bei der inkrementellen Wartung wird die sofortige (*immediate maintenance*) von der abgeleiteten Wartung (*deferred maintenance*) unterschieden. Im ersten Fall erfolgt die Wartung innerhalb der gleichen Transaktion, in der eine Änderung auf einer Basis-Relation durchgeführt wird. Diese Art der Wartung ist aufwendiger, garantiert aber die Serialisierbarkeit und Konsistenz zwischen den Basis-Daten und den abgeleiteten Views. Die Wartung außerhalb der Transaktion bezeichnet man als deferred maintenance.

Im folgenden soll auf der Basis des Meta-Modells die inkrementelle Wartung von materialisierten Hypertext-Views vorgestellt werden. Der Wartungsalgorithmus verwendet hierfür ausschließlich das erweiterte Meta-Modell und die Basis-Relationen. Die Änderungen werden zunächst mit Hilfe von Delta-Mengen erfaßt. Die Wartung erfolgt außerhalb der auslösenden Transaktion.

Änderungen auf der Datenbank haben verschiedene Auswirkungen auf die daraus generierten HTML-Seiten. Zunächst sollen die zwei Datenbank-Operationen *Löschen* und *Einfügen* auf den Basis-Relationen und deren Auswirkung auf die materialisierten Hypertext-Views betrachtet werden.

Löschen

Eine Instanz einer Basis-Relation wird gelöscht:

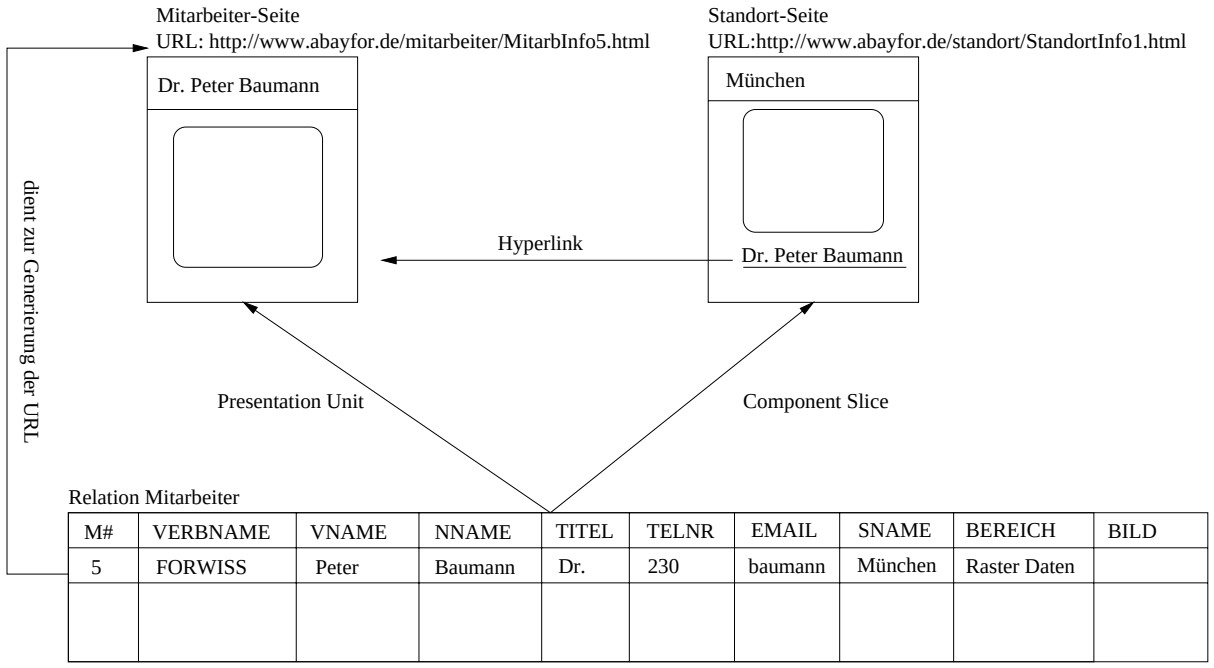
auf HTML-Seitenebene werden folgende zwei Fälle unterschieden (siehe Abbildung 7.9):

- *Fall 1: Presentation Unit:* Die Basis-Relation ist Owner-Relation einer Presentation Unit. Der Primärschlüssel des gelöschten Tupels ist direkt am Aufbau einer URL beteiligt. Die aus dem Primärschlüssel des Tupels generierten URLs (HTML-Seiten) müssen gelöscht werden.

- *Fall 2: Component Slice:* Die Relation ist Owner-Relation eines Component Slices. Es existieren HTML-Seiten, die Attribute des gelöschten Tupels (beispielsweise innerhalb eines Hyperlinks) beinhalten; der URL der HTML-Seite wird aber aus dem Primärschlüssel einer anderen Owner-Relation generiert. Alle HTML-Seiten, die den Component Slice beinhalten müssen neu generiert werden.

Abbildung 7.9 soll dies verdeutlichen. Aus der Relation Mitarbeiter werden sowohl die Mitarbeiter-HTML-Seiten als auch die Standort-HTML-Seiten generiert. Der URL der Mitarbeiter-HTML-Seite wird mit Hilfe des Primärschlüssels der Owner-Relation Mitarbeiter generiert. Wird eine Instanz der Relation Mitarbeiter in der Datenbank gelöscht, muß demnach auch die zugehörige Mitarbeiter-HTML-Seite gelöscht werden.

Im Gegensatz hierzu beinhaltet die Standort-Seite nur eine Komponente, die aus der Relation Mitarbeiter generiert wird. Diese Seite muß deshalb nur neu generiert werden, wobei der Eintrag Dr. Baumann von der Seite entfernt wird.



M#	VERBNAME	VNAME	NNAME	TITEL	TELNR	EMAIL	SNAME	BEREICH	BILD
5	FORWISS	Peter	Baumann	Dr.	230	baumann	München	Raster Daten	

Abbildung 7.9: Löschen von Instanzen

Einfügen

Ein neues Tupel wird in eine Relation eingefügt:

beim Einfügen eines Tupels in eine Basis-Relation lassen sich folgende zwei Fälle unterscheiden:

- *Fall 1: Presentation Unit:* Alle HTML-Seiten, deren URL aus dem Schlüssel der eingefügten Instanz aufgebaut wird, müssen neu generiert werden.
- *Fall 2: Component Slice:* Alle HTML-Seiten, die eine Komponente der Owner-Relation beinhalten müssen neu generiert werden.

Auf HTML-Seitenebene müssen deshalb alle Seitentypen, die Tupel der Relation beinhalten oder selbst aus Tupeln der Relation aufgebaut sind neu generiert werden. Da Anker und Ziel eines Hyperlinks laut Definition die gleiche Owner-Relation haben, werden sowohl Quell- als auch Ziel-Seitentyp neu aufgebaut.

Algorithmus zur inkrementellen Wartung

Für die inkrementelle Wartung von HTML-Seiten auf der Basis eines Meta-Modells werden folgende Komponenten benötigt:

- Geeignetes logging-Verfahren zur Protokollierung von Änderungen
- Geeignete Datenstruktur zur Verwaltung essentieller Informationen über Abhängigkeiten zwischen Basis-Relationen und Seitentypen.

- Algorithmus zur inkrementellen Wartung (Seitengenerierung)

Im folgenden sollen Lösungsansätze zu den einzelnen Punkten vorgestellt werden.

Die Δ-Mengen-Technik ist ein logging-Verfahren, das Mengendifferenzen verwaltet. Sie wurde in [Hum92] zur effizienten Integritätssicherung in deduktiven Datenbanksystemen durch Logikprogrammtransformation entwickelt, in [Höf96] als Basis zur Realisierung der Schema-Evolution in objekt-orientierten Datenbanksystemen und in [Kem96] zur Umsetzung langer Transaktionen eingesetzt. Δ-Mengen protokollieren die Veränderung einer Menge $\mathcal{M}$ von Elementen E, die sich aus Operationen $O_{\mathcal{M}}$ ergeben. Für Δ-Mengen gilt das IDS-Modell [Kem96]. IDS steht für die Operationen *insert-delete-select* auf die alle Operationen abgebildet werden. Die durch Delta-Mengen erfaßten Änderungen ergeben sich deshalb aus einer Folge von Löschungen d und Einfügungen i von Elementen:

$$[O_M](M) = O_n(E) \circ \cdots \circ O_1(E) \text{ und } O_i \in \{i, d\}$$

Dabei lassen sich disjunkte Δ^+-Mengen und Δ^--Mengen unterscheiden. Δ^+-Mengen sammeln alle durch Einfügungen neu zur Menge M dazukommenden Elemente E auf. Δ^--Mengen verwalten dagegen alle Löschungen und enthalten diejenigen Elemente, die aus M entfernt worden sind. Im vorliegenden Ansatz werden Delta-Mengen verwendet, um Änderungen zu protokollieren, die seit der letzten inkrementellen Wartung der HTML-Seiten auf Basis-Relationen durchgeführt wurden. In einer relationalen Datenbank können Δ-Mengen durch separate Relationen modelliert werden, die jeweils die gelöschten oder eingefügten Tupel der zugehörigen Basis-Relation beinhalten. Dabei wird davon ausgegangen, daß die Δ^--Menge und Δ^+-Menge disjunkt sind. Wurde seit der letzten inkrementellen Wartung der HTML-Seiten beispielsweise ein Tupel eingefügt und anschließend wieder gelöscht, so wird diese Instanz aus beiden Δ-Mengen entfernt.

Zur Wartung materialisierter Views dient häufig ein View-Dependency-Graph, der die Abhängigkeit einer View von den zugehörigen Basis-Relationen dokumentiert. In HDBM ist diese Abhängigkeit durch das Meta-Schema gegeben.

Auf der Basis des Meta-Schemas und den Delta-Mengen ist nun die Wartung materialisierter HTML-Seitentypen möglich. In Anlehnung an [Sin98] werden die folgenden drei Schritte unterschieden: (1) Löschen von HTML-Seiten, (2) Generierung von HTML-Seiten, in die Werte eingefügt werden und (3) Generierung von HTML-Seiten aus denen Werte gelöscht werden. Um die Konsistenz der Web-Site zu gewährleisten, müssen die folgenden Schritte für alle Seitenschematas durchgeführt werden.

Schritt 1: Löschen von URLs

Aus einer Basis-Relation wurde ein Tupel gelöscht. Die Basis-Relation ist Owner-Relation eines Seitentyps. Daraus folgt, daß der Primärschlüssel des gelöschten Tupels am Aufbau eines URL beteiligt war. Die aus dem Primärschlüssel des Tupels generierte URL (HTML-Seite) muß gelöscht werden. Hierfür wird eine neue Funktion `URL_delete` eingeführt:

Funktion:	`URL_delete`
INPUT:	Name einer Presentation Unit (Slice-Typ)
	Δ^--Menge der zugehörigen Owner-Relation
OUTPUT:	Menge von URLs
Beschreibung:	Generiert aus dem Namen der Presentation Unit und den Primärschlüsseln der Instanzen der Δ^--Menge eine Menge von URLs und löscht diese URLs

Schritt 2: Generierung von HTML-Seiten mit Hilfe von Δ^+-Mengen
Die Generierung von HTML-Seiten, in die neue Werte eingefügt werden, erfolgt unter Verwendung der Δ^+-Mengen. Dabei ist zu berücksichtigen, daß nur die betroffenen Instanzen eines Seitentyps neu generiert werden sollen. Ist ein Seitentyp S von den Relationen R_1,... R_n abhängig, so wird für die Identifizierung der betroffenen URLs eine Relation nach der anderen durch ihre Δ^+-Menge ersetzt. Als Ergebnis werden nur die URLs geliefert, deren HTML-Seiten Instanzen aus der jeweiligen Δ^+-Menge beinhalten. Anschließend werden die zugehörigen HTML-Seiten aus den abhängigen Relationen neu generiert. Hierfür wird eine neue Funktion `genslice_add` eingeführt, die als Eingabe eine Presentation Unit erhält und als Ausgabe eine Menge von HTML-Seiten zu der jeweiligen Presentation Unit liefert. Folgender Algorithmus soll die Vorgehensweise für die Wartung eines HTML-Seitentyps bezüglich der verschiedenen Relationen verdeutlichen. Dabei seien $R_1 \ldots R_n$ die Relationen, von denen der Slice-Typ abhängig ist und $\Delta^+(R_n)$ die Menge der Instanzen, die neu in die Relation R_n eigefügt wurden.

Funktion:	`genslice_add`
INPUT:	Name einer Presentation Unit (Slice-Typ)
	Δ^+-Mengen der abhängigen Basis-Relationen
	Abhängige Basis-Relationen
OUTPUT:	Menge von HTML-Seiten
Beschreibung:	Generiere <Presentation Unit>
	mit Hilfe von $\Delta^+(R_1), R_1, R_2, \ldots R_n$;
	...
	Generiere <Presentation Unit>
	mit Hilfe von $R_1, \ldots R_{n-1}, \Delta^+(R_n), R_n$;

Schritt 3: Generierung von HTML-Seiten mit Hilfe von Δ^--Mengen
Dies betrifft Löschungen in Basis-Relationen, deren Instanzen Komponenten anderer Seitentypen bilden und deshalb nicht an der Generierung von URLs beteiligt sind. Für die Generierung der HTML-Seiten ist neben dem aktuellen Zustand einer Basis-Relation auch der alte Zustand der Basis-Relation relevant. Während der aktuelle Zustand einer Relation für die Generierung der betroffenen HTML-Seiten dient, wird der alte Zustand verwendet, um die jeweiligen HTML-Seiten zu finden, die neu generiert werden müssen. Hierfür wird eine neue Funktion `genslice_del` eingeführt, die als Eingabe einen Seitentyp

erhält und als Ausgabe eine Menge neu generierter HTML-Seiten liefert. Sei S ein Seitentyp und von den Basis-Relationen $R_1,\ldots R_n$ abhängig, wobei R_1 die Owner-Relation der Presentation Unit sei. $\Delta^-(R_n)$ sei die Menge der Instanzen, die aus der Relation R_n gelöscht wurden. Dann werden die HTML-Seitentypen nach folgenden Regeln aufgebaut.

Funktion:	`genslice_del`
INPUT:	Name einer Presentation Unit (Slice-Typ)
	Δ^--Mengen der abhängigen Basis-Relationen
	Abhängige Basis-Relationen
	Alter Zustand der Basis-Relationen
OUTPUT:	Menge von HTML-Seiten
Beschreibung:	Generiere <Presentation Unit>
	mit Hilfe von $R_1, \Delta^-(R_2), R_2, R_3, OldR_3, \ldots R_n, OldR_n$;
	...
	Generiere <Presentation Unit>
	mit Hilfe von $R_1, R_2, OldR_2, \ldots \Delta^-(R_n), R_n$;

Die Δ^--Menge der Owner-Relation R_1 braucht nicht mehr berücksichtigt werden, da bereits alle korrespondierenden URLs in Schritt 1 gelöscht wurden.

7.3 Zusammenfassung

Die Ablage von Struktur, Navigation und Layout einer Hypertext-Applikation innerhalb eines relationalen Datenbanksystems eröffnet vielfältige Möglichkeiten. In dem vergangenen Kapitel wurden Lösungsansätze zur Realisierung von Ad-hoc-Anfragebearbeitung und inkrementeller Wartung von Hypertext-Views skizziert. Hierfür wurde das Meta-Modell für die einzelnen Anwendungen geeignet erweitert. Bei der Ad-hoc-Anfragebearbeitung ergibt sich für den Anwender die Möglichkeit, auf der Basis von SQL einfache Anfragen über einen Web-Browser abzusetzen. Die Ergebnispräsentation erfolgt mit Hilfe des im Verlauf des Entwurfsprozesses definierten Hypertext-Modells der Web-Site. Dabei werden drei verschiedene Anfragetypen unterschieden: Ein Slice/Eine Relation-Anfrage, Viele Slices/Eine Relation-Anfrage und Viele Relationen-Anfrage.

Neben der online-Seitengenerierung können die Datenbankinhalte auch offline in materialisierte HTML-Seiten ausgespoolt werden. Nachteil dieser Variante ist die erschwerte Wartung von HTML-Seiten und die hierdurch bedingten Konsistenzverletzungen zwischen Basis-Relationen und Hypertext-Views. Auf der Basis der Delta-Mengen-Technik, die in Basis-Relationen durchgeführte Änderungen protokolliert und dem erweiterten Meta-Schema wird ein Algorithmus zur inkrementellen Wartung von materialisierten HTML-Seiten skizziert. Die inkrementelle Wartung der von einer Änderung betroffenen HTML-Seiten erfolgt in drei Schritten: im ersten Schritt werden alle HTML-Seiten eines Seitentyps gelöscht, deren URL aus den Primärschlüsseln der gelöschten Tupel aufgebaut ist; im

zweiten Schritt werden alle HTML-Seiten neu generiert, deren Inhalt ergänzt wird. Hierzu werden die Δ^+-Mengen der abhängigen Basis-Relationen ausgewertet. Im letzten Schritt werden alle HTML-Seiten neu generiert, aus deren Inhalt Tupel gelöscht werden sollen. Hierfür sind die Δ^--Mengen relevant. Die Abfolge der drei Schritte für jede Presentation Unit einer Web-Site ermöglicht die inkrementelle Wartung materialisierter HTML-Seiten auf der Basis eines Meta-Modells. Nicht beschrieben wurde dabei die Abarbeitung und Markierung einzelner Einträge der Δ-Mengen für die Aktualisierung verschiedener Seitentypen.

Kapitel 8

Zusammenfassung und Ausblick

8.1 Zusammenfassung

Datenbankbasierte Web-Anwendungen stellen heute neue Anforderungen an Anwendungsentwicklung, logische Modellierung, Datenspeicherung und Datenverwaltung. Die im Rahmen dieser Arbeit vorgestellte HDBM (Hypermedia Database Methodology) gewährleistet einen durchgängigen Entwurfsprozeß und ein integriertes Management großer Web-Sites auf der Basis datenbankbasierter Modellierungskonzepte.

Die HDBM basiert auf bekannten konzeptuellen und logischen Datenmodellen aus dem Datenbank- und Hypermedia-Bereich, die in geeigneter Weise kombiniert und erweitert wurden. Den Kern der HDBM bildet ein neu entwickeltes Meta-Schema, das die Abbildung der Anwendungsdomäne auf eine Hypertext-Anwendung definiert. Die Relationen des Meta-Schemas zusammen mit den Relationen des Anwendungsbereichs, den sogenannten Basis-Relationen, werden als HDBM-Web-Datenbank-Schema bezeichnet.

Der Entwurfsprozeß von HDBM ist – ähnlich dem traditionellen DB-Entwurf – aus fünf Designphasen aufgebaut: Anforderungsanalyse, Konzeptuelles Design, Logisches DB-Design, Logisches Hypertext-Design und Physisches Design. Das konzeptuelle Design umfaßt ER-Design, Navigational-Design, Slice-Design und Application-Design und basiert auf den Modellierungskonzepten des ER-Modells und des erweiterten RMM-Datenmodells. Dies ermöglicht sowohl die konzeptuelle Modellierung der Anwendungsdomäne als auch die Modellierung von Hypertext-Struktur und -Navigation einer Web-Site.

Im Verlauf des logischen Designs werden die konzeptuellen Schemata mit Hilfe von Abbildungsregeln auf ein relationales Web-Datenbank-Schema abgebildet. Dabei erfolgt die Abbildung des Entity-Relationship-Schemas auf die Basis-Relationen der Datenbank. Die Hypertext-Struktur der Anwendung wird auf Instanzen des Meta-Schemas abgebildet. Die Befüllung der Meta-Relationen erfolgt online über hierfür entwickelte CASE-Werkzeuge, die im Rahmen einer anderen am FORWISS durchgeführten Forschungsarbeit genauer beschrieben werden. Die Ablage von Hypertext-Struktur, -Navigation und -Präsentation in einer Datenbank ermöglicht auf der einen Seite die schnelle Entwicklung, Wartung und hohe Konfigurierbarkeit der Hypertext-Anwendung mit Hilfe jedes beliebigen Web-Browsers.

Programminstallationen auf dem Client oder die Verwendung von FTP zur Fernwartung von HTML-Seiten sind nicht nötig, da alle Daten online über das Web abgerufen und gewartet werden. Auf der anderen Seite gewährleistet die Ablage sämtlicher Daten innerhalb einer Datenbank die Zuverlässigkeit, Erweiterbarkeit, Sicherheit und Konsistenz des Datenbestandes.

Zusammenfassend bringt die integrierte Verwaltung von Anwendungsdomäne und Hypertext-Anwendung innerhalb eines Datenbanksystems folgende Vorteile mit sich:

- **Durchgängiger Entwurfsprozeß:** Der Entwurf der Datenbank und der zugehörigen Hypertext-Anwendung erfolgen durch aufeinander abgestimmte konzeptuelle und logische Datenmodelle. Die Abfolge der Entwurfsphasen ermöglicht einen durchgängigen Entwurfsprozeß.
- **Rapid Prototyping:** Die Erfassung von Meta-Daten ermöglicht die schnelle Erstellung eines Prototyps, der die Grundfunktionalität des Gesamtsystems veranschaulicht.
- **Einfache Wartbarkeit und Skalierbarkeit:** Die Eingabe und Wartung der Hypertext-Anwendung erfordert keine Programmierkenntnisse. Durch die klare Trennung von Hypertext-Struktur, -Layout und -Navigation ist die einfache Skalierbarkeit des Systems gegeben.
- **Konsistenz zwischen DB- und Hypertext-Schema:** Die Ablage der Hypertext-Meta-Daten innerhalb einer Datenbank ermöglicht die Überwachung der Konsistenz zwischen Meta-Daten und Basis-Relationen durch das Datenbanksystem.
- **Unterstützung von Ad-hoc-Anfragebearbeitung:** Durch die Abbildung der Hypertext-Struktur auf ein Meta-Schema, können diese Informationen bei der Auswertung beliebiger SQL-Anfragen an die Basis-Relationen berücksichtigt und zur Generierung der Ergebnis-Seiten genutzt werden.
- **Inkrementelle Wartung von materialisierten Views:** Die im Meta-Schema definierte Abbildung zwischen HTML-Seitentypen und Basis-Relationen ermöglicht die inkrementelle Wartung von offline aus der Datenbank generierten HTML-Seiten, die im Dateisystem materialisiert werden.

Die Ablage von Meta-Daten innerhalb einer Datenbank bietet viele Vorteile, begrenzt aber auch die Möglichkeiten der freien Gestaltung von HTML-Seiten. Als Basis für den Entwurf des Meta-Modells diente deshalb ein bekanntes Hypermedia-Entwurfsmodell (RMDM), das geringfügig erweitert wurde.

Das neu entwickelte Meta-Schema bietet:

- **Schachtelung von Seitentypen:** HTML-Seiten sind aus verschiedenen Fragmenten (Slices) aufgebaut, die beliebig tief ineinander geschachtelt sein können. Dies ermöglicht die Anzeige von Attributen verschiedener, miteinander in Beziehung stehender Basis-Relationen auf einer HTML-Seite.

- **Intra-Record- und Inter-Record-Navigation:** Um zwischen Slices navigieren zu können, werden zwei Navigationsarten unterschieden: Inter-Record- und Intra-Record-Navigation. Diese gestatten sowohl die Navigation zwischen verschiedenen Datenbank-Tupeln als auch zwischen verschiedenen Slices ein und desselben Datenbank-Tupels.

- **Durchgängige Positionierung der Attribute:** Durch die einheitliche Modellierung der Elemente eines Slice-Typs kann die Intra-Dokument-Struktur unabhängig von dem jeweiligen Typ des Elements (M-Slice, Attribut, Hyperlink oder Zugriffsstruktur) festgelegt werden. Dies ermöglicht eine durchgängige Positionierung aller Elemente auf einer HTML-Seite.

- **Anwendungsspezifische Hyperlinks:** Hyperlinks werden getrennt von der Struktur der HTML-Dokumente gespeichert. Die Anker von berechneten Hyperlinks können aus verschiedenen Attributen einer Basis-Relation aufgebaut sein. Reihenfolge und Anzahl der Attribute werden in der Relation ANKERSLICE bestimmt.

- **Verschiedene Zugriffsstrukturen:** Neben einem internen und externen Index wird eine Guided Tour und eine Indexed Guided Tour als Zugriffsstruktur angeboten. Bei einem Index werden die Instanzen eines M-Slices als Liste dargestellt. Dies ermöglicht die Generierung beliebig komplexer Listen, mit und ohne Hyperlinks.

- **Erweiterbare Web-Typen:** Die Web-Typen werden in einer separaten Relation zusammen mit den für die Behandlung des Web-Typs notwendigen HTML-Tags abgelegt. Dies ermöglicht die Definition neuer Web-Typen und die einfache Skalierbarkeit des Meta-Schemas.

- **Konfigurierbares Layout:** Das Layout einer Anwendung wird durch eine separate Meta-Relation verwaltet und kann unabhängig von der Hypertext-Struktur verändert werden.

Die Architektur des Gesamtsystems wurde offen gestaltet. Als Datenbanksystem wurde Oracle zusammen mit dem Web-Application-Server 3.0 eingesetzt. Die dynamische Generierung der HTML-Seiten mit Hilfe der Meta-Daten erfolgt durch ein hierfür entwickeltes Tool, den `webgenerator`. In diesem Zusammenhang wurden zwei Implementierungsvarianten vorgestellt. Bei der ersten Variante werden die Meta-Daten zur Laufzeit durch PL/SQL-Prozeduren ausgewertet und für die Generierung des jeweiligen HTML-Seitentyps verwendet. Der Name des Seitentyps und Primärschlüssel der Instanz wird der PL/SQL-Prozedur als Parameter übergeben. Bei der zweiten Variante wertet ein Werkzeug die Meta-Daten im Vorfeld aus und generiert für jeden Seitentyp eine PL/SQL-Prozedur. Großer Vorteil dieser Alternative ist, daß die so generierten PL/SQL-Prozeduren vom Anwender nachbearbeitet und wieder in der Datenbank abgelegt werden können. Außerdem bringt die Auswertung der Meta-Daten im Vorfeld eine Performance-Verbesserung mit sich. Beide Alternativen wurden prototypisch implementiert und hinsichtlich ihrer Vor- und Nachteile und der unterschiedlichen Performance diskutiert.

8.2 Ausblick

Die in dieser Arbeit vorgestellte Entwurfsmethodik deckt die wesentlichen Eigenschaften für die Entwicklung von datenbankgestützten Web-Anwendungen ab. Die Ablage von Hypertext-Meta-Daten innerhalb eines Datenbanksystems eröffnet noch weiterführende Funktionalitäten, die in diesem Kapitel erläutert werden.

Die HDBM-Entwurfsmethode und das hierzu gehörende Meta-Schema bieten zur Strukturierung und Abbildung von HTML-Formularen keine Unterstützung, da dies der Inhalt einer anderen am FORWISS durchgeführten Forschungsarbeit ist. Die Recherche in einer datenbankbasierten Web-Anwendung erfordert die dynamische Generierung von HTML-Eingabeformularen, in die der Anwender die Selektionskriterien einträgt. Nach dem Ausfüllen und Abschicken des Formulars erhält der Anwender eine Ergebnisliste mit einer Kurzdarstellung der mit Hyperlinks unterlegten Treffer. Dies bietet gleichzeitig den Einstiegspunkt in das Informationssystem. Über die Auswahl eines Treffers gelangt der Anwender in die durch diese Arbeit beschriebene Detail-Darstellung der Datenbank-Tupel. Für die Modellierung von HTML-Formularen muß das vorgestellte Meta-Schema u.a. um verschiedene Slice-Typen für die Recherche (Query-Slice) und Trefferanzeige (Result-Slice) erweitert werden. Ähnliches gilt für die Modellierung der HTML-Formulare zur Online-Wartung der Daten der Anwendungsdomäne.

Der Einsatz eines Meta-Schemas ermöglicht die ausschließliche Entwicklung und Wartung der datenbankgestützten Web-Anwendung über einen Web-Browser. Dies umfaßt das Anlegen der Basis-Relationen in der Datenbank, die Befüllung des Meta-Schemas und die Eingabe und Wartung der Daten der Anwendungsdomäne. Während das Anlegen von Relationen und die Wartung der Anwendungsdomäne bereits durch verschiedene auf dem Markt befindliche Tools im Web-Browser unterstützt werden, erfordert die Erfassung der Meta-Daten die Entwicklung eines entsprechenden Werkzeugs. Im Rahmen dieser Arbeit wurden die Meta-Daten über ein einfaches Browser-basiertes Interface in die Datenbank eingegeben. Hierfür waren Kenntnisse über die Basis-Relationen und Meta-Relationen erforderlich. Für die Zukunft ist die Gestaltung einer benutzerfreundlicheren Komponente auf der Basis von Java geplant. Mit Hilfe des Data Dictionary der Datenbank können Schema-Informationen zur dynamischen Generierung der Seitentyp-Vorlagen genutzt werden. Dabei erscheint eine schrittweise Vorgehensweise sinnvoll. Im ersten HTML-Formular wählt der Anwender die Owner-Relation und den gewünschten Titel des HTML-Seitentyps aus. Anschließend werden in weiteren HTML-Formularen Attribute, Position, Bezeichnung, Web-Typ, Zugriffsstrukturen, Hyperlinks und Layout des Seitentyps definiert. Erst nach der Bestätigung und dem Abschicken des letzten Formulars werden alle bis dahin gesammelten Informationen in die Datenbank durchgeschrieben (multiple-page-Transaktion). Eine andere Variante zur Erfassung der Meta-Daten ist die Verwendung von CASE-Tools, die aus dem konzeptuellen Hypertext-Modell die entsprechenden Einträge im Meta-Schema generieren.

Ein in dieser Arbeit nicht ausführlicher behandelter Bereich sind Benutzerprofile und Benutzerrechte (Authentifizierung und Autorisierung). Benutzerrechte von Anwendern

im Web werden im Falle einer datenbankgestützten Web-Anwendung durch das Datenbanksystem verwaltet. Der Web-Server reicht Kennung und Paßwort an den DB-Client weiter, der sich für den Anwender in der Datenbank anmeldet. Neben den vom Datenbanksystem verwalteten Benutzerrechten und Rollen ist es für Web-Anwendungen sinnvoll, auch die Berechtigung zur Ansicht einzelner Seitentypen im Meta-Schema abzulegen (Browse-Berechtigungen). Dies ermöglicht, den individuellen Zugriff einzelner Benutzer oder Benutzergruppen auf der Basis verschiedener Benutzerprofile zu gestalten. Weiterhin können für individuelle Benutzer im Meta-Schema Interessenbereiche definiert werden, die bei der dynamischen Generierung der HTML-Seiten berücksichtigt werden. Die Hypertext-Oberfläche paßt sich in diesem Fall adaptiv dem Anwenderprofil an. Die Verwaltung der verschiedenen Hypertext-Sichten und Benutzergruppen in der Datenbank ermöglicht daneben die Kontrolle der Aktivitäten einzelner Anwender durch Log-Files.

Die Ablage verschiedener Browser-Typen und -Versionen mit deren spezifischen Eigenschaften bietet ein weiteres Anwendungsfeld von Meta-Daten. Abhängig von dem jeweiligen Browser-Profil, kann die erstellte Hypertext-Oberfläche adaptiv hinsichtlich StyleSheets, XML und JavaSkript optimiert werden. Style Sheets ermöglichen die klare Trennung von Layout und Inhalt einer Web-Site und codieren die Präsentation in wiederverwendbare Komponenten. Leider werden Style Sheets zur Zeit von gängigen Browsern nicht vollständig unterstützt und konnten deshalb bei der Modellierung nicht berücksichtigt werden.

Datenbankbasierte Web-Sites setzten sich üblicherweise aus einem dynamisch aus der Datenbank generierten und einem statischen Teil zusammen. Bei der HDBM wird der statische Teil noch außerhalb des DB-Systems abgelegt. Um die einheitliche Entwicklung und Wartung der Anwendung zu gewährleisten, wäre es aber sinnvoll auch die statischen HTML-Dokumente beispielsweise in BLOBs innerhalb der Datenbank abzulegen und über einen Web-Browser zu warten.

In jüngster Zeit gewinnen Meta-Daten für unterschiedlichste Anwendungsgebiete zunehmend an Bedeutung. Seit dem Aufkommen des World Wide Web, Mitte der neunziger Jahre, stellten Millionen von Organisationen und Privatleuten ihre Informationen in unterschiedlichsten Formaten und Sprachen im World Wide Web zur Verfügung. Als Antwort hierauf wurde 1995 die sogenannte Metadaten-Bewegung zur strukturierten Ablage von Daten über Daten (Meta-Daten) initiiert [Bak99].

> *The creation and maintenance of metadata is so fundamental to the mission of some activities and business that one might even speak of a large and growing metadata industry.*

Dabei stimmen Forscher heute in der Annahme überein, daß nicht ein Typ von Meta-Daten für alle Anwendungen geeignet ist. Auch bezogen auf das Anwendungsgebiet der datenbankbasierten Web-Sites eröffnen Meta-Daten über die Strukur, Navigation und das Layout von Daten im Web neue Perspektiven und verschiedenste Anwendungsfelder.

Anhang A

BNF-Regeln

Anbei werden die Regeln der im Kapitel 6.2.2 verwendeten *Erweiterten Backus-Naur-Form* beschrieben. Im Gegensatz zu der einfachen Backus-Naur-Form werden die eckigen und geschweiften Klammern ergänzt. Terminale, die nur auf der rechten Regelseite vorkommen dürfen werden zur besseren Lesbarkeit **fett** gedruckt:

- ⟨ ... ⟩: Nichtterminale
- (...): Terminalzeichen, die nicht näher spezifiziert werden
- ::= : Trennzeichen zwischen linker und rechter Seite
- **a, b, c ...1, 2, 3 ...**: Terminale werden zur besseren Lesbarkeit fett gedruckt
- { ... }: Optionale Wiederholung
- [...]: Optionen
- | : Alternativen

Anhang B

Slice-Typen

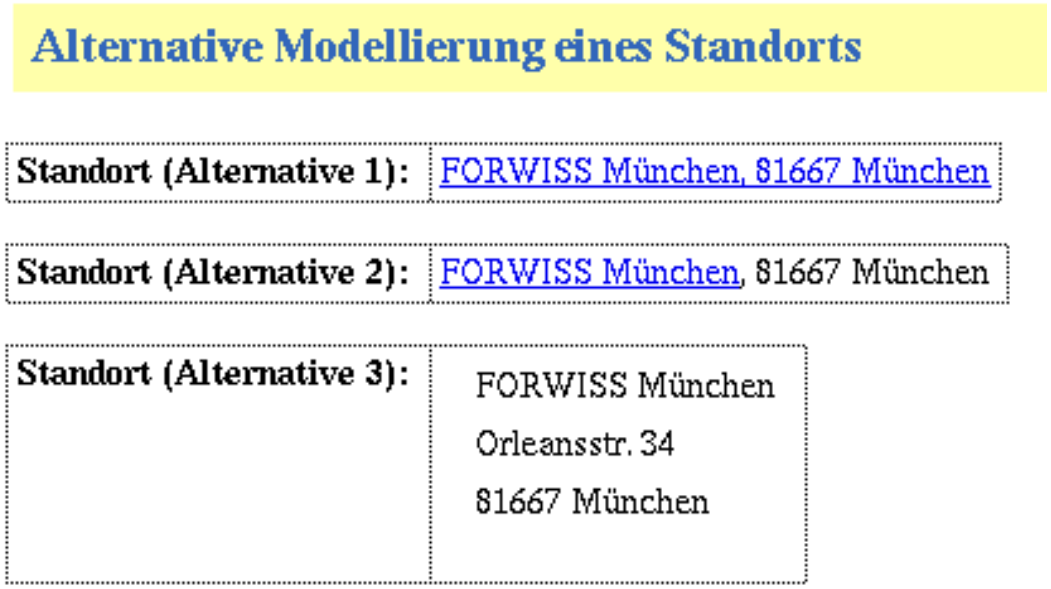


Abbildung B.1: Alternative Modellierung eines Standorts

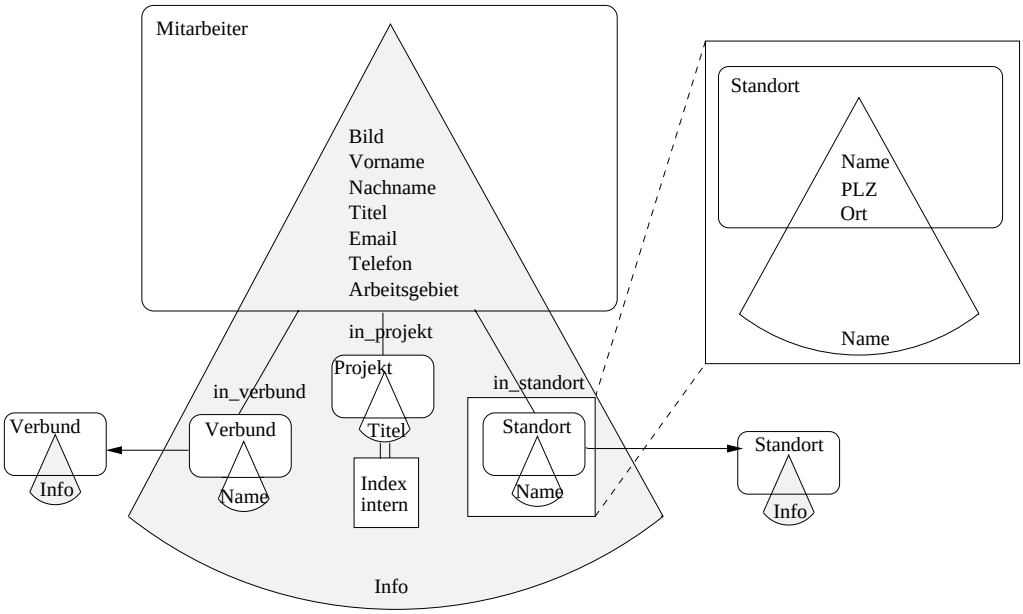


Abbildung B.2: Modellierung des Standort-Slice: Alternative 1

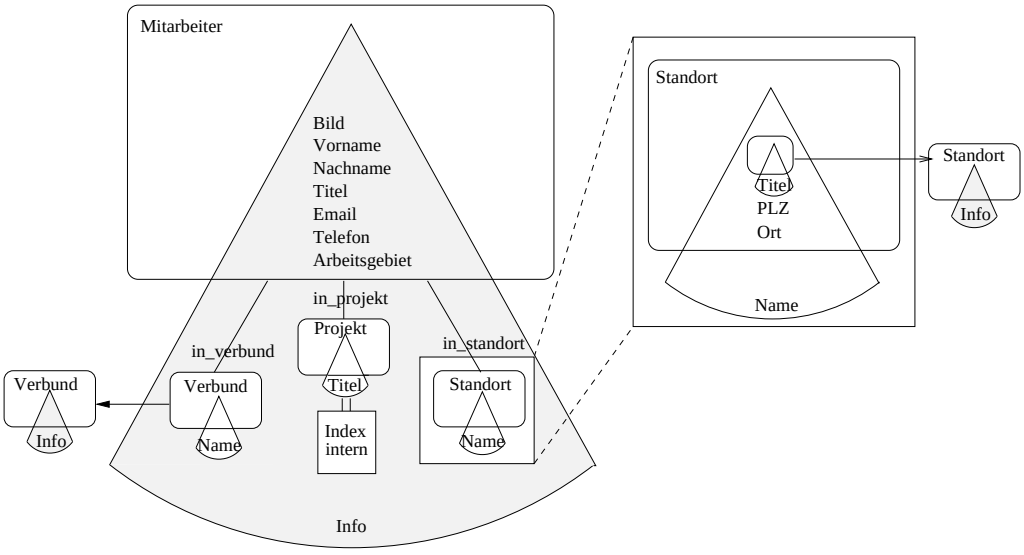


Abbildung B.3: Modellierung des Standort-Slice: Alternative 2

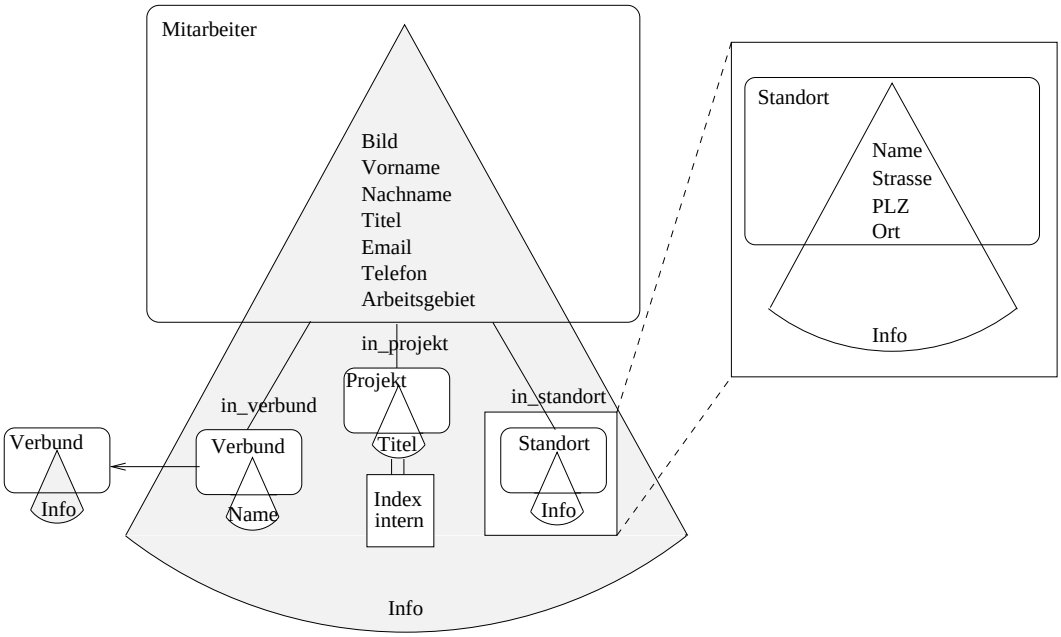


Abbildung B.4: Modellierung des Standort-Slice: Alternative 3

Abbildung B.5: Alternative Modellierung einer Projektliste

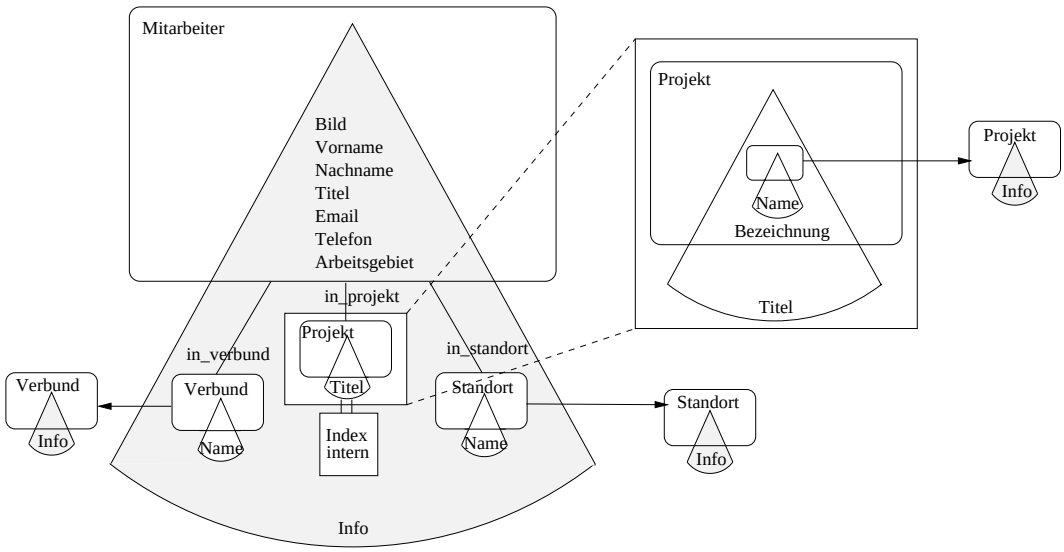


Abbildung B.6: Modellierung des Projekt-Slice: Alternative 1

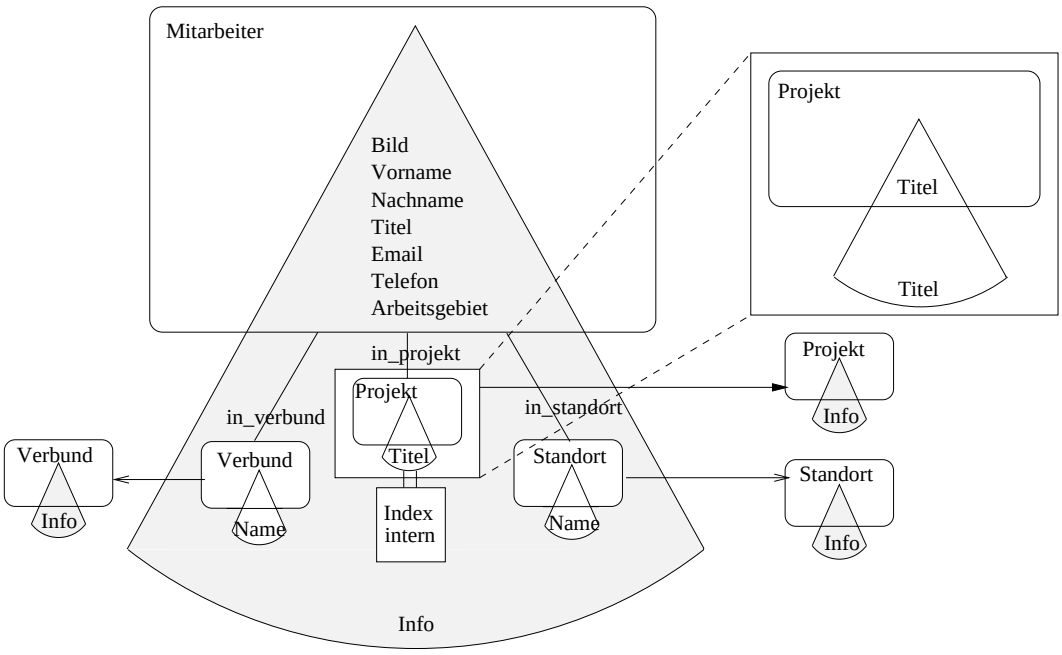


Abbildung B.7: Modellierung des Projekt-Slice: Alternative 2

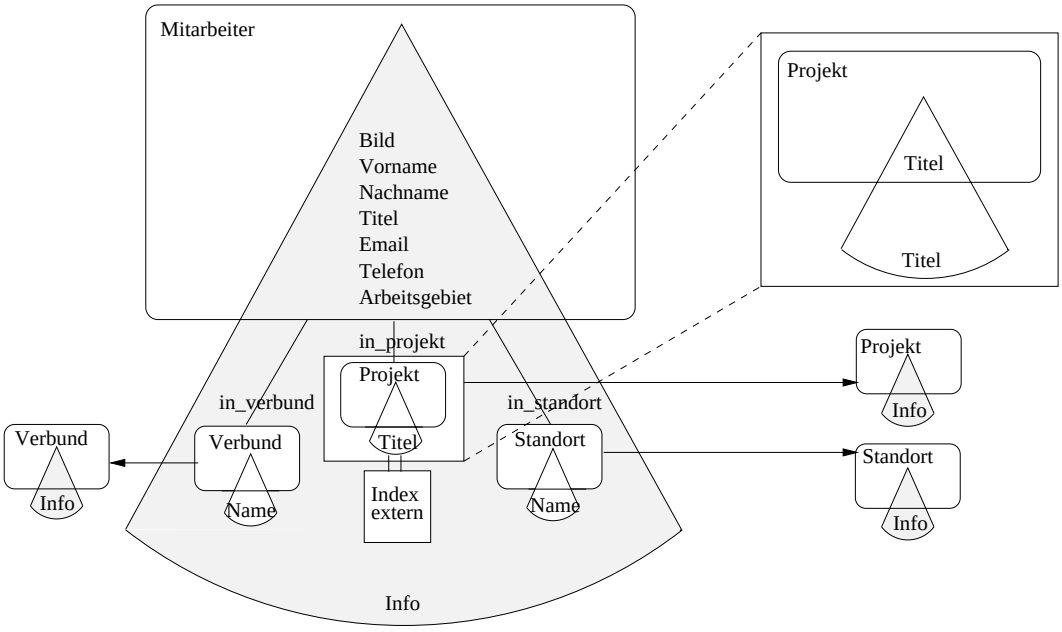


Abbildung B.8: Modellierung des Projekt-Slice: Alternative 3

Anhang C

Performance-Messung: abayfor-Datenbank-Schema

```
CREATE TABLE BILD
       (NUM             NUMBER(4) CONSTRAINT pk_bild PRIMARY KEY,
        BILD            BLOB
       );

CREATE TABLE LOGO
       (NUM             NUMBER(4) CONSTRAINT pk_logo PRIMARY KEY,
        LOGO            BLOB
       );

CREATE TABLE VERBUND
       (VNUM            NUMBER(4) CONSTRAINT pk_verbund PRIMARY KEY,
        NAME            VARCHAR2(1000),
        BEZEICHNUNG     VARCHAR2(50),
        GRDATUM         VARCHAR2(50),
        BESCHREIBUNG    VARCHAR2(4000),
        URL             VARCHAR2(100),
        LOGO            CONSTRAINT fk_logo REFERENCES LOGO(NUM)
       );

CREATE TABLE STANDORT
       (SNUM            NUMBER(4) CONSTRAINT pk_standort PRIMARY KEY,
        NAME            VARCHAR2(500),
        STRASSE         VARCHAR2(50),
        PLZ             VARCHAR2(50),
        ORT             VARCHAR2(50),
        WEGBESCHR       VARCHAR2(1000)
       );
```

```
CREATE TABLE MITARBEITER
       (M#              NUMBER(4) CONSTRAINT pk_mitarbeiter PRIMARY KEY,
        VNUM            CONSTRAINT fk_verbund REFERENCES VERBUND(VNUM),
        SNUM            CONSTRAINT fk_standort REFERENCES STANDORT(SNUM),
        VNAME           VARCHAR2(50),
        NNAME           VARCHAR2(50),
        TITEL           VARCHAR2(50),
        EMAIL           VARCHAR2(50),
        TELEFON         VARCHAR2(50),
        TELEFAX         VARCHAR2(50),
        BEREICH         VARCHAR2(1000),
        BILD            CONSTRAINT fk_bild REFERENCES BILD(NUM)
       );

CREATE TABLE PROJEKT
       (PNUM            NUMBER(4) CONSTRAINT pk_projekt PRIMARY KEY,
        NAME            VARCHAR2(500),
        BEZEICHNUNG     VARCHAR2(500),
        STARTDAT        VARCHAR2(50),
        ENDEDAT         VARCHAR2(50),
        BESCHREIBUNG    VARCHAR2(4000),
        URL             VARCHAR2(200)
       );

CREATE TABLE PROJ_MITARB
       (M#      CONSTRAINT fk_mitarbeiter REFERENCES MITARBEITER(M#),
        PNUM    CONSTRAINT fk_projekt REFERENCES PROJEKT(PNUM)
       );
```

Anhang D

Performance-Messung: abayfor-HTML-Seitentypen

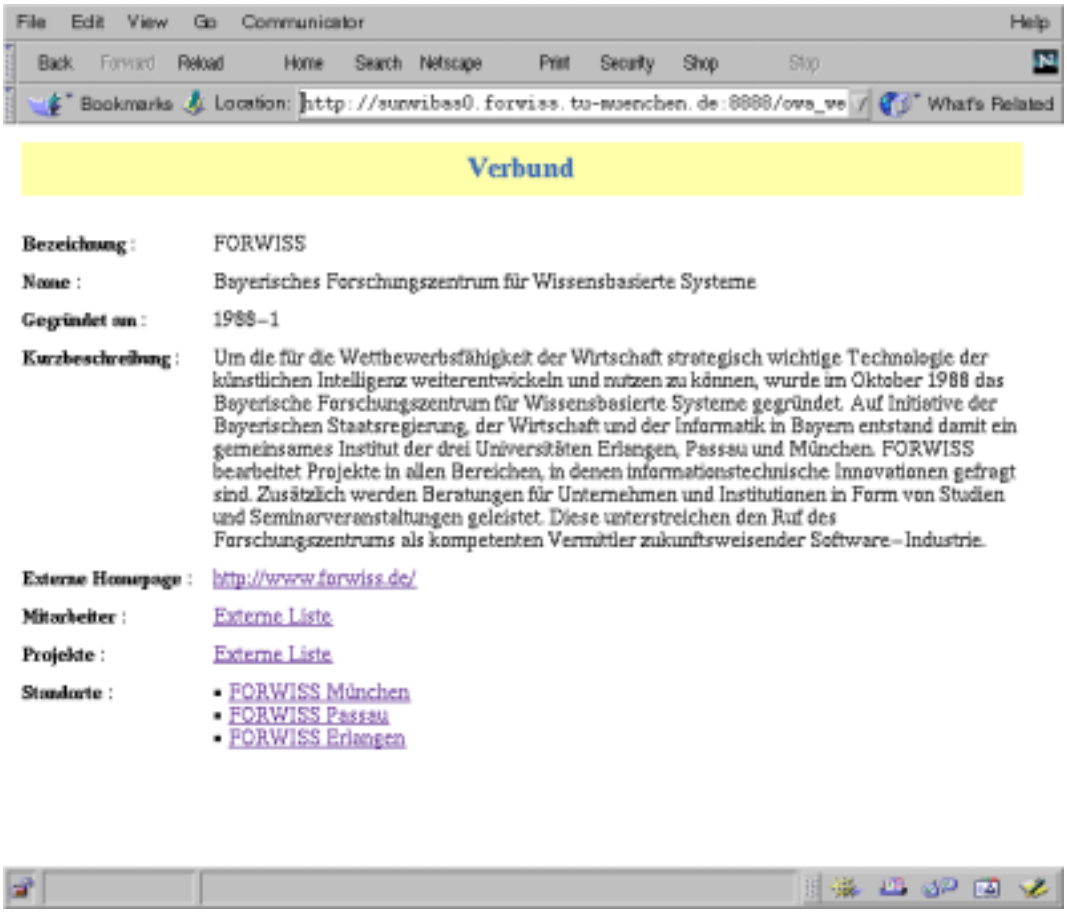


Abbildung D.1: Verbund-HTML-Seite

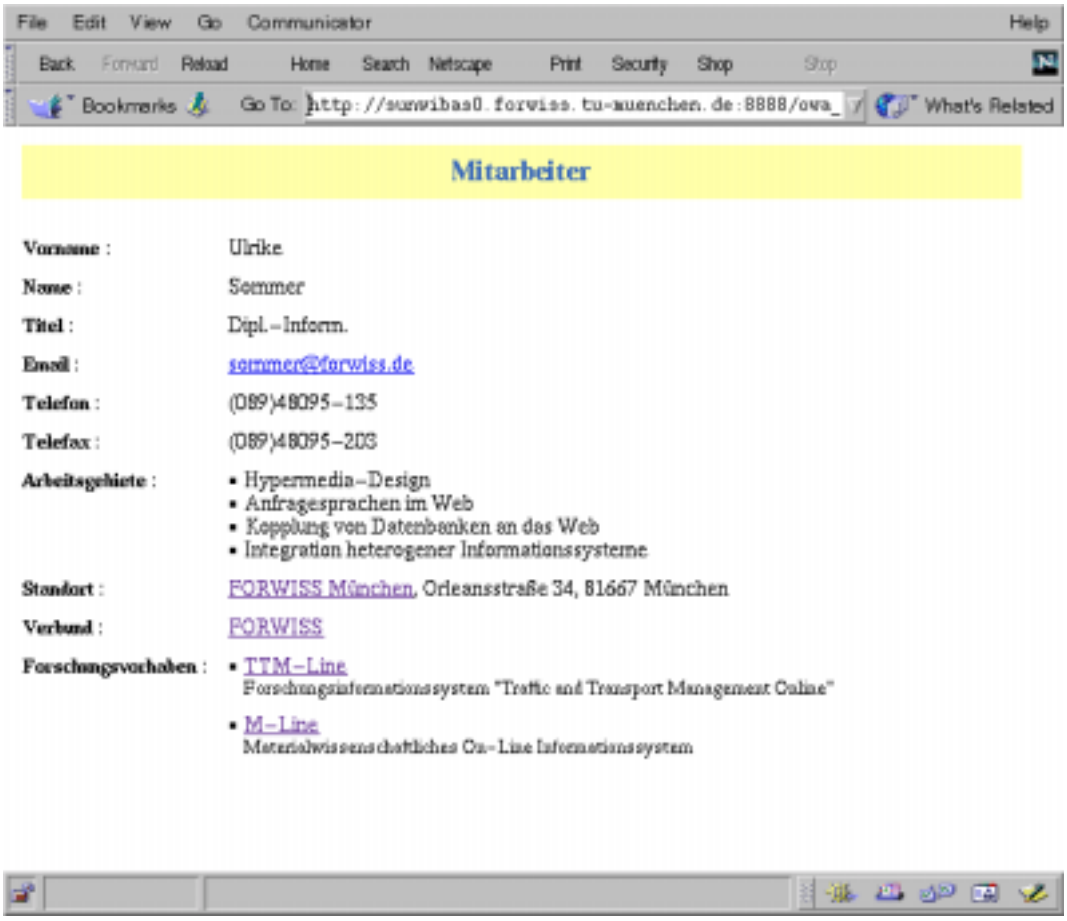


Abbildung D.2: Mitarbeiter-HTML-Seite

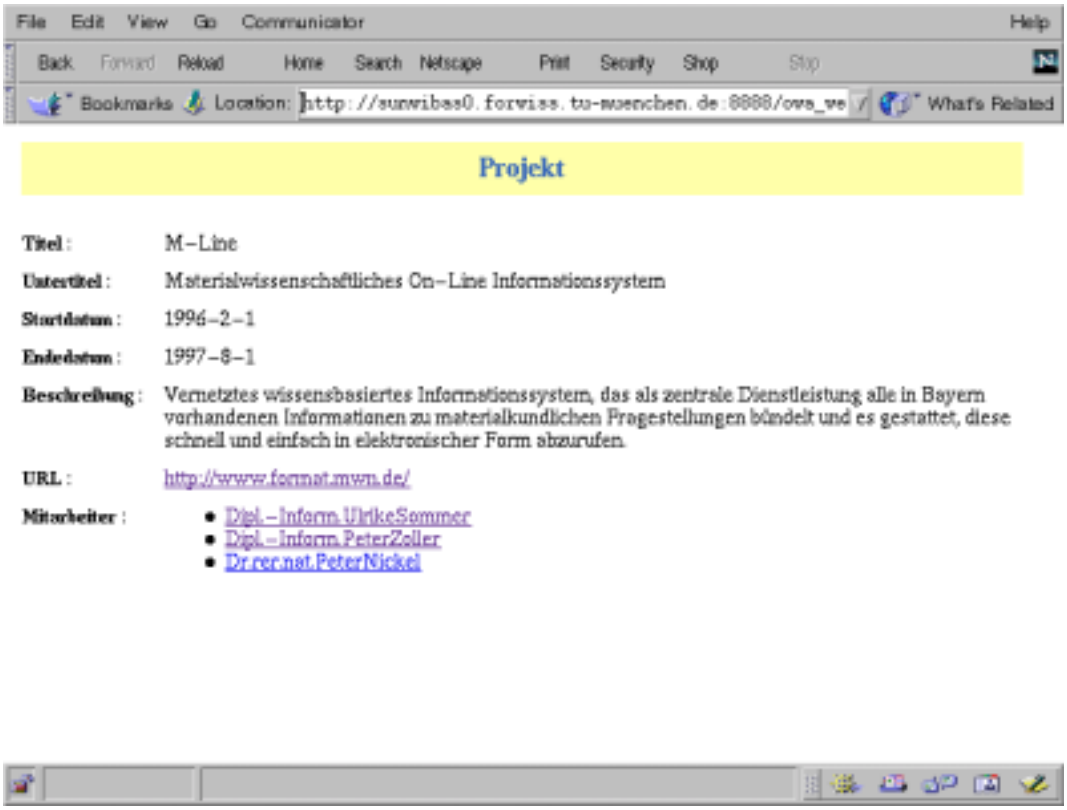


Abbildung D.3: Projekt-HTML-Seite

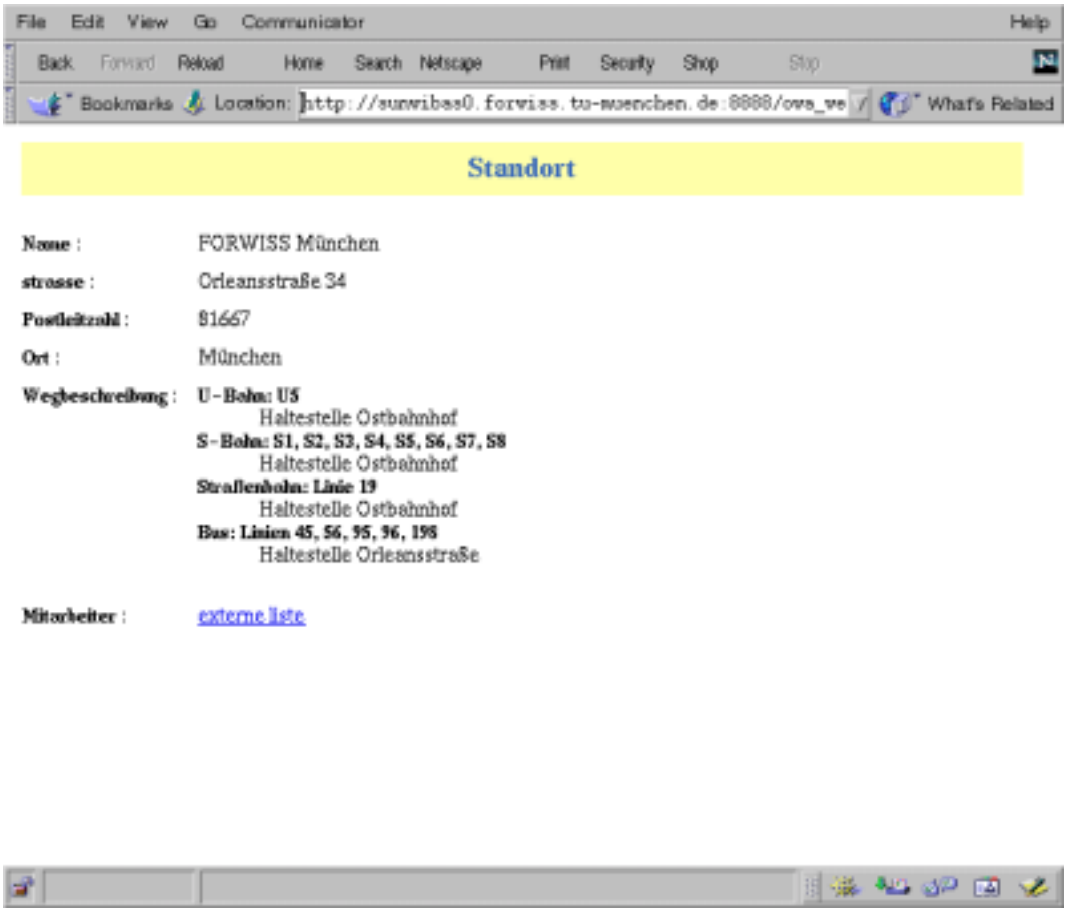


Abbildung D.4: Standort-HTML-Seite

Anhang E

Glossar

Application-Design: Im Verlauf des Application-Designs erfolgt die Abbildung von Slice-Typen auf HTML-Seitentypen. Das Hauptkonzept bei der Modellierung bilden HTML-Seitentypen, die eine homogene Menge gleich strukturierter HTML-Seiten beschreiben. Ergebnis des Application-Designs ist das Application-Diagramm, das die Web-Site auf einer logischen Ebene abbildet.

Component Slice: Als Component Slice bezeichnet man Slice-Typen, die selbst einen Bestandteil eines anderen Slice-Typs bilden. Ein Slice-Typ S ist genau dann ein Component wenn es mindestens eine Presentation Unit gibt, die S beinhaltet.

Element: Ein M-Slice wird durch eine Menge möglicherweise geschachtelter Elemente beschrieben. Ein Element kann entweder ein Attribut, eine Zugriffsstruktur, ein Hyperlink oder wieder ein M-Slice sein.

Guided Tour: Als Guided Tour bezeichnet man einen geführten Rundgang durch die Instanzen eines HTML-Seitentyps. Jede HTML-Seite beinhaltet einen Verweis zur vorherigen und zur nächsten HTML-Seite.

Head Slice: Werden einem Owner-Entity verschiedene Slice-Typen zugeordnet, die über Intra-Record-Navigation miteinander in Beziehung stehen, so bildet ein Slice-Typ den Einstiegspunkt zu diesem Owner-Entity. Diesen Einstiegspunkt bezeichnet man als Head Slice.

HTML-Seitentyp: HTML-Seitentypen beschreiben eine homogene Menge gleich strukturierter HTML-Seiten. Eine HTML-Seite entspricht einem Objekt mit einem eindeutigen Identifikator (URL) und verschiedenen Web-Attributen und bildet eine Instanz eines HTML-Seitentyps. Jeder HTML-Seitentyp entspricht auf konzeptueller Ebene einer Presentation Unit.

Hyperlink: Ein Hyperlink in HTML besteht aus einem Quellanker und einem Ziel, das ein Ziel-Anker oder ein HTML-Knoten sein kann. Das Ziel wird durch einen URI beschrieben. In HDBM bestehen Hyperlinks aus einem Quell-Anker und einem

Ziel-Knoten, wobei Anker und Ziel M-Slices sind. Der Anker ist der Bereich, der als Ausgangspunkt für einen Link dient (anklickbarer Bereich). Das Ziel ist eine HTML-Seite.

Hypermedia: Vereinigung von Hypertext und Multimedia. Hypertext-Knoten enthalten neben Text auch Graphiken, Audio, Video, Animationen und andere Elemente.

Hypertext: Nicht-lineare Repräsentation von Wissen. Hypertext bildet ein Netzwerk aus Knoten (Text) und Kanten (Links), über die sich ein Benutzer verschiedene Wege durch ein Netzwerk suchen kann.

Hypertext-Anwendung und Hypertext-Applikation: Eine Anwendung auf der Basis eines Hypertext-Systems bezeichnet man als Hypertext-Anwendung oder Hypertext-Applikation. Hypertext-Systeme bilden die Umgebung für Hypertext-Anwendungen.

Hypertext-Knoten: Ein Hypertext-Knoten ist eine in sich abgeschlossene Informationseinheit. Durch die Vernetzung der Informationseinheiten entsteht Hypertext. Im WWW entspricht ein Hypertext-Knoten einem HTML-Dokument.

Hypertextsystem: System zum Erstellen, Navigieren und Verwalten von Hypertext. Beispiele für Hypertextsysteme sind das World Wide Web und das Dexter-Referenz-Modell.

Index: Als Index bezeichnet man eine Liste von Instanzen eines Slice-Typs. Häufig sind dies Hyperlinklisten. Ein Index darf in der HDBM aber auch nur aus Attributen aufgebaut sein.

Indexed Guided Tour: Als Indexed Guided Tour bezeichnet man eine Kombination zwischen einem externen Index und einer Guided Tour.

Inter-Record-Navigation: Navigation zwischen verschiedenen Datenbank-Tupeln mit Hilfe einer der in der RMM definierten Zugriffsstrukturen.

Intra-Record-Navigation: Navigation innerhalb ein und desselben Datenbank-Tupels. Die Darstellung eines DB-Tupels wird in diesem Fall durch verschiedene Slice-Typen realisiert, die miteinander durch Hyperlinks verbunden sind.

Materialisierte Hypertext-Views: Durch Hypertext-Views werden keine Sichten über Klassen oder Relationen sondern Hypertext-Knoten und Hyperlinks definiert. Eine Hypertext-View bezeichnet Hypertext, dessen Inhalt aus einer Datenbank generiert wurde. Bei Materialisierten Hypertext-Views werden die Daten innerhalb von Hypertext-Knoten (HTML-Dateien) im Dateisystem abgelegt.

M-Slice: Ein M-Slice gruppiert Attribute eines Entities, die eine Einheit auf einer HTML-Seite bilden. Jeder M-Slice ist eindeutig einem Entity zugeordnet, das man

als *Owner-Entity* des M-Slices bezeichnet. Das M vor dem Slice steht für die russische Matrjeska-Puppe und soll die rekursive Schachtelung von M-Slices ineinander verdeutlichen. Jeder M-Slice ist aus Elementen aufgebaut, die entweder ein Attribut, ein Hyperlink, eine Zugriffsstruktur oder wieder ein M-Slice sein können. M-Slices werden im einfachen Sprachgebrauch auch einfach als Slices bezeichnet.

Navigational-Design: Im Verlauf des Navigational-Designs erfolgt die Modellierung der Hyperlinks und Zugriffsstrukturen zwischen Slice-Typen. Das Navigational-Design entspricht der Modellierung der Inter- und Intra-Dokument-Struktur.

offline-Seitengenerierung: Bei der offline-Seitengenerierung dient das Datenbanksystem nur zur Datenspeicherung der Anwendungsdomäne. Die Daten bzw. Updates werden in regelmäßigen Abständen oder iterativ von der Datenbank in flache HTML-Dateien überspielt.

online-Seitengenerierung: Bei der online-Seitengenerierung werden die HTML-Seiten dynamisch zur Laufzeit aus dem Datenbanksystem erzeugt.

Owner-Entity und Owner-Relation: Jedem Slice ist eindeutig ein Entity-Typ zugeordnet, das als Owner-Entity des Slices bezeichnet wird. Ein Slice beschreibt die Darstellung und Strukturierung des Entities auf einer HTML-Seite. Jedes Owner-Entity wird auf relationaler Ebene auf eine Owner-Relation abgebildet.

Presentation Unit: Eine Presentation Unit ist ein Slice-Typ, dessen Instanzen auf eigenständigen HTML-Seiten angezeigt werden. Jede Presentation Unit entspricht auf logischer Ebene einem HTML-Seitentyp. Ein Slice-Typ S ist genau dann eine Presentation Unit wenn kein anderer Slice-Typ, S als Komponente beinhaltet.

RMM: Die RMM (Relationship Management Methodology) ist eine Hypermedia-Entwurfsmethode, die durch einen methodischen Design-Ansatz auf der Basis des RMM-Datenmodells (RMDM) den strukturierten Entwurf von Hypermedia-Applikationen unterstützt. Das zentrale Modellierungskonstrukt der RMM sind M-Slices.

Slice-Design: Das Slice-Design bezeichnet die Modellierung der Intra-Dokument-Struktur von HTML-Seiten. Das Hauptkonzept der Modellierung bilden M-Slices, die aus verschiedenen Elementen aufgebaut sind.

Template: HTML-Templates bezeichnen HTML-Seiten, die um Datenbankoperationen ergänzt wurden. Die Syntax ist durch eine entsprechende Grammatik definiert, welche den gewünschten Befehlsumfang bietet. Bei einer Anfrage an die Datenbank analysiert bzw. übersetzt ein Web-Gateway das HTML-Template, reicht die Anfragen an die Datenbank weiter und fügt die Anfrageergebnisse wieder in die HTML-Seite ein.

Web-Datenbank-Schema: Das HDBM-Web-Datenbank-Schema besteht aus einer Menge von Basis-Relationen und acht Meta-Relationen. Die Basis-Relationen modellieren die Anwendungsdomäne. Die Hypertext-Struktur, -Navigation und das Layout

werden auf Instanzen der Meta-Relationen abgebildet. Die acht Meta-Relationen bilden das *Meta-Schema*.

Web-Typ: Jedes Attribut eines Slices wird auf HTML-Seitenebene durch einen Web-Typ angezeigt, der zur richtigen Präsentation eines DB-Attributs auf der HTML-Seite dient (z.B. TEXT, EMAIL, URL, BOOL, DOWNLOAD, IMAGE).

Zugriffsstruktur: Eine Zugriffsstruktur definiert die Art und Weise wie auf eine Menge von Instanzen eines Slice-Typs zugegriffen wird. Es werden vier verschiedene Zugriffsstrukturen unterschieden: Index Extern, Index Intern, Guided Tour und Indexed Guided Tour.

Abbildungsverzeichnis

Literaturverzeichnis

[AAA+99] S. Abiteboul, V. Aguilera, S. Ailleret, B. Amann, S. Cluet, B. Hills, F. Hubert, J.-C. Mamou, A. Marian, L. Mignet, T. Milo, C. Souza dos Santos, B. Tessier und A. Vercoustre. XML Repository and Active Views Demonstration. In *Proceedings of the 25th International Conference on Very Large Databases (VLDB'99)*, Edinburgh, Scotland, 1999.

[Abi99] S. Abiteboul. On Views and XML. In *Proceedings of the Eighteenth ACM SIGACT-DIGMOD-SIGART Symposium on Principles of Database Systems (PODS'99)*, Philadelphia, Pennsylvania, 1999.

[ABS00] S. Abiteboul, P. Buneman und D. Suciu. *Data on the Web*. Morgan Kaufmann Publishers, San Francisco, California, 2000.

[ACHK93] Y. Arens, C. Chee, C. Hsu und C. Knoblock. Retrieving and Integrating Data from Multiple Information Sources. *International Journal of Cooperative Information Systems (IJCIS)*, 2(2):127–158, 1993.

[AGW98] R. Assfalg, U. Goebels und H. Welter. *Internet-Datenbanken: Konzepte, Methoden, Werkzeuge*. Addison-Wesley, Bonn, 1998.

[AHR+98] S. Abiteboul, J. Mc Hugh, M. Rys, V. Vassalos und J. Wiener. Incremental Maintenance for Materialized Views over Semistructured Data. In *Proceedings of 24rd International Conference on Very Large Data Bases (VLDB'98)*, New York, USA, 1998.

[AMM97a] P. Atzeni, G. Mecca und P. Merialdo. Design and Maintenance of Data-Intensive Web Sites. Technical Report 25, Dipartimento di Informatica e Automazione, Universita' di Roma Tre, Juni 1997.

[AMM97b] P. Atzeni, G. Mecca und P. Merialdo. To Weave the Web. In *Proceedings of the 23rd International Conference on Very Large Databases (VLDB'97)*, Athens, Greece, 1997.

[ANS] ANSI/X3/SPARC Study Group on Data Base Management Systems. Interim Report 75-02-08. FDT (Bulletin of ACM-SIGMOD) 7(2), 1975.

[Ass96] R. Assfalg. *Integration eines offenen Hypertext-Systems in den Internet-Mehrwertdienst World Wide Web.* Dissertation, Universität Konstanz, Hartung-Gorre Verlag, Konstanz, 1996.

[Bak99] T. Baker. Organizing the Web: an Update on the Metadata Movement. *Asia Library News*, 1999.
URL: `http://www.aliva.org/tomas_baker.htm`.

[BG98] W. Benn und I. Gringer. Zugriff auf Datenbanken über das World Wide Web. *Informatik Spektrum*, 21(1):1–8, 1998.

[BLCL+94] T.J. Berners-Lee, R. Cailliau, A. Luotonen, H. Nielsen und A. Secret. The World Wide Web. *Communications of the ACM*, 37(8):76–82, 1994.

[BLLJ98] B. Bos, H. Lie, C. Lilley und I. Jacobs. *Cascading Style Sheets, level 2 CSS2 Specification.* World Wide Web Consortium (W3C), 1998.
URL: `http://www.w3c.org/TR/REC-CSS2`.

[BM72] R. Bayer und E. McCreight. Organization and Maintenance of Large Ordered Indexes. *Acta Informatica 1*, 3:173–189, 1972.

[BN96] M. Bichler und S. Nusser. Developing Structured WWW-Sites with W3DT. In *Proceedings of WebNet 96 World Conference on the WWW and Internet (WebNet'96)*, San Francisco, California, USA, 1996.

[Boo94] G. Booch. *Objektorientierte Analyse und Design.* Addison-Wesley, Bonn, Paris [u.a.], 1994.

[BPSM98] T. Bray, J. Paoli und C. Sperberg-McQueen. *Extensible Markup Language (XML) 1.0.* World Wide Web Consortium (W3C), W3C Recommendation, 1998.
URL: `http://www.w3c.org/TR/REC-xml`.

[Bra95] P. De Bra. Finding Information on the Web. *CWI quarterly*, 8(4):289–306, Dezember 1995.

[Bra98] N. Bradley. *The XML Companion.* Addison-Wesley, Harlow, England, 1998.

[BRJ96] G. Booch, J. Rumbaugh und I. Jacobson. *The Unified Modeling Language for Object-Oriented Development.* Rational Software Corporation, Santa Clara, CA, 1996.

[Bus45] V. Bush. As We May Think. *The Atlantic Monthly*, Juli 1945.

[CG88] B. Campbell und J.M. Goodman. HAM: A general purpose hypertext abstract machine. *Communications of the ACM*, 31(7):856–861, 1988.

[CGMH+94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman und J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of the 10th Meeting of the Information Processing Society of Japan (IPSJ)*, Tokyo, Japan, 1994.

[Che76] P. Chen. The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, Marz 1976.

[CHRS98] A. Christiansen, M. Höding, C. Rautenstrauch und G. Saake. *Oracle8 effizient einsetzen.* Addison-Wesley, Bonn, Reading, Massachusetts [u.a.], 1998.

[CINS97] T. Catarci, L. Iocchi, D. Nardi und G. Santucci. Conceptual Views over the Web. In *Proceedings of the 4th Workshop Knowledge Representation Meets Databases (KRDB-97)*, Athens, Greece, 1997.

[Con87] J. Conklin. Hypertext: An Introduction and Survey. *IEEE Computer*, 20(9):17–41, September 1987.

[CSP98] S. Christodoulou, G. Styliaras und T. Papatheodorou. Evaluation of Hypermedia Application Development and Management Systems. In *HyperText 98*, Pittsburgh, 1998.

[CVW95] A. Clausnitzer, P. Vogel und S. Wiesener. A WWW interface to the OMNIS/Myriad literature retrieval engine. *Computer Networks and ISDN systems*, 27:1017–1026, 1995.

[CZ95] D. Chapman und E. Zwicky. *Building Internet Firewalls.* O'Reilly & Associates, Inc., USA, 1995.

[Dat95] C. Date. *An Introduction to Database Systems.* Addison-Wesley Publishing Company, Bonn, Reading, Massachusetts [u.a.], 1995.

[Dea99] S. Deach. *Extensible Stylesheet Language (XSL) Specification.* World Wide Web Consortium (W3C), 1999.
URL: `http://www.w3c.org/TR/1999/WD-xsl-19990421`.

[DG97] O. Duschka und M. Genesereth. Infomaster - An Information Integration Tool. In *Proceedings of the International Workshop on Intelligent Information Integration*, Freiburg, Deutschland, September 1997.

[DH95] W. Dalitz und G. Heyer. *Hyper-G, das Internet-Informationssystem der 2. Generation.* dpunkt.verlag, Heidelberg, 1995.

[DHW97] M. Dörr, H. Haddouti und S. Wiesener. The German National Bibliography 1601-1700: Digital Images in a Cooperative Cataloging Project. In *Proceedings of IEEE International Forum on Research and Technology Advances in Digital Libraries (ADL'97)*, Washington, D.C., Mai 1997. IEEE Computer Society.

[DI95] A. Diaz und T. Isakowitz. RM Case: Computer-Aided Support for Hypermedia Design and Development. In *Proceedings of the 1995 International Workshop on Hypermedia Design (IWHD'95)*, 1995.

[Dic97] H. Dicken. *JDBC Internet-Datenbank-Anbindung mit Java.* Internat. Thomson Publ., Bonn, 1997.

[Dig] Digital Equipment. *Alta Vista (tm) Search.*
URL: `http://www.altavista.digital.com`.

[Dit98] K. Dittrich. Datenbanktechnologie im Umbruch: neue Anforderungen - Funktionalität - neue Architekturen. In *Euroforum-Konferenz Datenbank 2000, Datenbankmanagement und -technik im Wandel*, Freising, Deutschland, Juli 1998.

[ED99] C. Enguix und J. Davis. Filling the Gap: New Models for Systematic Page-based Web-Application Development & Maintenance. In *International Workshop Web Engineering '99*, Toronto, Canada, 1999.

[EN94] R. Elmasri und S. Navathe. *Fundamentals of Database Systems.* The Benjamin/Cummings Publishing Company, Inc., Redwood City, California, second. Auflage, 1994.

[FFK+97] M. Fernandez, D. Florescu, J. Kang, A. Levy und D. Suciu. STRUDEL: A Web-site Management System. In *Proceedings of the ACM SIGMOD international conference on Management of data (SIGMOD'97)*, Tucson, Arizona, 1997.

[FGM+97] R. Fielding, J. Gettys, J. Mogul, H. Frystyk und T. Berners-Lee. *Hypertext Transfer Protocol - HTTP/1.1*, Januar 1997.
URL: `http://www.ics.uci.edu/pub/ietf/http`.

[FGN98] G. Falquet, G. Guyot und L. Nerima. Language and Tools to Specify Hypertext Views on Databases. In *International Workshop on the Web and Databases (WebDB'98)*, Valencia, Spain, 1998.

[FGP96] G. Falquet, J. Guyot und I. Prince. Generating Hypertext Views on Databases. In *Proceedings of the 14th INFORSID Conference*, Bordeaux, 1996.
URL: `http://cuiwww.unige.ch/db-research/hyperviews/reports/index.html`.

[FHN97] P. Fröhlich, N. Henze und W. Nejdl. Meta-Modeling for Hypermedia Design. In *Second IEEE Computer Metadata Conference (Meta-Data'97)*, Silver Spring, Maryland, 1997.

[FLM98] D. Florescu, A. Levy und A. Mendelzon. Database Techniques for the World Wide Web: A Survey. *ACM SIGMOD Records*, 27(3):59–64, 1998.

[FP98] P. Fraternali und P. Paolini. Model-Driven Development of Web Applications: the Autoweb System. In *Proceedings of the VI International Conference on Extending Database Technology (EDBT'98)*, Valencia, Spain, 1998.

[FWM97] T. Fiebig, J. Weiss und G. Moerkotte. RAW: A Relational Algebra for the Web. In *Proceedings of 16th ACM Symposium on Principles of Database Systems (PODS'97)*, Tucson, Arizona, 1997.

[GM95] A. Gupta und I. Mumick. Maintenance of Materialized Views: Problems, Techniques, and Applications. *Data Engineering Bulletin*, 18(2):3–18, Juni 1995.

[GMP95] F. Garzotto, L. Mainetti und P. Paolini. Hypermedia Design, Analysis, and Evaluation Issues. *Communications of the ACM Special Issue*, 38(8):74–86, August 1995.

[GMS93] A. Gupta, I. Mumick und V. Subrahmanian. Maintaining Views Incrementally. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, D.C., 1993.

[GPS93] F. Garzotto, P. Paolini und D. Schwabe. HDM - A Model-Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems*, 11(1):1–26, Januar 1993.

[GR93] J. Gray und A. Reuter. *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann Publishers, Inc., San Mateo, USA, 1993.

[Gra99] T. Gray. *The Annotated XML 1.0 Specification*, 1999.
URL: `http://www.xml.com`.

[Gun96] S. Gundavaram. *CGI programming on the World Wide Web.* O'Reilly, Sebastopol, 1996.

[Hal88] F. Halasz. Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM*, 31(7), 1988.

[HB98a] D. Harvey und S. Beitler. *The Developer's Guide to Oracle Web Application Server 3.* Addison-Wesley Longman Verlag GmbH, Bonn, Reading, Massachusetts [u.a.], 1998.

[HB98b] G.-J. Houben und P. De Bra. Generating Hypermedia Applications for Volatile Database Output. In *Proceedings of WebNet 98 World Conference on the WWW and Internet (WebNet'98)*, Orlando, Florida, USA, 1998.
URL: `http://wwwis.win.tue.nl/~houben/pub`.

[HB99] G.-J. Houben und P. De Bra. Retrieval of Volatile Database Output Through Hypermedia Applications. In *Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences (HICSS-32)*, Hawaii, 1999.

[Heu92] A. Heuer. *Objektorientierte Datenbanken: Konzepte, Modelle, Systeme.* Addison-Wesley, Bonn, München, Paris[u.a.], 1992.

[Höf96] G. Höfling. *Schema-Evolution in objekt-orientierten Datenbanksystemen.* Dissertation, Fakultät für Informatik, Technische Universität München, München, 1996.

[HLU98] U. Herrmann, D. Lenz und G. Unbescheid. *Oracle8 für den DBA.* Addison-Wesley Longman Verlag GmbH, Bonn, Reading, Massachusetts [u.a.], 1998.

[HR99] T. Härder und E. Rahm. *Datenbanksysteme, Konzepte und Techniken zur Implementierung.* Springer-Verlag, Berlin, Heidelberg, 1999.

[HS94] F. Halasz und M. Schwartz. The Dexter Hypertext Reference Model. *Communications of the ACM*, 37(2):30–39, Februar 1994.

[Hum92] R. Hums. *Effiziente Integritätssicherung in deduktiven Datenbanksystemen durch Logikprogrammtransformation.* Dissertation, Fakultät für Informatik, Technische Universität München, München, 1992.

[IKK97] T. Isakowitz, A. Kamis und M. Koufaris. Extending RMM: Russian Dolls and Hypertext. In *Proceedings of the 30th Annual Hawaii International Conference on System Sciences (HICSS-30)*, Hawaii, 1997.

[Inf99] Informix. *Creating Web-Enabled Database Applications Using the Informix Web DataBlade Module*, 1999.
URL: `http://www.informix.com/informix/whitepapers/index.html`.

[Ink] Inktomi Cooperation. *HotBot (tm).*
URL: `http://www.hotbot.com`.

[IRB95] T. Isakowitz, E. Rossi und P. Balasubramanian. RMM: A methodology for structured hypermedia design. *Communications of the ACM Special Issue*, 38(8):34–44, August 1995.

[Jan00] Eshref Januzaj. Implementierung eines Meta-Modells zur dynamischen Generierung datenbankgestützter Internet-Anwendungen. Diplomarbeit, Technische Universität München, 2000.

[Jep97] B. Jepson. *Java Database Programming.* John Wiley & Sons, New York. Chichester, Brisbane [u.a.], 1997.

[Kem96] J. Kempe. *Das Delta-Transaktionsmodell zur Unterstötzung von Nicht-Standard-Anwendungen.* Dissertation, Fakultät für Informatik, Technische Universität München, Aachen, Shaker Verlag, 1996.

[KLM+97] A. Kawaguchi, D. Lieuwen, I. Mumick, D. Quass und K. Ross. Concurrency Control Theory for Deferred Materialized Views. In *International Conference on Database Theory (ICDT'97)*, Delphi, Greece, 1997.

[KS91] H. Korth und A. Silberschatz. *Database System concepts.* McGraw Hill, New York, 1991.

[KS95] D. Konopnicki und O. Shmueli. A Query System for the World-Wide Web. In *Proceedings of the 21th International Conference on Very Large Data Bases (VLDB'95)*, Zürich, Switzerland, 1995.

[KS98] D. Konopnicki und O. Shmueli. Bringing Database Functionality to the WWW. In *International Workshop on the Web and Databases (WebDB98)*, Valencia, Spain, 1998.

[Lem95] L. Lemay. *Web Publishing mit HTML.* Markt und Technik, München, 1995.

[Loe97] H. Loeser. Datenbankanbindung an das WWW. In *Datenbanksysteme in Büro, Technik und Wissenschaft (BTW '97)*, Ulm, 1997.

[Loe98] H. Loeser. Techniken für Web-basierte Datenbankanwendungen: Anforderungen, Ansätze, Architekturen. *Informatik Forschung und Entwicklung*, 13(4):196–216, 1998.

[LPJ+95] C. Liu, J. Peek, R. Jones, B. Buus und A. Nye. *Internet Server.* O'Reilly/International Thomson Verlag GmbH, Bonn, 1995.

[LPVV99] B. Ludäscher, Y. Papakonstantinou, P. Velikhov und V. Vianu. View Definition and DTD Inference for XML. In *Post-ICDT Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats (SSD'99)*, Jerusalem, Januar 1999.

[LRO96] A. Levy, A. Rajaraman und J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB'96)*, Bombay, India, 1996.

[LS99] O. Lassila und R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification.* World Wide Web Consortium (W3C), Februar 1999.
URL: `http://www.w3c.org/TR/1999/REC-rdf-syntax-19990222`.

[LSS96] L. Lakshmanan, F. Sadri und I. Subramanian. A Declarative Language for Querying and Restructuring the World-Wide-Web. In *Proceedings of the 6th*

International Workshop on Research Issues in Data Engineering (RIDE'96), New Orleans, 1996.

[MAM+98] G. Mecca, P. Atzeni, P. Merialdo, A. Masci und G. Sindoni. From Databases to Web-Bases: The ARANEUS Experience. Technical Report 34, Dipartimento di Informatica e Automazione, Universita' di Roma Tre, Mai 1998.

[Mau96] H. Maurer. *Hyperwave*. Addison-Wesley, Harlow, Reading, Menlo [u.a.], 1996.

[Mic97] Microsoft. *ISAPI Programming, Microsoft Interactive Developer*, 1997. URL: `http://www.microsoft.com/Mind/0197/ISAPI.htm`.

[Mih96] George Andrei Mihaila. WebSQL - An SQL-like Query Language for the World Wide Web. Diplomarbeit, University of Toronto, 1996.

[MM97] A. Mendelzon und T. Milo. Formal Models of Web Queries. In *Proceedings of the Sixteenth ACM SIGMOD Intern. Symposium on Principles of Database Systems (PODS'97)*, Tucson, Arizona, Mai 1997.

[MM99] P. Marden und E. Munson. Why Current Style Sheet Standards Have Failed to Improve Document Engineering. In *International Workshop Web Engineering '99*, Toronto, Canada, Mai 1999.

[MMM96] A. Mendelzon, G. Mihaila und T. Milo. Querying the World Wide Web. In *Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems (PDIS'96)*, Miami, Florida, 1996.

[MMM98] G. Mecca, Alberto Mendelzon und P. Merialdo. Efficient Queries over Web Views. In *Proceedings of the VI International Conference on Extending Database Technology (EDBT'98)*, Valencia, Spain, 1998.

[MV98] U. Masermann und G. Vossen. Suchmaschinen und Anfragen im World Wide Web. *Informatik Spektrum*, 21(1):9–15, 1998.

[Net] Netscape. *Netscape API Functions*. URL: `http://home.netscape.com/servers/index.html`.

[NN95] J. Nanard und M. Nanard. Hypertext Design Environments and the Hypertext Design Process. *Communications of the ACM*, 38(8):49–56, August 1995.

[NS96] T. Nguyen und V. Srinivasan. Accessing Relational Databases from the World Wide Web. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, 1996.

[O2 96] O2 Technology. *O2 Web User Manual, Release 4.6*, 1996.

[Ope96] Open Market. *FastCGI: A High-Performance Web Server Interface*, 1996. Technical White Paper
URL: `http://fastserv.name.net/whitepapers/fcgi-whitepaper.shtml`.

[Ora96] Oracle. *WebServer 3.0 Transactions: Bringing the Enterprise to the Web*, 1996. White Paper.

[PM96] R. Pönighaus und J. Mitlöhner. Relational Databases and the World Wide Web: Automatic Generation of Hypertext based on Reverse-engineered Meta Information. In *Proceedings of WebNet 96 World Conference on the WWW and Internet (WebNet'96)*, San Francisco, 1996.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy und W. Lorensen. *Objektorientiertes Modellieren und Entwerfen.* Coedition der Verlage Carl Hanser und Prentice Hall Int., München, 1993 (Deutsche Ausgabe der Originalausgabe von 1991).

[Sch97] F. Schanda. Sichere und zuverlässige Transaktionsverarbeitung über das Internet. In *Online'97*, Hamburg, Deutschland, 1997.

[Sin98] G. Sindoni. Incremental Maintenance of Hypertext Views. In *Proceedings of the Workshop on the Web and Databases (WebDB'98)*, Valencia, Spain, 1998.

[SJ96] M. Staudt und M. Jarke. Incremental Maintenance of Externally Materialized Views. In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB'96)*, Bombay, India, 1996.

[SK97] U. Sommer und J. Kempe. Java löst Entwicklungsschub aus. *Business Online*, 10:53–58, 1997.

[SLHS93] J. Schnase, J. Leggett, D. Hicks und R. Szabo. Semantic Data Modeling of Hypermedia Associations. *ACM Transactions on Information Systems*, 11(1):27–50, Januar 1993.

[SP99] D. Schwabe und R. De Almeida Pontes. A Method-based Web Application Development Environment. In *International Workshop Web Engineering '99*, Toronto, Canada, 1999.

[Spe98] G. Specht. *Multimedia-Datenbanksysteme: Modelle-Architekturen-Retrieval.* München, 1998. Habilitationsschrift, Technische Universität München.

[SR95] D. Schwabe und G. Rossi. The Object-Oriented Hypermedia Design Model. *Communications of the ACM Special Issue*, 38(8):1–26, August 1995.

[SS83] G. Schlageter und W. Stucky. *Datenbanksysteme: Konzepte und Modelle.* Teubner Studienbücher, Stuttgart, 1983.

[Ste95] L. Stein. *How to set up and maintain a World Wide Web Site.* Addison-Wesley Publishing Company, Inc., Reading Massachusetts [u.a.], 1995.

[Sun99] Sun Microsystems. *The JDBC(tm) Database Access API*, 1999. URL: `http://java.sun.com/products/jdbc`.

[SW88] J. Smith und S. Weiss. An Overview of Hypertext. *Communications of the ACM*, 31(7), 1988.

[SZ97] U. Sommer und P. Zoller. Online-Datenbanken. In Klaus-Peter Boden und Michael Barabas, Hrsg., *Internet – von der Technologie zum Wirtschaftsfaktor, Deutscher Internetkongress 97*, Düsseldorf, dpunkt.verlag, 1997.

[SZ99] U. Sommer und P. Zoller. WebCon: Design and Modeling of Database driven Hypertext Applications. In *Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences (HICSS-32)*, Hawaii, 1999.

[Tra96] Transaction Software GmbH, München. *Transbase Relational Database System Version 4.3.3 - System and Installation Guide*, 1996.

[Tro98] O. De Troyer. Designing Well-Structured Websites: Lessons to Be Learned from Database Schema Methodology. In *Proceedings of the 17th International Conference on Conceptual Modeling (ER'98)*, Singapore, November 1998. Lecture notes in computer science, Vol 1507, Springer, 1998.

[Ull88] J. Ullman. *Principles of Database and Knowledge-Base Systems.* Volume 1, Computer Science Press, 1988.

[Urm97] S. Urman. *Oracle 8 PL/SQL Programming.* Osborne McGraw-Hill, Berkeley, New York, St. Louis [u.a.], 1997.

[Vos94] G. Vossen. *Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme.* Addison-Wesley Longman Verlag GmbH, Bonn, Reading, Massachusetts [u.a.], 1994. Second Edition.

[Wie98] S. Wiesener. *SemaLink.* Dissertation, Fakultät für Informatik, Technische Universität München, Shaker Verlag, Aachen, 1998.

[Wor98a] World Wide Web Consortium (W3C). *Query for xml: position papers*, 1998. URL: `http://www.w3c.org/TandS/QL/QL98/pp.html`.

[Wor98b] World Wide Web Consortium (W3C). *w3c´s dom web page*, 1998. URL: `http://www.w3c.org/DOM`.

[Zol96] P. Zoller. *Anbindung von Datenbanksystemen an das World Wide Web.* Diplomarbeit, Technische Universität München, 1996.

[ZS98] P. Zoller und U. Sommer. WebCon: A Toolkit for an Automatic, Data Dictionary Based Connection of Databases to the WWW. In *Proceedings of the ACM Symposium on Applied Computing (SAC'98)*, Atlanta, Georgia, 1998.