Untersuchungen zu Datentyperweiterungen für XML-Dokumente und ihre Anfragemethoden am Beispiel von DB2 und Informix

Diplomarbeit

Universität Rostock, Fachbereich Informatik

vorgelegt von: Beate Porst
geboren am: 17. April 1970 in Neuruppin

Betreuer: Prof. Dr. Andreas Heuer
Dr. Ing. Holger Meyer
Dr. Ing. Meike Klettke
Dr. Ing. Antje Düsterhöft

Bearbeitungszeitraum: 01.11.1998 - 30.04.1999

Zusammenfassung

Abstract

CR-Klassifikation

Key Words

XML, objekt-relationale Datenbanken, semistrukturierte Daten

Inhaltsverzeichnis

Kapitel 1

Einleitung

Mit der weltweiten Zunahme an elektronisch publizierten Dokumenten wird Datenbanken zur Speicherung, Verwaltung und Manipulation dieser Dokumente eine größer werdende Aufmerksamkeit gewidmet. Für zahlreiche Anwendungen, wie Digitale Bibliotheken, Online-Nachschlagewerke etc., ist insbesondere die Informationsgewinnung aus diesen Dokumenten ein wichtiges Kriterium. Volltext-Retrieval-Systeme stellen hier eine populäre Möglichkeit zur Informationsextraktion dar. Neben reinen Textdateien existiert eine Dokumentenkategorie, die eine implizite Struktur aufweist, die aber aufgrund ihrer Heterogenität nicht durch ein Datenbankschema beschrieben werden kann. Das Problem traditioneller Volltext-Retrieval-Systeme ist nun, daß der Schwerpunkt ihrer Anfragen auf der reinen Inhaltssuche beruht. Anfragen, die hingegen die Struktur sowie eine Kombination aus Struktur und Inhalt betreffen, werden nicht unterstützt, obwohl hier ein großes Potential zur Verbesserung der Retrieval-Qualität liegt.

Die beiden Projekte und GETESS und Global-Info (siehe Abschnitt 1.1 und 1.2) befassen sich unter anderem mit dem oben genannten Problem des Text-Retrievals in semistrukturierten Dokumenten. Da die in dieser Diplomarbeit konzipierte Datentyperweiterung auch ein Ansatz für diese Projekte sein könnte, sollen die Ziele und Aufgaben der Projekte (im speziellen der Teilprojekte der Universität Rostock) hier kurz vorgestellt werden.

1.1 GETESS

Gegenstand des BMBF[1]-Forschungsprojektes GETESS (*GE*rman *T*ext *E*xploitation and *S*earch *S*ystem) ist die Entwicklung eines intelligenten Werkzeuges zur Informationsrecherche im Internet.

[1] Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie

Aufgrund der Mächtigkeit der Informationsmenge im World Wed Web (WWW) wird es immer schwieriger, gezielt Informationen zu finden. Suchmaschinen wie AltaVista arbeiten bei ihrer Suche größtenteils mit syntaktischen Methoden, deren Beschränkungen jedoch zu unbefriedigenden Ergebnissen führen. Das Projekt GETESS widmet sich nun dieser Problematik, indem es zur Verbesserung der Ergebnisse des Information Retrieval die Semantik von WWW-Dokumenten mit Hilfe eines eingeschränkten natürlich-sprachlichen Prozessor extrahiert und aus diesen Informationen *abstracts* bildet. Die GETESS-Suchmaschine wird dann in der Lage sein, neben den herkömmlichen Informationen auch die in einer Datenbank gespeicherten Abstracts mit in die Suche einzubeziehen und, falls erforderlich, zwischen den Abstracts Beziehungen herzustellen.

Das Gesamtsystem GETESS besteht aus folgenden Kernteilen:

Dialogsystem: Das Dialogsystem soll sowohl die natürlich-sprachliche Interaktion mit dem Nutzer unterstützen, als auch formale Anfragen, die in der Regel von erfahrenen Anwendern bevorzugt werden.

Suchmaschine: Kernpunkt des GETESS-Systems bildet eine Suchmaschine, die neben herkömmlichen Informationen, in der Lage sein muß, Informationen aus Datenbanken mit in die Suche einzubeziehen. Diese Suchmaschine weist ein typische Gatherer-Broker-Struktur auf. In der Gatherer-Phase des Suchprozesses wird aus den analysierten Internetinformationen ein Suchindex gebildet, sowie die Erzeugung der Abstracts vorgenommen. Im Broker-Prozeß werden die Nutzeranfragen durch das Dialogsystem auf Anfragen der IRQL[2] (Information Retrieval Query Language) abgebildet. Sie ermöglicht eine Kombination von Datenbank- und Information Retrieval-Anfragen.

Natürlich-sprachliche Verarbeitung: Die natürlich-sprachliche Verarbeitung wird zum einen zur Analyse der Nutzeranfragen innerhalb des Dialogsystems verwendet, Zum anderen bildet sie die linguistische Grundlage zur Extraktion von Informationen aus den Abstracts, sowie der Generierung natürlich-sprachlicher Antworten aufbauend auf den Fakten der Abstract-Datenbank.

Ontologie: Die Ontologie repräsentiert ein semantisches Referenzmodell einer bestimmten Anwendungsdomäne, wodurch eine inhaltliche Einordnung der Suchanfrage ermöglicht wird und zudem Kontextinformationen bereitgestellt werden können.

Die Aufgabe der Universität Rostock im Gesamtprojekt GETESS ist die Realisierung der Suchmaschine. Darin enthalten ist die Konzeption einer verteilten

[2]Die IRQL wird derzeit am Lehrstuhl Datenbank- und Informationssystem entwickelt. Sie soll als Basissprache die Lücke zwischen IR und DB-Anfragen schließen.

Datenbanklösung zur Verwaltung der Abstracts, der Wissensbasis und der Lexika sowie die Entwicklung einer Sprache für den einheitlichen Zugriff auf strukturierte und unstrukturierte Informationen (IRQL).

Die Untersuchungen der Möglichkeiten der Datentyperweiterungen, die im Rahmen dieser Diplomarbeit vorgenommen wurden, könnten hier den erster Ansatzpunkt hinsichtlich der IRQL in GETESS bilden.

1.2 Global-Info

Das Ziel des Förderkonzeptes "Globale Elektronische und Multimediale Informationssysteme" (Global-Info) des BMBF ist die Schaffung einer digitalen Bibliothek, die den Wissenschaftlern in Deutschland einen effizienten Zugang zu den weltweit vorhandenen elektronischen und multimedialen Informationen ermöglicht. Diese Informationen können in verteilten Informationssystemen digital gespeichert sein. Bei der Gestaltung der Strukturen wird insbesondere Wert auf das Zusammenwirken aller am Prozeß der Bereitstellung und Nutzung von Information und Dokumenten beteiligten Partner (Produzenten, Distributeure, Konsumenten) gelegt [Sch98].

Vom Global-Info Consortium wurden gemeinsame Grundsätze und die inhaltlichen Schwerpunkte für die Gestaltung des Projektes formuliert. Die fünf Schwerpunkte des Projektes sind nachfolgend aufgeführt.

1. Ergänzung und Bearbeitung von Inhalten: Dokumententypen, Verfahren und Werkzeuge für elektronisches Publizieren, Transfer, Speicherung, Konvertierung, Indizierung
2. dVernetzung von Lehr- und Lernmaterialien
3. Formale Beschreibung, Identifikation und Retrieval, Metadaten, Vernetzung
4. Nutzung von Inhalten: Alterting, Awareness, Informationsverbände, Informationsvermittlung, Evaluierung von Ergebnissen, Oberflächen, Intelligente Agenten, Paßwortproblematik
5. Wirtschaftlichkeitsmodell, Billing und Abrechnung, Statistik

Die Einteilung des Projektes in die verschiedenen Schwerpunkte dient dabei nur der Gliederung und Organisation der Arbeit und nicht der inhaltlichen Abgrenzung. Vielmehr sind die Übergänge zwischen den Schwerpunkten fließend bzw. sie überlappen sich.

Die Beteiligung der Universität Rostock am Projekt Global-Info erfolgt über die Sonderfrodermaßnahme BlueRose, deren Zielsetzung und Einordnung in das Gesamtprojekt im folgenden Unterabschnitt erläutert werden.

1.2.1 Die Sonderfördermaßnahme BlueRose

Die Universität Rostock ist mit den Fachbereichen Informatik, Mathematik, Chemie und Physik sowie der Universitätsbibliothek und dem Rechenzentrum ein Projektpartner der BlueRose (*B*uilding *L*ibraries *U*nifying *E*nhanced *R*etrieval-*O*riented user *SE*rvices) Sonderfördermaßnahme. Sie befaßt sich mit der Entwicklung von Retrieval-orientierten Diensten zur Nutzung elektronischer Literatur durch Wissenschaftler, Bibliothekare und Studenten. Die konkreten Ziele von BlueRose sind [Heu98]:

- eine einheitliche Benutzungsoberfläche für alle hier entwickelten Nutzer-Dienste
- die Vernetzung von (heterogenen) Dokumenten und anderen Informationen wie Tagungen, Autoren, Forschungsgruppen, Zeitschriftenreihen
- Integration verschiedener Techniken für Anfrage- und Retrieval-Funktionalitäten
- Bereitstellung aktueller, adaptierbarer, lokaler Sichten auf große verteilte Dokumentenbestände
- Benachrichtigung von Benutzern über Änderungen in den Fachinformationen im passiven und aktiven Fall
- Retrieval-Unterstützung durch fachspezifische, mehrsprachige Erschließungswerkzeuge

Aufgrund der Zielsetzung, ist BlueRose im wesentlichen dem Schwerpunkt 4 zuzuordnen. Abbildung 1.1 zeigt die Rahmenarchitektur des Projektes Global-Info. Farblich abgehoben ist dabei der Bereich, der durch die Zielsetzung von BlueRose abgedeckt wird.

Die Ziele, die im Rahmen von BlueRose durch die Universität Rostock verfolgt werden, sind zum einen die *Verwaltung semistrukturierter Datenbanken* zur Aufnahme von Volltexten und Metadaten über Dokumente. Diese Datenbanken sollen als lokale integrierte Dokumentenserver (LDS) in elektronischen Bibliotheken fungieren. Desweiteren wird durch die Integration von semistrukturierten Dokumenten und strukturierten Metadaten ein kombinierter Information-Retrieval und Anfrage-Prozeß impliziert. Dieser Prozeß erfordert geeignete Kopplungen für *verteilte Anfragen und verteiltes Retrieval*. Daneben sollen *Import-Funktionalitäten* für föderierte Datenbestände und Kopplungen des Anfrage- und

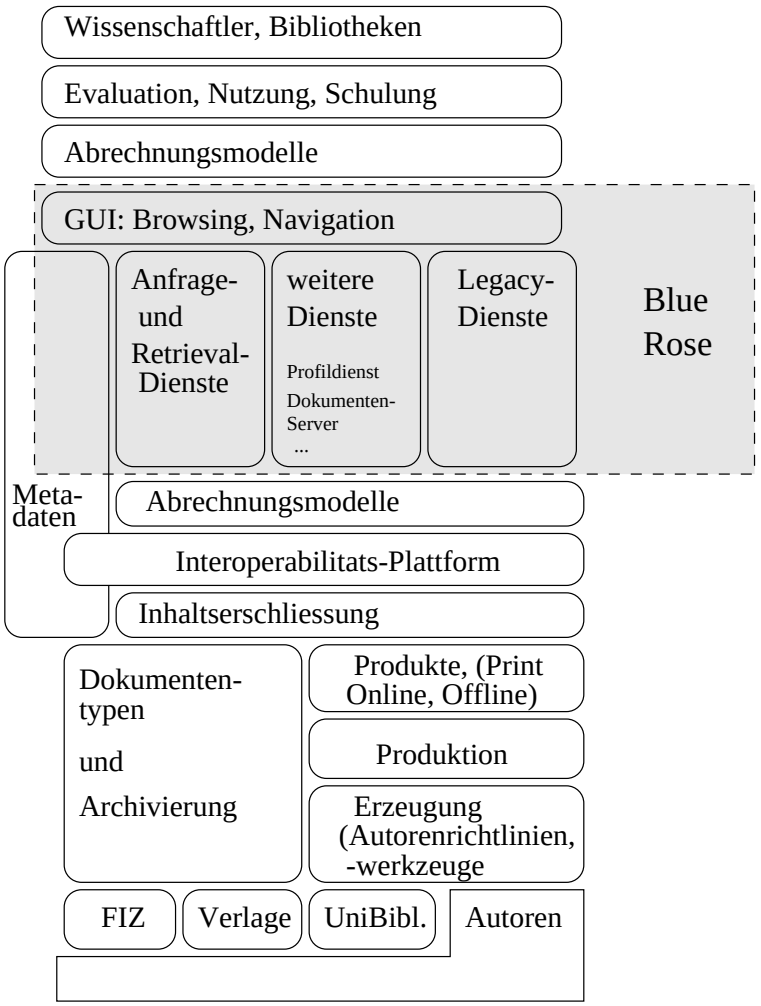

Abbildung 1.1: Einordnung von BlueRose in die Global-Info Rahmenarchitektur

Retrieval-Dienstes an Literatur-Recherche-Werkzeuge und Suchmaschinen wie MeDoc und InterDoc entwickelt werden.

Der Bezug zwischen den Untersuchungen zu Datentyperweiterungen im Rahmen dieser Arbeit und den Aufgaben der Universität Rostock im Global-Info Projekt wird durch das zu entwickelnde Modul des Lokalen Dokumenten Servers (siehe Abbildung 1.2) hergestellt. Dieses Modul dient der Aufnahme lokal zu verwaltender semistrukturierter Dokumente sowie strukturierter Informationen. Die zu speichernden Dokumente besitzen dabei eine implizite XML-Struktur, die aufgrund ihrer Semantik eine gute Möglichkeit zur Integration von Datenbanken und Information Retrieval bietet. Zur Realisierung des LDS wurde in BlueRose eine Kombination aus objektrelationalem und Volltext-Datenbanksystem

vorgeschlagen. Aufgrund der Erweiterbarkeit der objektrelationalen Datenbanksysteme wäre hier die Entwicklung eines XML-Datentyps auf Basis vorhandener Volltexterweiterungen denkbar.

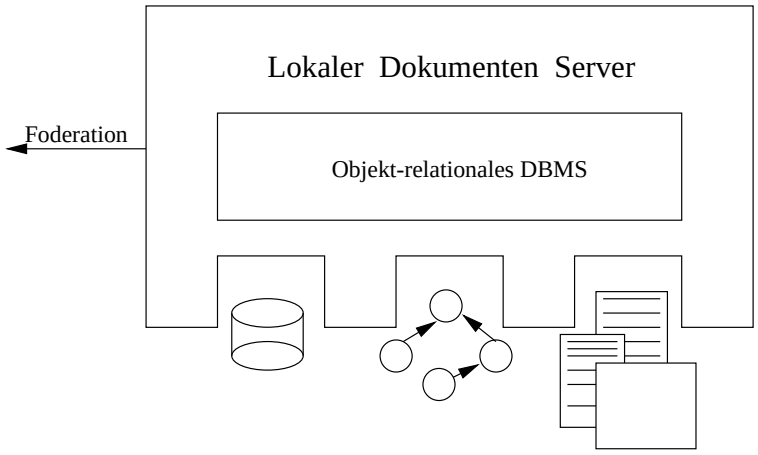

Abbildung 1.2: Lokaler Dokumenten Server (LDS)

1.3 Ziel der Arbeit

Aufbauend auf Untersuchungen der Volltext-Retrieval-Erweiterungen der Datenbanksysteme DB2 und Informix soll für beide Systeme eine Konzeption erarbeitet werden, die es ermöglicht, XML-Dokumente darzustellen, zu verwalten und mit Anfragemethoden finden zu können. Desweiteren wird für eines der beiden Systeme eine prototypische Implementierung vorgestellt. Ein weiteres Ziel der Diplomarbeit bildet der perspektivische Vorschlag eines XML-Datentyps, mit dem auf "ideale" Weise XML-Dokumente dargestellt, verwaltet und angefragt werden können.

1.4 Aufbau der Arbeit

Im weiteren Verlauf der Arbeit werden zunächst die Grundlagen zu *semistrukturierten Daten* und der Beschreibungssprache *XML* gegeben (Kapitel 2 und 3). Daran anschließend wird in Kapitel 4 die allgemeine Volltext-Retrieval-Funktionalität der Datenbanksysteme DB2 und Informix untersucht und einander gegenübergestellt. In diesem Kapitel soll sich zunächst nur die Leistungsfähigkeit im Allgemeinen herauskristallisieren. Kapitel 5 steht unter der Fragestellung, inwieweit sich die Volltexterweiterungen der Datenbanksysteme für das Information

Retrieval von semistrukturierten Dokumenten eignen und welche Erweiterungen auf diesen Systemen realisierbar sind. Für das Datenbanksystem DB2 wird neben der konzeptionellen Erweiterung die prototypische Realisierung vorgestellt. Da die Erweiterung eines bestehenden Volltext-Retrieval-Systems erwartungsgemäß nicht unproblematisch sein kann, wird in Kapitel 6 eine Konzeption eines "idealen" XML-Datentyps vorgestellt, der aufgrund seiner Funktionen gleichermaßen Anfragen über die Struktur als auch den Inhalt unterstützt. Zum Abschluß der Arbeit werden die wichtigsten Punkte noch einmal zusammengefaßt und ein Ausblick auf zukünftige Arbeiten gegeben.

Kapitel 2

Semistrukturierte Dokumente

Die Menge an elektronisch zur Verfügung stehenden Daten ist in den letzten Jahren dramatisch angestiegen. Die Palette reicht von unstrukturierten Daten in Dateisystemen zu hochstrukturierten Daten in Datenbanksystemen. Eine weitere Art, die man speziell im Zusammenhang mit dem World Wide Web antrifft, sind semistrukturierte Daten.

Im weiteren Verlauf diese Kapitels werden zunächst die Eigenschaften semistrukturierter Daten beschrieben sowie einige Datenmodelle für diese Dokumentenart vorgestellt (Abschnitt 2.1 und 2.2). Im Anschluß daran wird die Eignung dieser Datenmodelle hinsichtlich der Verwaltung und Anfrage von XML-Dokumenten aus dem Bereich "Digitale Bibliotheken" bewertet.

2.1 Eigenschaften semistrukturierter Daten

In der kurzen Einführung wurde noch nicht verdeutlicht, worin sich semistrukturierte Dokumente von anderen Dokumenten abheben. Dies soll nachfolgend anhand der charkteristischen Merkmale aus [Abi97] vorgenommen werden.

- Die Struktur der Daten ist unregelmäßig

  Bei der Verwaltung einer Menge von semistrukturierten Dokumenten kann nicht davon ausgegangen werden, daß die Strukturelemente aller Dokumente homogen sind. Vielmehr können Elemente unvollständig sein oder zusätzliche Informationen beinhalten. Entsprechend kann auch die gleiche Information durch unterschiedliche Typen repräsentiert werden (Bsp. Preise in Euro oder Mark).

- Das Schema ist implizit in den Daten enthalten

Semistrukturierte Dokumente besitzen in der Regel eine präzise Struktur, die implizit enthalten ist.

Mit Hilfe von Werkzeugen zur Strukturerkennung können die Strukturinformationen und die zwischen den Strukturelementen enthaltenen Beziehungen extrahiert werden. Die Möglichkeiten dieser Beziehungen gehen über die Fähigkeiten von Standarddatenbankmodellen hinaus, so daß die Interpretation speziellen Applikationen obligt.

- Die Struktur ist unvollständig

  Ein kaum lösbares Ziel ist die Strukturierung sämtlicher Daten, da diese vielfach einen sehr differenzierten Strukturierungsgrad aufweisen. Mit Hilfe von Information Retrieval Werkzeugen kann zwar eine eingeschränkte Strukturerkennung erfolgen, jedoch wird es immer Dokumente geben, die aus Datenbanksicht unstrukturiert sind.

- Das Schema ist flexibel

  In Standarddatenbanken wird eine strenge Typisierung angewendet. Jedes Attribut/Objekt des Datenbankschemas besitzt einen im voraus festgelegten Datentyp, wodurch nicht konforme Update-Operationen auf diesen Attributen/Objekten abgelehnt werden. Für viele Anwendungen ist diese strenge Typisierung jedoch zu einschränkend. Im Lore Projekt ([AQMW97]) wird daher zur flexiblen Typisierung ein *data guide* verwendet. Dieser enthält zwar Informationen über den gegenwärtigen Typ; dieser muß jedoch nicht korrekt sein. Im Unterschied zu Update-Operationen einer Datenbank, werden hier alle Daten akzeptiert.

- Das Schema ist relativ groß

  Aufgrund der Heterogenität der Dokumente ist das resultierende Schema, im Verhältnis zur Menge der Daten relativ groß. Dies steht im Widerspruch zur Theorie relationaler Datenbanken, da hier davon ausgegangen wird, daß das Schema in Dimensionen kleiner ist, als die zu speichernden Daten. Die Konsequenz ist, daß der Nutzer die Details der Struktur im Allgemeinen nicht kennt und dadurch Anfragen über das Schema die gleiche Bedeutung erhalten, wie Anfragen über Daten.

- Das Schema unterliegt häufigen Änderungen

  In Standarddatenbanken wird davon ausgegangen, daß auf einem erzeugten Schema im weiteren Verlauf kaum Änderungen vollzogen werden, da diese in der Regel mit erheblichem Aufwand verbunden sind. Man betrachtet es daher auch als unveränderbar. Im Zusammenhang mit semistrukturierten Daten wird jedoch von einer sehr flexiblen Struktur ausgegangen, die häufigen Änderungen unterliegen kann. Diese Tatsache ist auch der Grund,

warum bei der Verwaltung dieser Daten oft keine Datenbank verwendet wird.

- Der Unterschied zwischen Struktur und Daten ist unscharf

  In Standarddatenbanken wird grundsätzlich zwischen dem Schema (beschreibt die Struktur der Daten) und den Daten (stellen Instanzen dar) unterschieden. Im Zusammenhang mit semistrukturierten Daten verschwimmen diese Unterschiede. Hier können aufgrund der relativ häufigen Schemaänderungen Daten zu Schemainformation werden und umgekehrt Schemainformationen zu Daten.

Auch bei den, im Zusammenhang mit dem Global Info Projekt zu verwaltenden XML-Volltexten handelt es sich um semistrukturierte Daten. Das Schema ist hier implizit im Dokument enthalten. Dieses kann aber in gewissen Grenzen, die durch die Document Type Definition (siehe Kapitel 3) festgelegt werden, variieren.

2.2 Datenmodelle für semistrukturierte Daten

Aufgrund der Merkmale des letzten Abschnittes eigenen sich Datenbankmodelle im Allgemeinen nicht zur Integration heterogener und wenig strukturierter Daten, da sie, mangels Flexibilität ein festes und im voraus festgelegtes Schema erwarten. Treten Verletzungen dieses Schemas durch Update-Operationen auf, wird die Aufnahme dieser Daten in das Datenbanksystem verweigert. Diese Tatsachen waren der Grund für die Vorstellung neuer Integrationsmodelle, in denen weder Typen noch Schemata existieren.

2.2.1 Schemalose Datenmodelle

Der schemalose Ansatz ist ein populärer Weg zur Modellierung semistrukturierter Daten. In diesen Modellen wird kein vorher festgelegtes Schema zur Repräsentation der Daten gefordert, wodurch die Integration heterogener semistrukturierter Dokumente möglich wird. Die benötigte Flexibilität der Modelle wird durch die Darstellung der Daten als Bäume oder Graphen erreicht, in denen neue Daten durch hinzufügen eines Knotens integriert werden können. Nachfolgend werden zwei Vertreter schemaloser Datenmodell vorgestellt.

2.2.1.1 Object Exchange Model (OEM)

Daten im Object Exchange Model ([AQMW97]) werden durch einen Graphen $G = (V,E)$ repräsentiert, wobei E die Menge der Kanten und V die Menge der

Knoten ist. Jede Kante des Graphen ist durch ein Label vom atomaren Typ *string* gekennzeichnet. Die Knoten des Graphen enthalten Objekte. Jedes Objekt besitzt eine eindeutige Objektidentität vom Typ *oid*. Alle Objekte des Graphen sind entweder atomar oder komplex. Werte eines atomaren Objektes müssen von einem atomaren Datentyp (*integer, real, string etc.*) sein. Die Werte eines komplexen Objektes ist die Menge der Objektreferenzen die von diesem Objekt ausgehen (bezeichnet durch die Menge der *(label, oid)* Paare). Abbildung 2.1 zeigt ein Beispiel eines OEM-Graphen, der mit Hilfe der Vorschriften der Buch-DTD[1] (siehe Anhang A.1) gebildet wurde. Die Buch-DTD[2] legt eine gewisse Struktur der Dokument fest, die jedoch zu viel Flexibilität für eine vernünftige Abbildung auf ein Datenbankschema bietet. Durch die Heterogenität können Strukturelemente semistrukturierter Dokumente unvollständig sein. Dies zeigt sich unter anderem in der Titelstruktur, die im Objekt &13 mit Ober- und Untertitel (Objekte &14 und &77), im Objekt &36 hingegen nur mit einem Obertitel (Objekt &49) angegeben wurde. Die Inhomogenität der beiden Buch-Objekte wird jedoch deutlicher durch die unterschiedliche Struktur der beiden Body-Objekte (Objekte &23 und &11) zum Ausdruck gebracht.

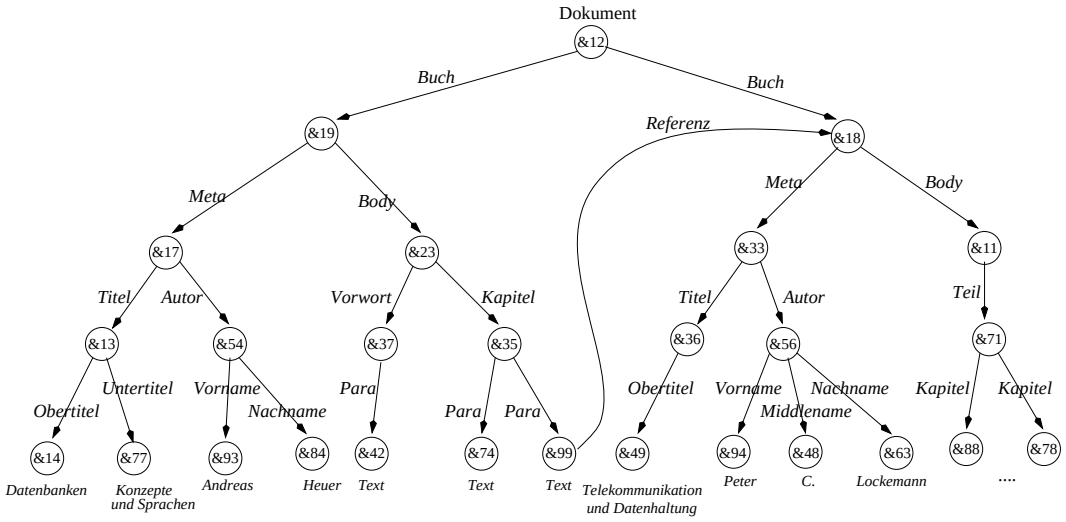

Abbildung 2.1: Beispiel eines OEM-Graphen

2.2.1.2 Das Baum-Modell

Das hier vorgestellte Baum-Datenmodell entstammt [BDS95]. In der vereinfachten Form beschreibt das Modell einen Baum mit markierten Kanten. Die

[1] Aus Platzgründen wurden nur einige Elemente ausgewählt.
[2] Die Begriffserklärung für eine DTD wird in Kapitel 3 abgehandelt.

Beschreibung der Kanten ist vom Typ *label*, der eine Vereinigung verschiedener Basis-Datentypen darstellt. Ein markierter Baum dieses Modells wird dann durch die rekursive Gleichung

$$tree = \mathcal{P}_{fin}\ (label \times\ tree)$$

repräsentiert, wobei $\mathcal{P}_{fin}(S)$ eine endliche Untermenge von S beschreibt.

Die Erstellung eines Baumes erfolgt mit Hilfe der folgenden Konstruktoren:

$$\begin{array}{lcl} \phi & : & tree \\ \{\_ \Rightarrow \_\} & : & label \times tree \rightarrow tree \\ \_ \cup \_ & : & tree \times tree \rightarrow tree \end{array}$$

Bäume des Modells sind Mengen, so daß ein leerer Baum durch die leere Menge (ϕ) dargestellt wird. Der Ausdruck $\{l \Rightarrow t\}$ beschreibt einen Baum, dessen Wurzel eine mit l markierte Kante in Richtung des Unterbaumes t besitzt. Die dritte Konstruktionsregel erzeugt durch die Verschmelzung der Wurzeln der Bäume der rechten und linken Seite einen neuen Baum. In Abbildung 2.2 wird an Hand eines Beispiels die Vereinigung durch das Verschmelzen der Wurzeln der Teilbäume gezeigt.

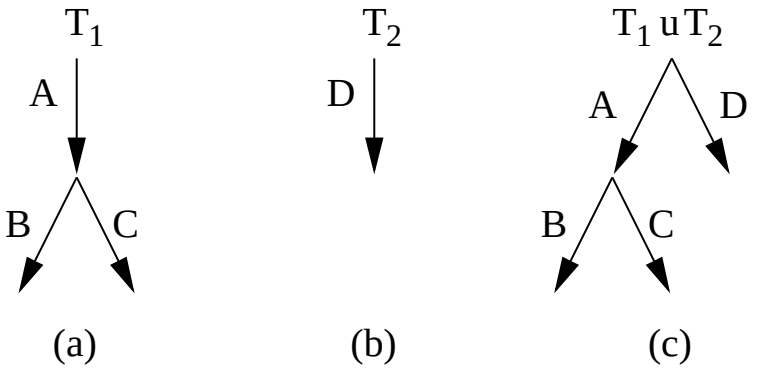

Abbildung 2.2: Beispiel der Vereinigung zweier Teilbäume

In [BDS95] wurde neben dem einfachen Baummodell eine Erweiterung vorgestellt, die auch Zyklen zwischen den Kanten des Baumes zuläßt (*rooted edgle-labeled graph*).

Vorteile der schemalosen Modelle: Der Hauptvorteil dieser Modelle ist ihre Einfachheit und ihre Flexibilität.

Nachteile der schemalosen Modelle: Anfragen müssen dynamisch ausgeführt werden. In traditionelle Datenbanken wird eine zweistufige Anfrageverarbeitung durchgeführt. Zunächst wird mit Hilfe der Schemainformationen eine Ausführungsplan generiert. In der zweiten Phase wird ohne zusätzlichen Zugriff auf die Schemainformationen der Plan ausgeführt. In einem Modell ohne Schema ist dies nicht realisierbar. Hier müssen sämtliche Informationen dynamisch gefunden werden. Im schlechtesten Fall ist dazu eine Untersuchung der gesamten Datenbank erforderlich.

2.2.2 Das Union-Datenmodell

Der Ansatz der Union-Typen entstammt [Clu97]. Um die Flexibilität zu erreichen, die für die Verwaltung semistrukturierte Daten erforderlich ist, werden hier Union-Datentypen einem objkt-orientierten Datenbanksystem hinzugefügt. Die Frage der Typsicherheit bei Verwendung von Unions wird in dem Modell durch Bezeichung der Unions erreicht, die dann einen expliziten Zugriff auf die instantiierten Komponenten des Union erfordern und dadurch eine statische Typprüfung erlauben.

Formal werden die Union-Typen durch die nachfolgenden Regeln beschrieben.

1. Sind $\tau_1, ..., \tau_n$ von einander verschiedene Typen und $a_1, ..., a_n$ von einander verschiedene Attributnamen, dann ist $(a_1 : \tau_1 + ... + a_n : \tau_n)$ ein Union-Typ.
2. $dom(a_1 : \tau_1 + ... + a_k : \tau_k) = \{[a_i : v_i] | 1 \leq i \leq k \wedge v_i \in dom(\tau_i)\}$.
3. - if $\forall i \in [1..n], \exists j \in [1..m] : a_i = b_j \wedge \tau_i \leq \tau_j$
   $(a_1 : \tau_1 + ... + a_n : \tau_n) \leq (b_1 : \tau'_1 + ... + b_m : \tau'_m)$
   - if $\exists j \in [1..m] : a_i = b_j \wedge \tau_i \leq \tau_j$
   $[a_i : \tau_i] \leq (b_1 : \tau'_1 + ... + b_m : \tau'_m)$

Das Symbol zwischen den Typen $\leq$ repräsentiert eine Untertypbeziehung. Aufgrund dessen besagt Regel 3 beispielsweise, daß das Tupel $[a_i : \tau_i]$ einen Untertyp zu $(b_1 : \tau'_1 + ... + b_m : \tau'_m)$ darstellt, falls im Union-Typ ein Attributname existiert, der dem Attributnamen des Tupels entspricht und deren korrespondierende Typen in einer Ober-Untertypbeziehung stehen.

Die Verwendung der Union-Typen soll anhand des nachfolgenden Beispiels verdeutlicht werden.

Beispiel:

```
...
<!ELEMENT Body ( (Vorwort, (teil | kapitel+))
                        | (teil | kapitel+) ) >
...
```

Das Beispiel zeigt einen sehr kleinen Ausschnitt aus der Buch-DTD (siehe Anhang A.1). Die Struktur des *Body*-Elementes beinhaltet entweder

1. ein Vorwort gefolgt von einem *teil* oder einem oder mehreren *kapiteln*

oder

2. ein *teil* oder ein oder mehrere *kapitel*.

Diese verschiedenen Alternativen können mit Hilfe der eingeführten Union-Typen folgendermaßen dargestellt werden.

union (a_1 : tuple (vorwort : string,
union (teil : Teil, kapitel : list(Kapitel)))
a_2 : union (teil : Teil, kapitel : list(Kapitel)))

Werden innerhalb einer Anfrage Variablen verwendet, die auf einem Union-Typen operieren, wird eine implizite Selektion der entsprechenden Komponente des Union-Typen durchgeführt.

Vorteile des Modells: Der Hauptvorteil des Ansatzes ist, daß die Implementierung nur wenige Modifikationen des Datenbanksystems erfordert

Nachteile des Modells: Es ist nicht gesichert, ob Union-Typen die Lösung für alle Probleme der Integration heterogener Daten darstellt. Da vor jedem Zugriff überprüft werden muß, welche Komponenten des Union instantiiert sind, kann die intensive Nutzung der Unions zu einem ernstzunehmenden Leitungsabfall des Systems führen.

2.3 Bewertung und Wahl eines Modells

In diesem Abschnitt soll die Frage geklärt werden, durch welches Datenmodell die Strukturen und Operationen der im Rahmen dieser Arbeit zu verwaltenden semistrukturierten Dokumente sinnvoll repräsentiert werden können. Bei der Beurteilung sind dabei nicht nur die Vor- und Nachteile der betrachteten Modelle heranzuziehen, sondern auch, ob das Datenmodell auf das konkret zum Einsatz kommende Datenbanksystem abgebildet werden kann. Durch die Aufgabenstellung sind bereits Richtlinien hinsichtlich des verwendeten Datenbanksystems gesetzt. Sowohl Informix Universal Server als auch DB2 Universal Database stellen objekt-relationale Datenbanksysteme dar.

1. **Object Exchange Modell** : Die Abbildung dieses Modells auf ein objekt-relationales Modell ist durchführbar, womit die benötigte Flexibilität hinsichtlich der Integration heterogener Dokumente erreicht wird. Jedoch sind

spätere Rekonstruktionen von Dokumenten und Browsing-Anfragen auf den Strukturen des objekt-relationalen Modells kaum realisierbar. Eine zweite Möglichkeit ist die Erhaltung der semistrukturierten Dokumente in ihrem Orginalzustand. Für Operationen auf den Dokumenten werden die Dokumente mit Hilfe von Wrappern in das Graphenmodell transformiert. Diese Variante kann aber zu erheblichen Leistungseinbußen führen, da für jede Operation zunächst eine Transformation durchzuführen ist.

2. **Baum-Modell** : Die Ausführungen zum Object Exchange Modell treffen im wesentlichen auch auf das Baum-Modell zu.

3. **Union-Datenmodell** : Union-Typen, in der im Abschnitt 2.2.2 genannten Form sind in keinem der beiden objekt-relationale Modelle vorhanden. Sie könnten durch Nullwerte und Integritätsbedingungen simuliert werden. In Informix wäre auch die Modellierung durch eine Typ-Hierarchie möglich. Beide Varianten stellen aber keine adequate Lösung dar. Zum einen birgen Nullwert generell Probleme (beispielsweise werden bei Verbunden mögliche Tupel nicht in das Ergebnis aufgenommen), zum anderen können durch die Modellierung der Strukturen der DTD als Attribute von Relationen keine Browsing-Anfragen realisiert werden. Hierfür müßte die Struktur in einer Anfrage der Relationenalgebra vernachlässigt werden können.

Für das Datenmodell wird eine hybride Lösung vorgeschlagen, die zum einen aus strukturierten Anteilen, aber auch aus den semistrukturierten Daten bestehen wird. Da der Anwendungsbereich auf die Verwaltung und Anfrage semistrukturierter Daten aus dem Bereich "Digitale Bibliotheken" beschränkt ist, sind bestimmte Informationen der Dokumente, auch aufgrund der Festlegungen der DTD, notwendig. Diese Informationen können durch spezielle Attribute des objekt-relationalen Datenmodells repräsentiert werden. Die als semistrukturierte Dokumente vorliegenden Volltexte werden ebenfalls durch ein Attribut dargestellt, jedoch wird für Information Retrieval-Anfragen eine Wrapper-Funktion eingesetzt, die eine Transformation der Dokumente von der Darstellung als Character Large Object (CLOB) in ein Baum-Modell durchführt. Durch die Transformation können Anfragen über die implizite Struktur und den Inhalt realisiert werden. Das im OEM-Modell angeführte Argument der Leistungseinbußen durch Wrapper-Einsatz tritt hier nicht in dem Maße auf, da nur IR-Anfragen auf Struktur und Inhalt eine Transformation erfordern.

Kapitel 3

XML - Extensible Markup Language

XML ist ein durch das World Wide Web Consortium (W3C) entwickeltes Dokumentenaustauschformat. Trotz der oberflächlichen Ähnlichkeit zur Hypertext Markup Language (HTML) besteht zwischen beiden ein fundamentaler Unterschied. HTML ist eine spezielle Anwendung der Standard Generalized Markup Language (SGML [Gol90]), die zur Auszeichung von Dokumenten dient. XML stellt dagegen eine Teilmenge von SGML dar, die wie SGML nur die Vorschriften bereitstellt eine Auszeichnungssprache zu definieren. Die Notwendigkeit der Entwicklung einer Teilmenge von SGML bestand aufgrund der Komplexität, die das größte Hindernis von Anwendungen mit SGML darstellt.

Ein XML-basiertes Dokument besitzt sowohl eine logische als auch eine physische Struktur (siehe Abbildung 3.1). Die logische Struktur erlaubt eine Einteilung des Dokumentes in benannte Abschnitte und Unterabschnitte, sogennannte *Elemente*. Durch die physische Struktur kann ein XML-Dokument aus einer oder mehreren Speicherungseinheiten, sogenannten *Entities*, bestehen. Dadurch können Informationen wiederverwendet und nicht XML-Daten (wie z.B. Bilder) durch Referenzen in das Dokument eingefügt werden.

Im Kontrast zu HTML beschreibt XML nicht zwingendermaßen das Layout eines Dokumentes wohl aber dessen Verhalten. Somit handelt es sich um eine Metasprache, deren Ziel die Beschreibung anderer Sprachen ist. Es existieren keine vordefinierten Elemente, so daß die Benennung und Verwendung sich nach den Vorstellungen des Entwicklers richtet. Um jedoch eine möglichst gute Kontrolle über die logische Dokumentenstruktur zu gewährleisten, existiert ein optionaler Mechanismus, der es erlaubt, Elemente in einer speziellen Dokumentenklasse zu spezifizieren (Document Type Definition - DTD).

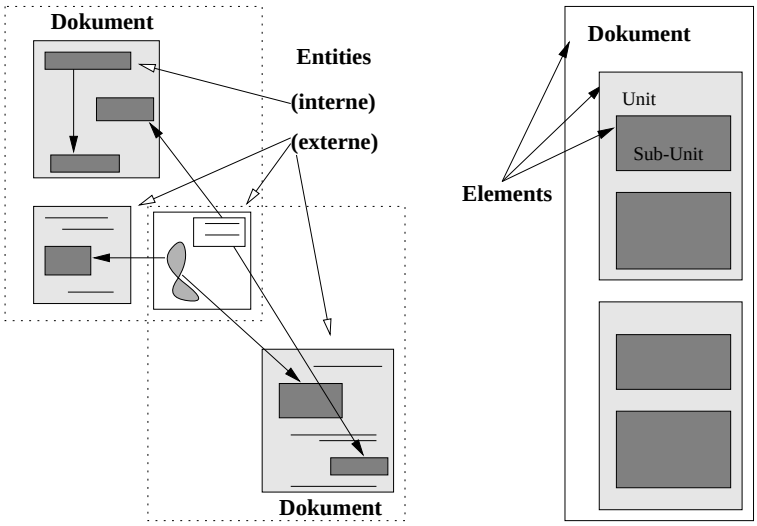

Abbildung 3.1: Physische vs. logische XML-Struktur [Bra98]

3.1 Document Type Definition (DTD)

Eine DTD ist durch eine formale Grammatik gekennzeichnet, die eine bestimmte XML-Sprache definiert. In ihr werden die Namen der Elemente (Tags) und ihre Schachtelung definiert (logische Struktur). Die Schachtelung wird erreicht, indem zu jedem Element angegeben wird, welche weiteren Elemente in seinem Inhalt auftreten dürfen. Die Strukturierung der Elemente kann durch nachfolgende logische Kombination von Elementen erreicht werden, die der Notation regulärer Ausdrücke ähnelt.

Sequenz:
(A , B) — *A* und *B* müsser in der angegebenen Reihenfolge aufgeführt werden

Alternative:
(A | B) — Hier muß entweder *A* oder *B* angegeben werden.

Wiederholung:
A* — A kann null oder beliebig oft nacheinander angegeben werden.
A+ — A muß mindestens einmal oder beliebig oft nacheinander angegeben werden.
A? — A kann null oder genau einmal angegeben werden.

Jede DTD enthält eine Menge von Deklarationen die eine Dokumenten-Baumstruktur definieren. Dieser Baum steht in einer bestimmten Beziehung zu allen Dokumenten (Instanzen), die dieser DTD entsprechen. Neben Elementen werden

in einer DTD Attribute spezifiziert, die die dazugehörigen Elemente näher beschreiben. Beispiel 3.1 zeigt einen Ausschnitt aus der Diplomarbeit-DTD (siehe Anhang A.2).

Beispiel 3.1
<!– Diplomarbeit DTD –>
<!ELEMENT Diplomarbeit (Titel, Bearbeitungszeit?)>
<!ATTLIST Diplomarbeit Sprache xml:lang NMTOKEN "de">

<!ELEMENT Titel (Obertitel, Untertitel?)>
<!ELEMENT Obertitel (#PCDATA) >
<!ELEMENT Untertitel (#PCDATA) >
<!ELEMENT Bearbeitungszeit (Vom, Bis)>
<!ELEMENT Vom (#PCDATA) >
<!ELEMENT Bis (#PCDATA) >

Die Deklaration der einzelnen Elemente der DTD beginnt mit `<!ELEMENT` gefolgt von einem eindeutigen Elementnamen. Anschließend wird die Grammatik des Elementes entsprechend der zuvor aufgeführten logischen Kombination notiert. Attribute eines Elementes werden gesondert beginnend mit `<!ATTLIST` deklariert. Die Beispiel-DTD besteht somit aus den Elementen *Diplomarbeit, Titel, Obertitel, Untertitel, Bearbeitungszeit, Vom* und *Bis*. Das Element *Diplomarbeit* stellt das Wurzelelement dieser DTD dar. Seine innere Struktur wird durch die Sequenz aus *Titel* und *Bearbeitungszeit* vorgegeben, wobei die Angabe der Bearbeitungszeit optional ist. Zusätzlich wurde für dieses Element ein Attribut spezifiziert, welches die verwendete Sprache festlegt. Die Angabe von #PCDATA in der Deklaration des Elementes *Obertitel* verweist darauf, daß der Inhalt ausschließlich Text (*Parsed Character Data*) enthalten darf.

3.2 Konformität

Bei der Prüfung der Korrektheit eines XML-Dokumentes wird zwischen den Begriffen *Wohlgeformtheit* und *Gültigkeit* unterschieden. Ein wohlgeformtes Dokument enthält mindestens ein oder mehrere Elemente, die durch ein Start- und End-Tag-Paar begrenzt werden und korrekt ineinander verschachtelt sind. Weiterhin existiert genau ein Wurzelelement, von dem kein Teil im Inhalt eines anderen Elementes enthalten ist. Als Konsequenz daraus gilt, daß für jedes Element k, das nicht die Wurzel ist, ein anderes Element v existiert, so daß sich k im Inhalt von v befindet, aber nicht im Inhalt eines anderen Elementes, das sich im Inhalt von v befindet.

1. v heißt Vater von k.

2. k heißt Kind von v.

Desweiteren darf in einem wohlgeformten Dokument kein Attributname mehr als einmal in einem Element verwendet werden und der Attributwert darf keinen Verweis auf ein externes Entity oder eine öffnende spitze Klammer enthalten. Ein gültiges XML-Dokument hingegen darf die logische Struktur seiner DTD nicht verletzen. Das bedeutet, daß ein gültiges Dokument keine Verletzungen der in [BPSM98] enthaltenen Gültigkeitsbeschränkungen aufweisen darf. Jedes gültige Dokument ist auch wohlgeformt. Beispiel 3.2 zeigt eine gültige und damit auch wohlgeformte Instanz der DTD aus Beispiel 3.1. Es ist insofern gültig, daß der Name der Dokumenttyp-Deklaration (DOCTYPE) mit dem Namen des Wurzelelementes korrespondiert und keine Gültigkeitsbeschränkung der XML-Spezifikation durch die Instanz verletzt wird.

Beispiel 3.2

```
<!DOCTYPE Diplomarbeit SYSTEM "da.dtd">
<Diplomarbeit Sprache="de">
  <Titel>
    <Obertitel>
      Untersuchungen zu Datentyperweiterungen für XML-Dokumente und
      ihre Anfragemethoden am Beispiel von DB2 und Informix
    </Obertitel>
  </Titel>
  <Bearbeitungszeit>
    <Vom>
      01.11.1998
    </Vom>
    <Bis>
      30.04.1999
    </Bis>
  </Bearbeitungszeit>
</Diplomarbeit>
```

Die Korrektheit von Dokumenten wird mit Hilfe von XML-Prozessoren überprüft. Konforme Prozessoren werden in *validierend* und *nicht-validierend* kategorisiert, wobei in Abhängigkeit dieser Kategorie auf Wohlgeformtheit oder Gültigkeit getestet wird.

Validierende Prozessoren müssen Verletzungen der Beschränkungen, die durch Deklarationen in der DTD formuliert werden, sowie jedes Nicht-Erfüllen der in der XML-Spezifikation formulierten Gültigkeitsbeschränkugen melden. Zur Erreichung dieses Ziels müssen sie die gesamte DTD und alle extern analysierten Entities, die im Dokument referenziert werden einlesen und verarbeiten.

Nicht-validierende Prozessoren müssen lediglich ein Dokuement einschließlich der internen DTD-Teilmenge auf Wohlgeformtheit prüfen. Dazu müssen sie alle Deklarationen einer internen DTD-Teilmenge und interne Parameter-Entities lesen und verarbeiten.

Kapitel 4

Evaluierung der Retrieval-Funktionalität von DB2 und Informix

In der Einführung dieser Arbeit wurde bereits auf die zunehmende Bedeutung von Datenbanken zur Speicherung, Verwaltung und Manipulation von multimedialen Dokumenten hingewiesen. Diese Datenbanken stellen jedoch neben der Forderung, große Datenmengen (CLOB[1], BLOB[2]) speichern zu können auch neue Herausforderungen an das Anfragesystem, da hier die strukturierten Anfragemöglichkeiten zur gezielten Informationsextraktion auf diesen Datenmengen versagen. Eine getrennte Anfrage zwischen den strukturierten Datenbankanteilen und den unstrukturierten Datenmengen über ein externes Retrieval-Werkzeug ist jedoch inakzeptabel. Einige Datenbankhersteller haben diesem Problem bereits Rechnung getragen und ihre Datenbanksysteme dahingehend erweitert, daß es möglich ist, neue Datentypen und Funktionen in das System zu integrieren, die Aufgaben ausführen, die zuvor durch externe Werkzeuge realisiert wurden. Durch diese Erweiterungen sind kombinierte Anfrage aus strukturierten und unstrukturierten Datenbankanteilen realisierbar.

Der überwiegende Teil diese Kapitels widmet sich den beiden Datenbanksystemen DB2 und Informix und im speziellen ihren Volltext-Erweiterungen. Diese Erweiterungen, basierend auf neuen Datentypen und zugehörigen Funktionen, sind integraler Bestandteil der Datenbanksysteme und bieten somit die Möglichkeit der kombinierte Anfrage über strukturierte und unstrukturierte Datenbankanteile.

Im weiteren Verlauf wird zunächst eine kurze Erklärung der wichtigsten Begriffe des Information Retrieval gegeben. Anschließend wird sowohl für das Datenbanksystem DB2 als auch für Informix die Funktionalität der jeweiligen Volltext-

[1] Character Large Object
[2] Binary Large Object

Retrieval-Erweiterung eingehend vorgestellt. Den Abschluß des Kapitels bildet dann eine Gegenüberstellung der beiden Systeme hinsichtlich ihrer Retrieval-Leistungsfähigkeit.

4.1 Begriffsklärung

Retrieval-Modelle: Information-Retrieval befaßt sich mit den Konzepten der Repräsentation und der Interpretation der Struktur von Informationen. Information-Retrieval selbst ist ein Kommunikationsprozeß, der auf der Eingabeseite aus der Analyse von Daten durch Verfahren der Wissensrepräsentation und deren Überführung in gespeichertes Wissen besteht und auf der anderen Seite Informationen liefert, die aufgrund von Transformationen aus der internen Wissensstruktur gewonnen werden.

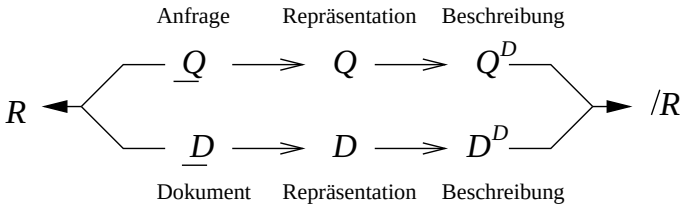

Abbildung 4.1: Konzeptionelles Modell für Retrieval

[Fuh97] verwendet das in Abbildung 4.1 dargestellte Modell als grundlegendes konzeptionelles Modell für Information-Retrieval. Dabei bezeichnet $\underline{D}$ die Menge der Dokumente in der Datenbasis und $\underline{Q}$ die Menge der Anfragen an das Retrieval-System. Zwischen den Dokumenten und den Anfragen besteht die Relevanzbeziehung R. Die Dokumentenrepräsentation D ist die semantische Sicht auf Dokumente. Die formalisierten Anfragen werden als Fragerepräsentation Q bezeichnet. Zum Zwecke des Retrievals werden die Repräsentationen in Beschreibungen (D^D, Q^D) überführt. Fragebeschreibungen sind beispielsweise boolesche Ausdrücke, Dokumentenbeschreibungen können gewichtete oder ungewichtete Indexierungen sein. Durch eine Retrieval-Funktion wird die Beschreibung von Frage-Dokument-Paaren verglichen und daraus ein Retrievalgewicht berechnet (im allgemeinen eine reelle Zahl). Die Erstellung der Beschreibungen und die Definition einer Retrieval-Funktion hängt vom jeweiligen Retrieval-Modell ab. Die nachfolgenden Unterpunkte geben eine kurze Beschreibung der wichtigsten Retrieval-Modelle.

Boolean Retrieval: Boolean-Retrieval ist das bekannteste und historisch erste Retrieval-Modell. Der Anwender formuliert seinen Informations-

bedarf als eine logische Kombination von Schlüsselworten, die durch die logischen Operatoren *AND*, *OR* und *NOT* verbunden werden. Aufgrund der binären Gewichtung der Terme kann die Retrieval-Funktion nur die Retrieval-Gewichte 0 und 1 liefern. Daraus resultiert eine Zweiteilung der Dokumente der Datenbasis in gefundene und nicht gefundene Dokumente.

- Vorteile des Modells:
  - effizient zu implementieren
  - Mächtigkeit — mit einer booleschen Anfrage kann jede beliebige Teilmenge von Dokumenten aus einer Datenbasis selektiert werden
- Nachteile des Modells:
  - Größe der Antwortmenge ist schwer zu kontrollieren
  - keine Ordnung der Antwortmenge nach mehr oder weniger relevanten Dokumenten, diese kann nur simuliert werden (beispielsweise durch Bestimmung der Vorkommenshäufigkeiten des Suchterms)
  - keine Gewichtung der Frageterme
  - strenge Trennung in gefundene und nicht gefundene Dokumente
  - schwierige Frageformulierung

Fuzzy Retrieval: Zur Überwindung einiger Nachteile des booleschen Retrievals wurde das Fuzzy-Retrieval vorgeschlagen, deren Grundlage die Theorie der Fuzzy-Logik bildet. Im Unterschied zum booleschen Retrieval können den Attributen gewichtete Indexierungen zugeordnet werden, wodurch die Retrieval-Funktion Werte im Intervall [0,1] liefert. Damit ergibt sich nun eine Rangordnung der Antwortdokumente.

- Vorteile des Modells:
  - Rangordnung der Antwortdokumente
- Nachteile des Modells:
  - keine Fragetermgewichtung
  - schwierige Frageformulierung
  - schlechte Retrievalqualität im Verhältnis zum Vektorraummodell

Vektorraummodell: Das Vektorraummodell wurde ursprünglich im Rahmen des SMART-Projektes [Sal71] entwickelt. Im Vektorraummodell werden Dokumente und Fragen als Punkte in einem Vektorraum aufgefaßt, der durch die Terme der Datenbasis aufgespannt wird. Beim

Retrieval wird dann nach solchen Dokumenten gesucht, deren korrespondierende Vektoren ähnlich dem Fragevektor sind. Für den Vektorraum wird dabei vorausgesetzt, daß alle Term-Vektoren orthogonal und normiert sind.

- Vorteile des Modells:
  - anschauliches Modell aufgrund der geometrischen Interpretation
  - benutzerfreundlich aufgrund einfacher Frageformulierung
  - Möglichkeit der Verbesserung der Retrieval-Qualität durch Relevance-Feedback
- Nachteile des Modells:
  - Modell enthält sehr viele Heuristiken[3]

Ranking: Ranking ist ein Rangordnungssystem, das jedem Dokument einen Meßwert zuordnet, der die Güte des Dokumentes zum Frageterm widerspiegelt. Für die oben beschriebenen Retrieval-Modelle hat das Ranking nur für Fuzzy-Retrieval und das Vektorraummodell Bedeutung. Aufgrund der strengen Trennung der Dokumente beim Boolean-Retrieval existiert hier keine Rangordnung. Diese könnte lediglich nachträglich durch Bestimmung der Häufigkeiten der Attributwerte erzeugt werden.

Relevance-Feedback: Bei der Begriffsbeschreibung des Vektorraummodells wurde als ein Vorteil gegenüber Boolean und Fuzzy-Retrieval die Möglichkeit der Verbesserung der Retrieval-Qualität durch Relevance-Feedback genannt. Beim Relevance-Feedback möchte man durch Relevance-Beurteilung der qualitativ besten Dokumente einer Anfrage die ursprünglichen Fragetermgewichte so verändern, daß sich ein verbesserter Fragevektor ergibt. Das Retrieval kann dann mit dem verbesserten Fragevektor erneut durchgeführt werden. Dieser Prozeß (Relevance-Berurteilung, Retrieval) kann auch iterativ erfolgen.

Indexierung: Ein Information-Retrieval wird in der Regel auf sehr großen Datenbeständen durchgeführt. Die Indexierung der Dokumente eines IR-Systems ist daher ein wichtiger Faktor, der sich trotz des Mehraufwandes an Speicher und Update-Kosten positiv auf die Performance bei der Lokalisierung der relevanten Dokumente einer Suchanfrage auswirkt. In [Alm98] werden verschiedene Indizierungsformen vorgestellt. Einige dieser Indizierungsarten werden sich in den nachfolgend beschriebenen Retrieval-Erweiterungen der Datenbanksysteme wiederfinden.

[3]Nachteil des Modells ist, daß durch den heuristische Ansatz zur Berechnung der Indexierungsgewichte, die Dokumentenrepräsentationen nur schlecht erweitert werden können. Hierzu müssen erst geeignete Gewichtungsformeln durch Experimente gefunden werden.

4.2 IBM DB2 Universal Database

DB2 Universal Database (UDB) ist das derzeit jüngste Mitglied der DB2 Produktfamilie. Der Name Universal Database soll dabei die Vielfalt an unterstützten Systemplattformen und Einsatzbereichen signalisieren. Die nachfolgenden Punkte stellen einige der in [Cha98] genannten Eckpfeiler der UDB dar:

- Unterstützung unterschiedlichster Hard- und Software-Umgebungen
  UDB Server existieren für Windows NT, OS/2 und viele Unix-basierte Systeme, einschließlich AIX, Solaris und HP-UX. UDB Clients sind für Window95, Windows3.1 und Macintosh-Systeme lieferbar.
- Unterstützung zweier unabhängiger Arten von Parallelität (symmetrisches Multiprozessing und massiv parallele "shared nothing" Konfigurationen)
- Sprachunterstützung für C, C++, Java, FORTRAN, COBOL und REXX für die Einbettung von SQL-Statements in Applikationen
- Unterstützung von Industriestandards wie ODBC[4] und SQL92[5]
- Graphische Schnittstellen für die interaktive Datenbanknutzung und zur Datenbankadministration
- Bereitstellung von erweiterten Datentypen und zugehörigen Funktionen

In der Produktspezifikation klassifiziert IBM die UDB als objekt-relationales Datenbanksystem. Die Datenbankfunktionalität stützt sich im wesentlichen auf relationale Konzepte, die durch die Möglichkeit der Integration nutzerdefinierter Datentypen[6] und Funktionen[7] erweitert wurden. Bei der Definition von neuen Datentypen besteht jedoch die Einschränkung, daß jeder neue Datentyp von einem Standard-Datentyp abgeleitet werden muß und nicht komplex strukturiert sein darf.

Neben der Option, entsprechend des Anwendungsfalles eigene Datentypen zu entwickeln, werden durch die UDB erweiterte Datentypen und Methoden für die verschiedenen multimedialen Bereiche (Audio, Video, Image, Text) bereitgestellt.

Im folgenden Abschnitt soll die Funktionalität der Erweiterung für den Bereich Text und Textrecherche eingehender betrachtet werden. Zu diesem Zweck wurden die im Produkthandbuch [IBM97] beschriebenen Retrieval-Funktionalitäten durch diverse Testanfragen evaluiert. Insofern durch die Analyse des Systems Abweichungen vom beschriebene Verhalten in [IBM97] registriert wurden, wird dies in den entsprechenden Unterpunkten zur Funktionsbeschreibung dargestellt.

[4] Open Database Connectivity
[5] Structured Query Language
[6] user defined datatypes (UDT)
[7] user defined functions (UDF)

4.3 DB2 TextExtender

Der TextExtender in der Version 5.2 ist ein Mitglied der DB2 Extender-Familie. Er stellt eine Erweiterung dar, die es ermöglicht, innerhalb einer strukturierten SQL-Anfrage Operationen zur Volltextsuche zu spezifizieren. Diese Erweiterung ermöglicht eine Kombination aus Anfragen über Attributen einer Relation und des Text-Retrievals in Dokumenten. Zusätzlich zu den nutzerdefinierten Funktionen enthält der TextExtender ein Application Programming Interface (API), dessen Funktionen in Anwendungsprogrammen zur Textsuche und -darstellung verwendet werden können.

Mit Hilfe der weiteren Mitglieder der Extender-Familie

- Image-Extender
- Audio-Extender
- Video-Extender

können kombinierte Suchanfragen über Image-, Audio- und Video-Datentypen in einer SQL-Anfrage realisiert werden.

Innerhalb dieses Abschnittes werden die verschiedenen Indizierungsformen sowie die Suchfunktionalität des TextExtenders beschrieben.

4.3.1 Indizierung

Obligatorisch für die Retrieval-Funktionalität des TextExtenders ist die Erzeugung eines Indexes auf den zu durchsuchenden Dokumenten, da die Funktionen nicht auf den Dokumenten selbst, sondern auf dem daraus gebildeten Index operieren. Jegliche Änderung an den Dokumenten sollte daher auch immer einen Index-Update nach sich ziehen, da andernfalls ein verfälschtes Retrieval-Ergebnis die Folge sein kann. Der TextExtender speichert die indizierten Terme[8] als invertierte Liste, das heißt, daß zu jedem Term die Dokumente vermerkt werden, die diesen Term enthalten.

Der TextExtender bietet verschiedene Indizierungsarten an, wobei durch jeden Index jeweils nur eine echte Teilmenge der zur Verfügung stehenden Suchoperationen unterstützt wird. Da pro Dokument/Attribut jedoch nur eine Indizierungsform zulässig ist, sind im vorhinein Überlegungen anzustellen, welche Suchoperationen primär genutzt werden sollen.

[8] Die indizierten Terme entsprechen je nach Art des Index entweder der exakte Form oder der Stammform des Begriffes im Dokument.

An dieser Stelle werden zunächst nur die Unterscheidungsmerkmale der Indizes aufgelistet. Die auf den jeweiligen Indizes möglichen Suchoperationen werden in Abschnitt 4.3.3 tabellarisch aufgezeigt.

Linguistischer Index: Die Erzeugung des linguistischen Indexes erfordert eine linguistische Verarbeitung des Textes. Folgende Schritte werden dabei vollzogen, wobei einige Funktionen in Abhängigkeit von der Dokumentensprache entfallen:

- Wort- und Satztrennung
- Erkennung der Satzanfänge
- Aufhebung von Worttrennungen
- Wortnormalisierung — Umformung von Großbuchstaben in entsprechende Kleinbuchstaben und Ersetzung von akzentierten Buchstaben durch nicht akzentierte Formen
  Bsp.: Tür $\longrightarrow$ tuer
- Wortstammreduktion
- Dekomposition zusammengesetzter Wörter (nur deutschsprachig)
- Stoppwortfilterung
- Part-of-speech Filterung
  Diese Wortfilterung ist ähnlich der Stoppwortfilterung. Es werden nur Substantive, Verben und Adjektive indiziert.
  Bsp.: I drive my car quickly. Indiziert werden *car* und *drive*. Die Worte *I* und *my* werden als Stoppworte entkannt, zusätzlich wird aber auch *quickly* durch die Part-of-speech Filterung aus der Indizierung entfernt.
- Feature extraction — Erkennung von Namen, Eigennamen, Abkürzungen
  Feature extraction wird nur für englischsprachige Dokumente unterstützt. Diese Option muß explizit bei der Indexerzeugung spezifiziert werden.

Bei einer Anfrage gegen einen linguistischen Index wird vor der Suche im Index die gleiche linguistische Verarbeitung auf die Begriffe des Suchterms angewendet.

Präziser Index: In dieser Indizierungsform werden die Begriffe entsprechend ihres Auftretens im Dokument indiziert. Der Prozess der Indexerstellung umfaßt dann nur die Schritte

- Wort- und Satztrennung

- Stoppwortfilterung

Dualer Index: Der duale Index stellt eine Kombination aus linguistischem und präzisem Index dar, so daß innerhalb eines Dokumentes sowohl nach der exakten als auch nach der Stammform gesucht werden kann. Die größere Suchfunktionalität schlägt sich jedoch in der Größe des Index und damit in einer längeren Antwortzeit nieder.

Ngram Index: Im Unterschied zu den zuvor beschriebenen Indizes, basiert der Ngram Index nicht auf einem Wörterbuch. Dem Namen des Index nach zu urteilen, beruht er auf dem n-gramm Ähnlichkeitsmaß, das die orthographische Ähnlichkeit zwischen zwei Zeichenketten dadurch berechnet, daß es die Anzahl der übereinstimmenden Buchstabenfolgen (n-Gramme) bestimmt und zur Gesamtzahl aller in beiden Zeichenketten vorkommenden n-Gramme ins Verhältnis setzt. Dies ist die einzige Indizierungsform, die die Fuzzy-Suche unterstützt, so daß Zeichenketten gefunden werden können, die ähnlich dem spezifizierten Suchterm sind.

Enthalten die zu indizierenden Dokumente DBCS-Zeichen[9] (chinesische, japanische, koreanische Schriftzeichen), so stellt der Ngram Index die einzig mögliche Indizierungsform dar, da DBCS Zeichen durch keinen anderen Index unterstützt werden.

Soll mit Hilfe des Ngram Index eine exakte Suche vorgenommen werden, so ist bei der Indexerzeugung der Parameter *case_ enabled* anzugeben.

Für das Erzeugen, Ändern und Löschen von Indizes zur Nutzung der Retrieval-Funktionalität steht ein Kommandozeilenprozessor zur Verfügung, der nicht Bestandteil des DB2 Datenbanksystems ist. Die Speicherung der Indexdateien erfolgt ebenfalls außerhalb der Datenbankumgebung, so daß an dieser Stelle auch nicht die Sicherheitsmechanismen des Datenbanksystems greifen. Dies stellt nur in Hinsicht auf die relativ gute Lesbarkeit der Datendateien des Index ein Sicherheitsrisiko dar, da hierdurch Informationen aus den Dokumenten unbefugt gelesen werden können. Die Verbindung zwischen den externen Indexdateien und dem internen Aufruf der Retrieval-Funktion wird durch die Erzeugung eines Handle-Attributes in der Dokumentenrelation geschaffen. Da sich auch das Dokument selbst außerhalb des Datenbanksystems befinden kann, besteht diese Relation u.U. nur aus dem Handle-Attribut selbst.

4.3.2 Suchfunktionalität

Für die interaktive Suche mit Hilfe des TextExtenders stehen die folgenden vier UDF bereit.

[9] double byte character sets

CONTAINS: Diese Funktion stellt die wichtigste Funktion bei der Volltextsuche mit Hilfe des TextExtenders dar. Da durch das System nur das Boolean-Retrieval-Modell unterstützt wird, liefert CONTAINS, in Abhängigkeit vom Suchergebnis, die Retrieval-Gewichte 0 oder 1 zurück.

$$Suchergebnis \begin{cases} 1 & : \ Suchargument \ \ tritt \ \ im \ \ Dokument \ \ auf \\ 0 & : \ sonst \end{cases}$$

Bei der CONTAINS-Funktion handelt es sich um eine skalare Funktion, die nicht in der FROM-Klausel eines SFW[10]-Blockes angegeben werden darf.

NO_OF_MATCHES: Diese Funktion liefert als Ergebnis die Anzahl der Treffer des Suchargumentes je Dokument. Es handelt sich hierbei ebenfalls um eine skalare Funktion, so daß dieselben Bedingungen gelten, wie für die CONTAINS-Funktion.

RANK: Die Ranking-Funktion soll die Güte eines Dokumentes in Bezug auf das Suchargument widerspiegeln. Beim TextExtender wird diese Rangfolge durch das Verhältnis der Anzahl der Treffer (NO_OF_MATCHES) zur Dokumentengröße errechnet. RANK ist ebenfalls eine skalare Funktion.

SEARCH_RESULT: Im Unterschied zu den zuvor aufgeführten Funktionen liefert diese UDF eine temporäre Relation, bestehend aus den drei Attributen NO_OF_MATCHES, RANK und Handle als Ergebnis des Aufrufes zurück, wobei nur Dokumente selektiert werden, die das spezifizierte Suchargument erfüllen. Der Vorteil dieser Funktion ist die signifikante Verbesserung des Durchsatzes bei der Anwendung auf sehr große Relationen im Vergleich zur RANK oder CONTAINS-Funktion. Der Aufruf der Tabellenfunktion ist nur innerhalb einer FROM-Klausel zulässig.

Jede dieser Funktionen sucht im Index nach dem Vorkommen des spezifizierten Suchargumentes. Zur effizienteren Anfrageausführung wird das Vorhandensein des Suchargumentes im Index nicht bei jedem Aufruf der Funktion geprüft. Statt dessen wird beim ersten Aufruf der UDF eine interne Liste erstellt, die alle Dokumente enthält, die das Suchargument erfüllen. Für jeden weiteren Aufruf auf der Menge der Dokumente einer Relation wird dann aufgrund der internen Liste bestimmt, ob das Dokument in die Ergebnismenge aufgenommen werden muß.

4.3.2.1 Boolesche Suche

Mit Hilfe der booleschen Operatoren & (AND), | (OR) und NOT wird eine logische Verknüpfung der Suchatome (Suchbegriff/Phrase + Attributierung) erreicht. Die Auswertungsreihenfolge ist dabei von links nach rechts, wobei das logische

[10] Select From Where

AND stärker bindet als das logische *OR*. Soll eine andere Auswertungsreihenfolge erzielt werden, so müssen die einzelnen Gruppen durch Klammerung geschachtelt werden. Der logische *NOT* Operator hat in Bezug auf *AND* und *OR* eine etwas eingeschränkte Anwendungsmöglichkeit. So darf direkt nach *NOT*, entsprechend den Regeln der Suchalgebra, nur ein *search primary* folgen. Ein *search primary* wird dabei durch folgende EBNF[11]-Regeln beschrieben:

```
search-primary   ::= Suchatom |
                     '(' search-primary2 ')'

search-primary2 ::= Suchatom { ',' search-primary2 }
```

Das bedeutet, daß nach *NOT* entweder nur ein einzelnes Suchatom (Suchbegriff/Phrase + Attributierung) oder eine durch Komma getrennte Liste von Suchatomen, die durch Klammern begrenzt wird, folgen darf. Die durch Komma getrennten Suchatome werden entsprechend einer Disjunktion ausgewertet.

4.3.2.2 Exakte Suche

Soll die exakte Form eines Suchterms im Index gefunden werden, so kann dies mit Hilfe des *PRECISE FORM OF* Attributes erfolgen. Voraussetzung hierfür ist jedoch ein präziser, dualer oder Ngram Index. Letzterer muß zusätzlich mit der *case_ enabled* Option erstellt worden sein.

4.3.2.3 Stammformsuche

Die Suche nach Variationen eines Terms wird durch die Stammformsuche ermöglicht (Attribut *STEMMED FORM OF*). Voraussetzung hierfür ist jedoch ein linguistischer oder dualer Index. Laut Produkthandbuch ist die Stammformsuche auch auf dem Ngram Index möglich. Da dieser Index aber nicht auf einem Wörterbuch basiert, stellt die Stammformsuche auf diesem Index nur eine Teilstringsuche dar.

Tests dieses Attributes haben leider gezeigt, daß der TextExtender speziell bei deutschsprachigen Dokumenten nicht immer das erwartete Resultat erzielt. So konnten z.B. bei einer Stammformsuche nach `"buch"` Dokumente die den Begriff `"Buch"` enthielten nicht gefunden werden, weil laut Aussage der TextExtender Entwicklungsabteilung das Wort *buch* als Imperativ des Verbs *buchen* interpretiert und daher auf das Verb *buchen* abgebildet wird. Bei den Versuchsanfragen wurden aber noch weitere Worte (Titel, Abschnitt) gefunden, die in ihrer Kleinschreibung durch den TextExtender nicht gefunden wurden.

[11] EBNF - erweiterte Backus-Naur Form

4.3.2.4 Fuzzy-Suche

Die Fuzzy-Suche ermöglicht es, Terme eines Dokumentes zu finden, die Transpositionen bzw. Substitutionen von Zeichen enthalten können. Wie groß die Ungenauigkeiten des Begriffes dabei sein dürfen, hängt von der im Suchausdruck angegebenen Stufe ab. Die Stufe eins läßt mit nur 20 % die größten Freiheiten bei der Treffergenauigkeit zu. In der Stufe fünf wird eine Genauigkeit von 90 % zwischen dem Suchterm und den Begriffen des Index gefordert. Die Fuzzy-Suche wird über das Attribut *FUZZY FORM OF* eingeleitet, wobei auf den Dokumenten ein Ngram Index vorhanden sein muß.

Für die Fuzzy-Suche gilt neben der Einschränkung auf den Ngram Index, daß mindestens die ersten drei Buchstaben des Wortes mit dem Suchterm übereinstimmen müssen, wodurch Suchbegriffe mit Rechtschreibfehlern am Wortanfang nicht lokalisiert werden können. Aus diesem Grund wird das im Abschnitt Fuzzy-Suche des Produkthandbuches dargestellte Beispiel:

```
SELECT subject
FROM   sample
WHERE contains(handle, ' FUZZY FORM OF 2 "compress" ')=1
```

nie das fehlerhafte Wort *conpress* finden.

4.3.2.5 Sound-Suche

Diese Funktion soll die Suche nach Begriffen unterstützen, die den gleichen Klang wie der Suchbegriff haben, aber aufgrund unterschiedlicher Schreibweisen normalerweise nicht gemeinsam gefunden werden.

Die Durchführung diverser Tests zeigte aber, daß diese Funktion für ernsthafte Anwendungen kaum geeignet ist, da sie, ähnlich wie die Fuzzy-Suche, eine Zeichenanalyse unter Beachtung von Transpositionen und Substitutionen vornimmt. Sprachspezifische Betonungen werden dagegen kaum berücksichtigt.

Das nachfolgende Beispiel zeigt, welche Begriffe aufgrund der Sound-Suche des Terms auf der linken Seite gefunden wurden.

cause $\Longrightarrow$ cause, curse, coarse, course, case, ...
bore $\Longrightarrow$ bore, boar, bare, bar, ...
Heer $\Longrightarrow$ Heer, Haar, Heuer, ...

Die Sound-Suche wird normalerweise nur auf dem dualen oder linguistischen Index ausgeführt. Die Anwendung der Suchart auf den präzisen oder Ngram Index sollte eine Warnung zur Folge haben, was innerhalb des Kommandozeilenprozessors auch korrekt erfolgte. Innerhalb der graphischen Benutzerumgebung

des Command Centers konnte aber mehr beobachtet, daß die Durchführung der Sound-Suche auf dem Ngram Index eine nichtbehandelte Exception auslöste, die das gesamte Control-Center terminieren ließ.

4.3.2.6 Proximity-Suche

Werden die Begriffe eines Suchausdrucks mittels der logischen Operatoren miteinander verknüpft, so kann jeder Begriff bei der Suche an einer beliebigen Position des Dokumentes auftreten. Wird jedoch eine gewisse Nähe der Begriffe zueinander gefordert, so daß ein semantischer Zusammenhang erhalten bleibt, kann dies über die Abstandsfunktionen erreicht werden. Die Abstandssuche des TextExtender kann sich entweder auf einen Satz (*IN SAME SENTENCE AS*) oder einen Abschnitt (*IN SAME PARAGRAPH AS*) des Dokumentes beziehen, was aus semantischer Sicht auch plausibel erscheint.

Bei der Anwendung der Satz-Abstandssuche gibt es jedoch Probleme, wenn der Satz eine Punktierung aufgrund von Abkürzungen oder Numerierungen enthält. Bei der Indexerzeugung werden an diesen Stellen Satzgrenzen erkannt, so daß die Satz-Abstandssuche auf verkürzte Sätze angewendet wird.

4.3.2.7 Freitextsuche

Gerade für Gelegenheitsanwender eines Information-Retrieval-Systems ist es oft nicht einfach, die Konstrukte der Retrieval-Sprache so anzuwenden, daß eine effektive Suche erfolgt. Einfacher wäre es, eine natürlich-sprachliche Anfrage zu formulieren, die durch das Retrieval-System semantisch ausgewertet wird. Die semantische Erkennung von Satz-Zusammenhängen ist aber keine triviale Angelegenheit. Beim TextExtender wurde daher versucht, ein simpleres Konzept der Freitextsuche umzusetzen. Zum einen spielt die Reihenfolge der Begriffe hier keine Rolle und zum anderen wird die sogenannte *lexikalische Nähe* unterstützt. Sie repräsentiert Wortpaare, die sowohl im Suchausdruck als auch im Dokument mit einer minimalen Häufigkeit (dies soll die Suche nach Stoppworten verhindern) und einem minimalen Abstand auftreten.

Ausgehend von der Funktionsbeschreibung der Freitextsuche sollten alle Begriffe des Suchterms mit einer minimalen Häufigkeit auch im Dokument vorkommen müssen, um in das Selektionsergebnis aufgenommen zu werden. Die Anwendung auf deutsche Dokumente zeigt aber ein anderes Verhalten, da hier das Retrieval-Ergebnis durch die Disjunktion der Dokumente, die Begriffe mit minimaler Häufigkeit enthielten, erzielt wurde.

4.3.2.8 Suche nach Synonymen

Die Suche nach Synonymformen stellt eine gute Möglichkeit dar, die Suche auf semantisch gleiche Begriffe auszudehnen. So werden bei der Synonymsuche nach dem Begriff *compute* auch die Begriffe *calculate, number, count, reckon, estimate und enumerate* in das Retrieval mit einbezogen. Laut der Produktbeschreibung des TextExtenders ist die Synonymsuche jedoch nicht in Kombination mit dem *NOT* Operator zulässig. Die nachfolgende Anfrage

```
SELECT subject
FROM   sample
WHERE contains(handle, ' NOT SYNONYM FORM OF "compute" ')=1
```

lieferte aber korrekterweise alle Dokumente, die weder den Suchbegriff noch dessen Synonymformen enthielten. Der Versuch, die Synonymsuche auf deutschsprachigen Dokumenten auszuführen, brachte leider keinen Erfolg. Anscheinend fehlte hier das deutschsprachige Synonymwörterbuch im Lieferumfang.

4.3.2.9 Thesaurus Sucherweiterung

Obwohl die Unterstützung der Synonymsuche bereits eine sehr gute Form der semantischen Anreicherung des Retrievals darstellt, werden in vielen Anwendungsdomänen erweiterte Konzepte benötigt. Der TextExtender bietet hier, mittels des Thesaurus Konzeptes die Möglichkeit, die Suche auf spezifische Anwendungsfälle auszudehnen. Hierzu ist es zunächst notwendig, die Terme der Anwendungsdomäne hierarchisch anzuordnen und zwischen den Termen Relationsbeziehungen zu bilden. Auf diese Weise können neben Synonymen auch Ober- und Unterbegriffe sowie assoziierte Terme definiert werden. Aus der SGML-Beschreibung der Terme und Relationen des Thesaurus wird ein internes Modell gebildet, das die Grundlage der Sucherweiterung innerhalb einer Anfrage darstellt. Abbildung 4.2 zeigt einen kleinen Ausschnitt aus der Hierarchie der Domäne *database management system*. Die Terme *relational database management system*, *object-oriented database management system* und *object-relational database management system* repräsentieren jeweils Unterbegriffe (NT) von *database management system*. Die Abkürzung *DBMS* ist über eine Synonymbeziehung (SYN) mit dem Begriff verbunden. *document management system* stellt einen assoziierten Term (RT) zu *database management system* dar.

Ein Retrieval des Begriffes *database management system*, welches auch das Synonym *DBMS* mit in die Suche einbeziehen soll, sieht dann folgendermaßen aus:

```
SELECT subject
FROM   sample
```

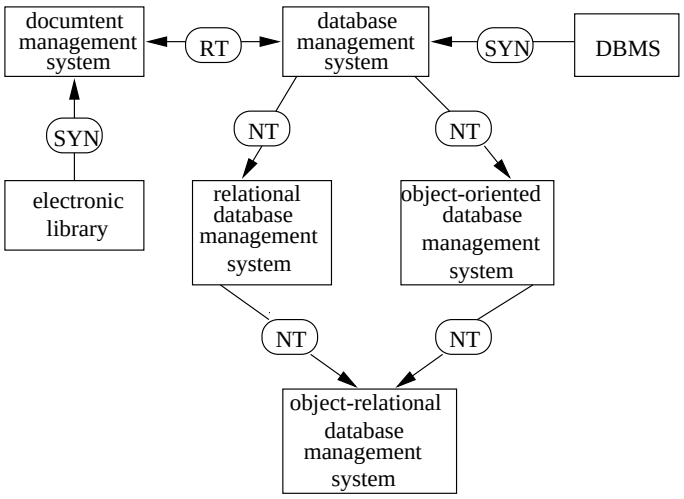

Abbildung 4.2: Beispiel einer Thesaurus-Hierarchie

```
WHERE contains(handle, ' THESAURUS "thesau_db" EXPAND "SYN"
                         TERM OF "database management system" ')=1
```

4.3.3 Zusammenfassung der Funktionalität des TextExtenders

Zum Abschluß der Beschreibung des TextExtenders werden in Tabelle 4.1 die Indizies und die auf ihnen möglichen Funktionen gegenübergestellt. Ein Plus-Zeichen bedeutet dabei, daß die Funktion auf dem Index ausgeführt werden kann. Ein Minus-Zeichen signalisiert, daß die Funktion für diesen Index nicht zur Verfügung steht. Das $\oplus$-Symbol deutet auf eine eingeschränkte Anwendbarkeit der Funktion hin. Hier muß dann entweder bei der Indexerzeugung eine zusätzliche Option angegeben werden oder die Funktionalität wird nur für ausgewählte Sprachen unterstützt.

Als Fazit der Untersuchung kann gesagt werden, daß der TextExtenders von der Volltext-Retrieval-Funktionalität her recht leistungsfähig ist, wenn man beachtet, daß es sich hierbei nur um einen Aufsatz auf ein objekt-relationales Datenbanksystem und nicht um ein reines Volltextdatenbanksystem handelt.

Trotz des positiven Gesamteindrucks müßten einige Funktionen (Sound-Suche, Fuzzy-Suche) in einer neuen Version mit einem verbesserten Algorithmus aufwarten, um sinnvoll eingesetzt werden zu können. Bei anderen Funktionen bedarf es noch einiger Nachbearbeitung (z.B. bei der Stammformsuche), um die fehlerfreie Funktionalität zu gewährleisten. Sollte der Hauptaspekt des TextExtenders weiterhin auf reinem Volltext-Retrieval bestehen, wäre die Erweiterung des Sy-

stems auf das Fuzzy-Retrievalmodell sinnvoll, um durch die Retrieval-Funktion nicht nur die Retrievalgewichte 0 und 1 zu erhalten, sondern Werte im Interval [0,1]. Auf dieser Basis könnten dann auch verschiedene Ranking-Algorithmen bereitgestellt werden. Günstig erscheint weiterhin die Möglichkeit mehrere Indizes auf einem Attribut zuzulassen, um die gesamte Retrieval-Funktionalität nutzen zu können. Aufgrund der Erzeugung eines Handle-Attributes dürfte die Unterscheidung zwischen den erzeugten Indizes prinzipiell auch nicht schwierig sein. Aus der Anwendersicht, war es in vielen Siuationen schwer erkennbar, welche Begriffe des Dokumentes zur Auswahl geführt haben. Eine Selektionsmöglichkeit (Highlight-Funktion) innerhalb der interaktiven Anfragesprache würde hier abhilfe schaffen.

	Linguistisch	**Präzise**	**Dual**	**Ngram**
exakte Suche	-	+	+	⊕
Stammformsuche	+	-	+	-
Fuzzy-Suche	-	-	-	+
Freitextsuche	+	+	+	-
Suche nach Synonymen	+	-	+	-
Sucherweiterung durch einen Thesaurus	+	+	+	-
Sound-Suche	+	-	+	-
Abstandssuche im gleichen Satz	+	+	+	+
Abstandssuche im gleichen Abschnitt	+	+	+	+
Bound	-	-	-	+
Feature-Suche	⊕	-	⊕	-

Tabelle 4.1: Überblick über Index und Suchfunktionalität des TextExtenders

4.4 Informix Universal Server

Das seit der Version 9.1 unter dem Namen Universal Server bekannte Datenbanksystem der Firma Informix Software Inc. ist eines der führenden objektrelationalen DBMS (ORDBMS). Im Vergleich zum DB2 ORDBMS unterstützt Informix weit mehr objekt-orientierte Konzepte. Dazu zählen unter anderem

- Vererbung — Einfachvererbung in Form einer Typ- und Tabellenhierarchie
- Typkonstruktoren für Collection- und Reihen[12]-Datentypen
- Kapselung

[12]Der Reihentyp heißt im englisch Orginal *row type*

- Erzeugung neuer Basisdatentypen
- funktionale Skalierbarkeit durch DataBlade-Technologie

Die im letzten Punkt genannte DataBlade-Technologie ist gekennzeichnet durch die Implementierung benutzerdefinierter Datentypen und Funktionen, wodurch Systemerweiterungen für spezifische Anwendungen möglich werden (siehe Abbildung 4.3). Mit Hilfe der DataBlade-Programmierschnittstelle (API) können DataBlade-Module auch durch Drittanbieter erstellt und anschließend durch Informix getestet und zertifiziert werden. Diese Vorgehensweise bietet dem Anwender zum einen die Flexibilität das System bei Bedarf erweitern zu können und zum anderen die Sicherheit, daß das zertifizierte DataBlade mit dem Datenbanksystem zusammenarbeitet.

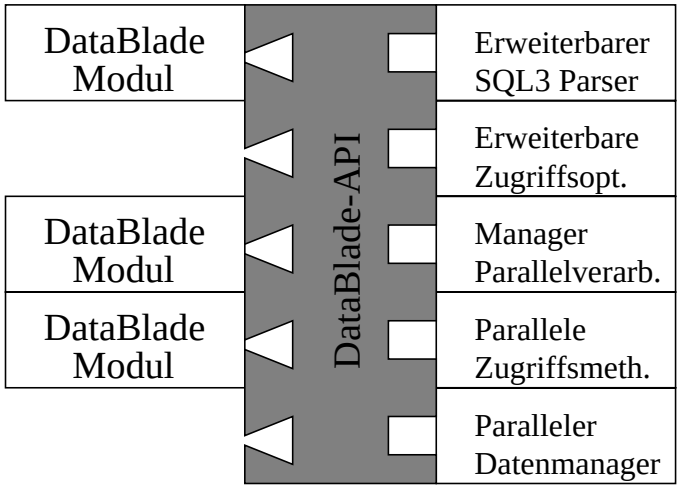

Abbildung 4.3: Schema der Datablade-Technologie

Im folgenden Abschnitt wird die Funktionalität eines dieser zertifizierten DataBlades für das Anwendungsgebiet der Volltextsuche näher untersucht. Die Auswahl dieses DataBlades erfolgte nicht nach qualitativen Gesichtspunkten, sondern aufgrund der Verfügbarkeit am Lehrstuhl für Datenbanken und Informationssysteme.

4.5 Excalibur DataBlade

Das untersuchte Excalibur Text Search DataBlade-Modul in der Version 1.10.UC2 [Inf97] der Firma Excalibur Technologies Cop. stellt Funktionen und Datentypen für das Volltext-Retrieval zur Verfügung. Neben diesem Modul, ist für die

Firma Excalibur ein weiteres DataBlade für das Image-Retrieval zertifiziert. Das Excalibur-Modul besteht aus einem Textsuch-Server in Form einer Bibliothek, die einen speziellen Teil des Informix Universal Servers darstellt. Beim Aufruf einer Funktion des Volltextsystems wird durch das dynamische Binden der Excalibur-Systembibliothek das Retrieval-Modul gestartet. Durch die DataBlade-Philosophie können die Retrieval-Funktionen sowohl in Kombination mit der eingebauten DBMS-Funktionalität als auch mit anderen DataBlades verwendet werden.

Das Excalibur Text-Retrieval-Modul setzt sich aus den vier generellen Komponenten *etx-Zugriffsmethode, etx_contains()* Operator, *Filterfunktionen* und *Systemroutinen* zusammen. In den beiden nachfolgenden Abschnitten wird jedoch nur auf die Zugriffsmethode und den etx_contains Operator näher eingegangen.

4.5.1 Indizierung

Für Anfragen auf unstrukturierten Daten mit Hilfe des Excalibur-DataBlades ist zunächst die Indizierung der Daten unter Verwendung der *etx* Zugriffsmethode erforderlich. Mit ihrer Hilfe wird ein sogenannter *etx-Index* erzeugt, der die spezifische Suche erlaubt. Innerhalb der Zugriffsmethode können verschiedene Parameter festgelegt werden, die die Charakteristik des Index und damit die auf ihm möglichen Suchfunktionalitäten festlegen. Die Erstellung des Index selbst erfolgt durch das DDL[13] Kommando *CREATE INDEX*, das in seiner Syntax um die Angabe einer Operatorklasse (siehe Tabelle 4.2) nach dem Attributnamen und einer *USING* Klausel erweitert wurde. Durch das Schlüsselwort *USING* wird die Angabe einer Zugriffsmethode eingeleitet. Die Zugriffsmethode kann entweder ein B-Baum, ein R-Baum oder eine Zugriffsmethode, die durch ein DataBlade-Modul definiert wurde, sein. Im Falle der Erzeugung eines Index für das Volltext-Retrieval mit dem Excalibur DataBlade ist hier die ext-Zugriffsmethode anzugeben. Die spezielle Operatorklasse eines durch Excalibur unterstützten Datentyps repräsentiert Funktionen für die Indexerzeugung und Anfrageoptimierung.

Datentyp	Operatorklasse
BLOB	etx_blob_ops
CLOB	etx_clob_ops
CHAR	etx_char_ops
VARCHAR	etx_varc_ops
LVARCHAR	etx_lvarc_ops
IfxDocDesc	etx_doc_ops

Tabelle 4.2: Unterstützte Datentypen und assoziierte Operatorklassen

Durch das Beispiel

[13] data definition language

```
CREATE INDEX contentidx
ON          tabelle (content etx_varc_ops)
USING       etx (WORD_SUPPORT = 'EXACT') IN sblobspace
```

wird ein Index auf dem Attribut *content* erzeugt. Die Domäne des Attributes ist vom Datentyp *varchar*, wodurch die Verwendung der Operatorklasse *etx_ varc_ ops* impliziert wird. In der *USING*-Klausel wurde die etx Zugriffsmethode mit dem Parameter *WORD_ SUPPORT* spezifiert.

Es wurde bereits erwähnt, daß innerhalb der Zugriffsmethode durch die Angabe von Parametern die Charakteristik des Index festgelegt wird. Im folgenden werden die Indexparameter sowie ihre möglichen Belegungswerte beschrieben.

WORD_SUPPORT : Durch diesen Parameter wird die mögliche Suchart festgelegt.

EXACT: Es wird nach der exakten Form des angegebenen Suchstrings im Anfrageausdruck gesucht. Dieser Wert ist die Standardbelegung, falls der Parameter im Index nicht spezifiziert wurde.

PATTERN: Durch den Index wird sowohl eine exakte als auch eine Fuzzy-Suche unterstützt.

PHRASE_SUPPORT : Je nach Belegungswert dieses Parameters wird eine Phrasen- und Abstandssuche ermöglicht.

NONE: Dies ist die Standardbelegung, falls der Parameter bei der Indexerzeugung nicht angegeben wurde. Es werden dann nur die Sucharten Stichwortsuche und Boolesche Suche unterstützt.

MEDIUM, MAXIMUM: Sie geben die Genauigkeit an, mit der bei der Phrasen und Abstandssuche gearbeitet werden soll. MEDIUM operiert mit einer schlechteren Genauigkeit, wobei die Wahrscheinlichkeit eines falschen Ergebnisses mit der zunehmenden Länge einer Phrase sinkt. Der Nachteil des schlechteren Suchergebnisses von MEDIUM wird jedoch durch die bessere Such-Performance aufgrund des kleineren Index amortisiert.

CHAR_SET : Die Angabe eines Zeichensatzes legt fest, welche Zeichen indexiert werden. Neben den Zeichensätzen ASCII, ISO und Overlap ISO können seit der Version 1.10.UC2 auch eigene Zeichensätze angegeben werden. Zu beachten ist, daß Zeichen des Dokumentes, die im angegebenen Zeichensatz nicht enthalten sind, als Leerzeichen interpretiert werden, so daß unter Umständen eine Aufsplittung von Wörtern erfolgt.

Beispiel: Im ASCII-Zeichensatz ist das deutsche ß nicht enthalten, so daß *Straße* als *Stra* und *e* indexiert wird.

STOPWORD_LIST : Durch diesen Parameter wird die Indizierung, der in der spezifizierten Stoppwortliste enthaltenen Begriffe verhindert. Die angegebene Stoppwortliste muß jedoch zuvor mit Hilfe der Routine *etx_CreateStopWlst* innerhalb eines logischen Speicherbereiches erzeugt worden sein. Die Verwendung einer Stoppwortliste ist in jedem Fall anzuraten, da dadurch die resultierende Größe des Index um einen wesentlichen Faktor reduziert wird.

INCLUDE_STOPWORDS : Dieser Parameter kann nur in Verbindung mit dem Parameter *STOPWORD_LIST* verwendet werden. Die möglichen Belegungen sind die booleschen Werte TRUE und FALSE. Durch den Wert TRUE werden die Begriffe der angegebenen Stoppwortliste zwar indiziert, jedoch werden sie nur berücksichtigt, falls im Suchausdruck der Parameter INCLUDE_STOPWORDS angegeben wurde.

Für ein Attribut einer Relation können ohne weiteres mehrere Indizes mit Hilfe der etx Zugriffsmethode erzeugt werden. Überraschender Weise wird jeweils der älteste Index für die Suche verwendet. Wichtig bei der Erzeugung eines Index, ist auch zu wissen, daß prinzipiell keine sprachspezifische linguistische Verarbeitung der Begriffe existiert. Sprachabhängige Symbole werden nur durch die Wahl des Zeichensatzen bei der Indexerstellung berücksichtigt.

4.5.2 Suchfunktionalität

Die gesamte Suchfunktionalität des Excalibur-DataBlades wird über den *etx_contains* Operator gesteuert. Dieser Operator besitzt drei Parameter, wobei der erste und zweite Parameter obligatorisch und der dritte Parameter optional ist.

1. Parameter: Er gibt den Namen des Attributes an, auf dessen etx-Index die Suche erfolgen soll.

2. Parameter: Dieser Parameter legt den Suchterm fest. Er kann entweder als einfache, durch Anführungszeichen umgebene Zeichenkette oder als Reihentyp mit zwei Attributen vom Datentyp *LVARCHAR* angegeben werden. Im Falle des Reihentyps wird durch das erste Attribut das Suchkriterium festgelegt und durch das zweite Attribut können Tuningparameter angegeben, die die Suchform beeinflussen. In den folgenden Unterabschnitten werden die einzelnen Argumente näher beleuchtet.

3. Parameter: Durch den dritten Parameter kann optional eine sogenannte *statement local variable* (SLV) angegeben werden. Sie ermöglicht es, den

in der Funktionsdeklaration angegebenen Ausgabeparameter[14] in einen anderen Teil des SQL-Statements zu überführen. Der Ausgabeparameter des etx_contains Operator ist vom benannten Reihentyp *etx_ReturnType*. Die Elemente des Reihentyps beinhalten Ranking- und Highlight-Informationen.

4.5.2.1 Stichwortsuche

Bei der Anwendung der Stichwortsuche auf einen Index werden sämtliche Begriffe des Suchkriteriums als Disjunktion behandelt, so daß alle Tupel der Relation selektiert werden, die wenigstens einen der Begriffe des Suchterms im Index des Attributes enthalten. Die Stichwortsuche ist eine der beiden Suchformen, die in jedem Fall möglich ist, selbst wenn bei der Indexerzeugung in der Zugriffsmethode keine Argumente angegeben wurden. Wird der zweite Parameter durch eine quotierte Zeichenkette ausgedrückt, so repräsentiert die Stichwortsuche die Standard-Suchart. Im Fall des Reihentyps kann die Angabe wiederum entfallen oder sie kann im zweiten Argument des Reihentyps über das Attribut-Wert-Paar *SEARCH_TYPE=WORD* explizit bestimmt werden. Das nachfolgende Beispiel zeigt den Fall eines Reihentyps mit expliziter Angabe der Stichwortsuche.

```
SELECT subject
FROM   sample
WHERE etx_contains(content,
        row('relationale Datenbank', 'SEARCH_TYPE=WORD'))
```

Werden innerhalb der Tuningparameter keine weiteren Attribute spezifiziert, so soll, gemäß des Produkthandbuches, bei der Suche ein *exact match* angewendet werden. Erstaunlicherweise wird hier nur die Reihenfolge der Buchstaben beachtet, nicht jedoch ihre Schreibweise, so daß das Wort *Datenbank* mit dem Wort *dAtENbanK* identisch ist. Da es auch keine weitere Option gibt, eine präzise Suche festzulegen, ist es zum Beispiel nicht möglich das deutsche Substantiv *Wissen* von dem Infinitiv *wissen* zu differenzieren.

4.5.2.2 Boolesche Suche

Durch die logischen Operatoren & (AND), | (OR) und ! (NOT) wird eine logische Verknüpfung der Begriffe des Suchkriteriums vorgenommen. Die Auswertungsreihenfolge ist dabei von links nach rechts, wobei das logische *AND* stärker bindet als das logische *OR*.

Die Boolesche Suche ist neben der Stichwortsuche die zweite Suchart, die auch ohne Parametrisierung der Zugriffsmethode ausgeführt werden kann. Sie wird durch

[14] dieser ist nicht mit dem return type einer Funktion zu verwechseln

die Zuweisung des Wertes *BOOLEAN_SEARCH* an den *SEARCH_TYPE*-Parameter eingeleitet.

Bespiel:

```
SELECT subject
FROM   sample
WHERE etx_contains(content, row('relational & Datenbank',
          'SEARCH_TYPE=BOOLEAN_SEARCH'))
```

Nachteilig bei der Anwendung der booleschen Suche ist die fehlende Möglichkeit Ausdrücke zu klammern, um eine andere Auswertungsreihenfolge zu erzielen. Obwohl es vom theoretischen Standpunkt korrekt ist, daß durch die Operatoren *AND*, *OR* und *NOT* sowie der Vorrangregel für *AND* jegliche Boolesche Funktionen durch die Darstellung als disjunktive Normalform ohne Klammerung beschrieben werden können, wird durch diese Darstellung weder Übersichtlichkeit noch einfache Anwendbarkeit erreicht.

4.5.2.3 Phrasensuche

Im Kontext des Excalibur DataBlade stellt eine Phrase ein Suchkriterium dar, das mehr als einen Begriff enthält. Im Kontrast zur Stichwortsuche, bei der die Begriffe des Suchkriteriums getrennt voneinander gesucht werden, bilden die Begriffe bei der Phrasensuche eine Einheit.

Durch das DataBlade-Modul werden zwei Formen der Phrasensuche unterstützt. Zum einen ist dies die exakte Phrasensuche, bei der die einzelnen Begriffe des Suchausdruckes in der selben Reihenfolge im Dokument auftreten müssen. Sie wird durch die Zuweisung des Wertes *PHRASE_EXACT* an den *SEARCH_TYPE*-Parameter eingeleitet. Die zweite Form ist die approximierte Phrasensuche, in der weder die Reihenfolge der Begriffe, noch ihre exakte Form eine maßgebliche Rolle spielt. Die Durchführung einer approximierten Phrasensuche wird durch das Attribut-Wert-Paar *SEARCH_TYPE=PHRASE_APPROX* ermöglicht. Im Unterschied zur Stichwort- und Booleschen Suche, die auf allen etx-Indizes möglich sind, kann die Phrasensuche nur auf einem etx-Index operieren, der mit dem Parameter *PHRASE_SUPPORT* erzeugt wurde.

Im Excalibur Produkthandbuch wird von der Verwendung der approximierten Phrasensuche aufgrund der schlechten Performance abgeraten. Dies konnte bei den durchgeführten Tests durch die zu geringe Dokumentenanzahl und -größe nicht beobachtet werden, jedoch war bei den Testanfragen die Retrieval-Qualität teilweise recht schlecht. Für die approximierte Phrasensuche reichte es in jedem Fall aus, wenn ein Begriff der Phrase indiziert war. In dem Fall des folgenden Beispiels wurden alle Dokumente selektiert, die den Begriff *Datenbank* enthielten.

```
SELECT subject
FROM   sample
WHERE etx_contains(content, row('xxx Datenbank yyy ',
        'SEARCH_TYPE=PHRASE_APPROX'))
```

4.5.2.4 Proximity-Suche

Wie bereits erwähnt, betrachten die Stichwort- und die Boolesche Suchform die einzelnen Terme getrennt voneinander, so daß auch der Kontext, in dem sie sich befinden, unterschiedlich sein kann. Eine Kontextabhängigkeit wird in gewisser Weise durch die exakte Phrasensuche erzielt, die aber durch die Forderung, daß sämtliche Begriffe in der angegebenen Reihenfolge auftreten müssen, im allgemeinen zu restriktiv ist. In vielen Situationen ist es so, daß zwar der semantische Zusammenhang gewünscht wird, eine exakte Formulierung der Phrase jedoch nicht angegeben werden kann. Ein Ausweg in Excalibur ist die Proximity-Suchform. Hier kann neben der Phrase ein Abstandsparameter spezifiziert werden, der die maximale Anzahl an Worten angibt, die zwischen den Begiffen des Suchkriteriums auftreten dürfen, wobei die beiden Begriffe selbst auch mitgezählt werden. In die Abstandsberechnung werden sämtliche Begriffe des Dokumentes mit einbezogen. Neben der geforderten Nähe der Begriffe zueinander, müssen sämtliche Terme im Dokument auftreten, nicht jedoch in der angegebenen Reihenfolge.

Beispiel:

```
SELECT subject
FROM   sample
WHERE etx_contains(content, row('relationale Datenbank',
        'SEARCH_TYPE=PROX_SEARCH(5)'))
```

4.5.2.5 Fuzzy-Suche

In den bisherigen Untersuchungen wurde davon ausgegangen, daß die Terme des Suchkriteriums in ihrer exakten Form (bzw. in der durch Excalibur unterstützten exakten Form, siehe Abschnitt 4.5.2.1) im Dokument auftraten. Die volle Mächtigkeit eines Volltext-Retrieval-Systems zeigt sich jedoch erst, wenn gewisse unscharfe Suchanfragen möglich werden, die neben der exakten Suche auch Suchanfragen nach Begriffen mit orthographischen Fehlern bzw. die Suche nach Wortteilen zulassen. Um solch eine Suche mit Hilfe des etx_contains Operators ausführen zu können, muß bei der Erzeugung des etx-Index der Parameter *WORD_SUPPORT* der etx-Zugriffsmethode mit dem Wert *PATTERN* belegt worden sein. Die größere Flexibilität in den Retrieval-Anfragen wird dann durch die Verwendung der vier verschiedenen Fuzzy-Parameter erreicht.

PATTERN_TRANS : Die Ergebnismenge der Anfrage enthält Dokumente, bei denen das Suchkriterium entweder exakt im Dokument auftritt oder deren Terme genau eine Transposition (Vertauschung zweier benachbarter Zeichen) gegenüber den Termen des Suchkriteriums aufweisen.

PATTERN_SUBS : Die Ergebnismenge der Anfrage enthält Dokumente, bei denen das Suchkriterium entweder exakt im Dokument auftritt oder deren Terme genau eine Substitution (Ersetzung eines Zeichens durch ein anderes) gegenüber den Termen des Suchkriteriums aufweisen.

PATTERN_BASIC : Durch die Parametrisierung einer Anfrage mit *PATTERN_BASIC* werden neben der spezifizierten Form des Suchausdruckes auch Ober- und Untermengen des Suchmuster im Index gesucht.

PATTERN_ALL : Dieser Parameter repräsentiert die höchste Abstraktionsebene der Fuzzy-Suche. Hier wird das Suchkriterium auf Begriffe mit multiplen Transpositionen und Substitutionen, sowie auf Ober- und Untermengen des Suchmusters erweitert.

Im Handbuch zum Informix-DataBlade wurde keinerlei Aussage darüber getroffen, wie weit die Begriffe Ober- bzw. Untermenge zu ziehen sind. Durch Testanfragen stellte sich heraus, daß ein Oberbegriff maximal zwei Zeichen am Wortanfang und drei Zeichen am Wortende über den Suchbegriff hinausgehen darf, so daß die Retrieval-Anfrage des Handbuches auf Seite 1-29:

```
SELECT subject
FROM   sample
WHERE etx_contains(content,
          row('retrieve', 'PATTERN_BASIS'))
```

zwar korrekterweise die Begriffe *retrieve* und *retrieving*, nicht jedoch den angegebenen Begriff *irretrieveable* finden wird. Desweiteren sollen durch diesen Parameter Transpositionen und Substitutionen unbeachtet gelassen werden. Dies ist jedoch nicht der Fall, da beispielsweise auch der fehlerhafte Begriff *retreive* gefunden wird.

4.5.2.6 Suche nach Synonymen

Ist der Inhalt der Dokumente, auf die sich eine Suchanfrage bezieht im vorhinein unbekannt, so ist es hilfreich, die Anfrage durch eine Synonymwortliste anzureichern, so daß neben den eigentlichen Suchtermen auch sinnverwandte Begriffe in die Suche einbezogen werden. Im Excalibur DataBlade kann die Synonymsuche über den Parameter *MATCH_SYNONYM* gesteuert werden. Wird diesem

Parameter kein Wert zugewiesen, so wird die Standardsynonymliste verwendet. Sie enthält jedoch nur Synonyme der englischen Sprache. Sollen Synonyme einer anderen Sprache zur Anwendung kommen oder ein erweiterter englischer Wortschatz, so kann über die Systemroutine *etx_CreateSynWlst* eine eigene Synonymwortliste hinzugefügt werden. Eine Excalibur-Synonymliste besteht aus mehreren Wortreihen. Das erste Wort jeder Reihe stellt den Descriptor dar; alle weiteren Worte sind sinnverwandte Begriffe des Descriptors.

Beispiel:
PENSION FERIENWOHNUNG APPARTEMENT

Damit eine Suchanfrage um die Synonyme eines Begriffe angereichert wird, muß dieser in der Synonymliste als Descriptor erscheinen und die Suchanfrage darf keinen Fuzzy-Parameter enthalten.

Beispiel:

```
SELECT subject
FROM   sample
WHERE etx_contains(content,
          row('Pension', 'MATCH_SYNONYM=syn_deutsch'))
```

Leider war im gesamten Handbuch kein Hinweis darauf zu finden, daß sämtliche Begriffe der Synonymliste in Großbuchstaben zu schreiben sind, so daß es einiger Versuche bedurfte, ehe die erste Suche auch eigene Synonyme mit einbezog.

4.5.2.7 Ranking der Ergebnisse

Der Indikator für die Güte eines Dokumentes zur gestellten Anfrage ist ein Wert zwischen 0 und 100. Beim Wert Null fand die Anfrage in den entsprechenden Dokumenten keinen Treffer. Der Wert 100 spiegelt das exakte Vorkommen des Suchkriteriums im Dokument wieder. Alle Wert zwischen diesen Extremwerten deuten darauf hin, daß das Suchkriterium nur in einer approximierten Form im Dokument auftritt. Für die Berechnung der Ranking-Werte werden die folgenden zwei Regeln angewendet:

1. Exakte Übereinstimmungen werden immer geringfügig besser bewertet als Pattern-Matches.

2. Alle Pattern-Matches werden gleich bewertet.

Zu Beginn des Abschnittes 4.5.2 wurden die Argumente des etx_contains Operator erläutert. Als drittes Argument kann hier eine optionale *statement local*

variable angegeben werden, durch die die Rankingwerte in anderen Teilen des SQL-Statements verwendet werden können.

Obwohl es generell schwierig, ist beim Boolean-Retrieval-Modell eine aussagekräftige Rangliste zu bilden, wäre es vielleich angebracht, auch die Vorkommenshäufigkeiten mit in die Rangliste einzubeziehen. So sind die Ranglistenwerte einer booleschen Anfrage, deren Terme durch Konjunktionen verbunden sind und die keine Fuzzy-Parameter enthält, generell 100.

4.5.2.8 Result-Highlighting

Sämtliche Begriffe eines Dokumentes, die durch das Suchkriterium einer Retrieval-Anfrage repräsentiert werden, können über die Funktion *etx_GetHilite* ausgegeben werden. Der Rückgabetyp dieser Funktion ist ein Reihentyp mit den Feldern *vec_offset* und *viewer_doc*. Das Feld *vec_offset* enthält für jeden übereinstimmenden Begriff die absolute Anfangsposition im Dokument und die Länge des Begriffes. Das Feld *viewer_doc* beinhaltet das von Formatierungsparametern befreite Dokument.

4.5.3 Zusammenfassung der Funktionalität des DataBlade

In der Tabelle 4.3 wird noch einmal der Zusammenhang zwischen den unterstützten Sucharten und den beiden Indexparametern *WORD_SUPPORT* und *PHRASE_SUPPORT* verdeutlicht. Ein Plus beim *WORD_SUPPORT* verweist darauf, daß die Suchart mit dieser Belegung des Indexparamters harmoniert. Die Angaben zu *Phrase_Support* bestimmen die Notwendigkeit dieses Paramters bei der Indexerzeugung, falls die entsprechende Suchart zum Einsatz kommt.

	WORD_SUPPORT		
Suchart	EXACT	PATTERN	Phrase-Support
Stichwort-Suche	+	+	nicht erforderlich
Boolesche Suche	+	+	nicht erforderlich
Fuzzy-Suche	-	+	nicht erforderlich
Phrasensuche	+	+	erforderlich
Proximity-Suche	+	+	erforderlich

Tabelle 4.3: Überblick über ausgewählten Indexparametern und der Suchfunktionalität

Als Fazit der Untersuchung des Excalibur-DataBlade sind viele Restrktionen hinsichtlich der Retrieval-Funktionalität zu konstatieren. Abgesehen vom Fehlen einer Wortstammreduktion, können auch exakte Anfragen nur in der in 4.5.2.1

genannten Weise durchgeführt werden. Desweiteren beziehen sich sämtliche Tuningparameter einer Retrieval-Anfrage stets auf das gesamte Suchkriterium, so daß eine spezifische Formulierung für einzelne Terme nur durch eine getrennte Anfrage realisierbar ist. Diese darf aber wiederum nicht mit weiteren *etx_contains* Operatoren in einer WHERE-Klausel kombiniert werden. Der negative Eindruck aufgrund des begrenzten Funktionsumfanges wird noch verstärkt, durch Funktionen die ein, von der Spezifikation abweichendes Retrieval-Ergebnis liefern (z.B. PATTERN_BASIC).

4.6 Vergleich der Retrievalfunktionalität

Abschließend sollen die beiden Volltexterweiterungen einem gemeinsamen Vergleich unterzogen werden. Als Vergleichsmaßstab dienen im wesentlichen die in [Rah98] verwendeten Kriterien. Neben der Unterstützung der Kriterien durch das System soll, insofern möglich, die Anwendbarkeit beurteilt werden. Maßstab für eine Anwendbarkeit ist schwerpunktmaßig die durch die Untersuchung festgestellte Retrieval-Qualität. Einfluß haben aber auch Faktoren wie beispielsweise die Handhabung.

Die Einordnung der Systeme hinsichtlich der Unterstützung der Kriterien erfolgt nach folgenden Gesichtspunkten.

- $\oplus$ Das Kriterium wird durch das System unterstützt.
- $\otimes$ Das Kriterium kann nur simuliert werden.
- $\ominus$ Das Kriterium wird nicht durch das System unterstützt.

Die Beurteilung der Anwendbarkeit wird aufgrund folgender Klassifikation vorgenommen

- $+$ Sehr gute Anwendbarkeit — Die Retrieval-Qualität entsprach den Erwartungen bzw. es wurden keine Einschränkungen festgestellt.
- $\pm$ Gute Anwendbarkeit — Die Retrieval-Qualität war überwiegend gut bzw. es wurden nur geringfügige Einschränkungen festgestellt.
- $\mp$ Eingeschränkte Anwendbarkeit — Die Retrieval-Qualität war nur ausreichend bzw. es wurden größere Einschränkungen festgestellt.
- $-$ Unzureichende Anwendbarkeit — Die Anwendung der Funktion wird aufgrund unzureichendender Retrieval-Ergebnisse oder anderer Einschränkungen nicht empfohlen.

Retrievalmodell. Sowohl durch den TextExtender als auch durch Excalibur wird lediglich das Boolean Retrieval-Modell unterstützt, so daß für beide Systeme auch die in 4.1 aufgeführten Nachteile gelten.

Indizierung. Beide Systeme arbeiten beim Volltext-Retrieval nicht auf den Datenbeständen selbst, sondern auf einem speziellen Index. Klarer Vorteil vom Excalibur-System ist, daß der Index nicht vom Datenbanksystem getrennt ist und der Index nach Änderungsoperationen automatisch angepaßt wird. Der TextExtender wiederum wartet für die Indizierung mit umfangreicher Sprachunterstützung auf, die in Excalibur nur durch einen entsprechenden Zeichensatz simuliert werden kann.
Excalibur: *Unterstützung* $\oplus$
TextExtender: *Unterstützung* $\oplus$

Wortstammreduktion. Die Wortstammreduktion ist eine der Kernpunkte für eine linguistische Suche. Erst dadurch wird es möglich die Suche auf alle Begriffsformen (Zeitformen, Deklinationen) zu erweitern. Der TextExtender führt eine Wortstammreduktion bei der Erstellung des linguistischen und des dualen Index durch, sowie auf den Termen von Anfragen, die auf diesen Indizes operieren. Durch die in Abschnitt 4.3.2.3 festgestellten fehlerhaften Retrieval-Ergebnisse wird die Anwendbarkeit der Stammformsuche des TextExtenders allerdings abgewertet. Durch Excalibur wird keine Stammformreduktion unterstützt. Das Fehlen der Stammformreduktion stellt eine große Schwäche des Excalibur dar, da die linguistischen Suche einen der Hauptaspekte des Text-Retrievals repräsentiert, um Unschärfen in Anfragen zuzulassen. In Excalibur kann nur eine sehr eingeschränkte Simulierung der Wortstammsuche über den Fuzzy-Parameter *PATTERN_BASIC* erfolgen.
Excalibur: *Unterstützung* $\ominus$
TextExtender: *Unterstützung* $\oplus$, *Anwendbarkeit* $\pm$

natürlich-sprachliche Anfragen. Eine natürlich-sprachliche Anfrage kann zwar in beiden Systemen formuliert aber nicht auf natürlich-sprachliche Weise verarbeitet werden. Im TextExtender kann eine solche Anfrage unter Verwendung der Freitextsuche (Abschnitt 4.3.2.7) auf Begriffshäufigkeiten und -Abstände untersucht und verarbeitet werden. Die semantischen Zusammenhänge der Begriffe werden aber nicht erkannt, so daß das Retrieval-Ergebnis unter Umständen völlig falsch ist. Im Excalibur-System kann eine natürlich-sprachliche Anfrage nur als Phrasen- oder Stichwortsuche ausgewertet werden, was ebenfalls zu falschen Retrieval-Ergebnisse führen kann.
Excalibur: *Unterstützung* $\otimes$, *Anwendbarkeit* $\mp$
TextExtender: *Unterstützung* $\otimes$, *Anwendbarkeit* $\mp$

Thesaurus. Der Vergleich des Kriteriums Thesaurus fällt klar zu gunsten des TextExtenders aus. Neben der Erweiterung einer Anfrage auf Synonymformen der enthaltenen Terme, kann zusätzlich für spezielle Anwendungsgebiete ein Thesaurus erzeugt werden (Abschnitt 4.3.2.9), der die Suche dann um zusätzliche Relationsbeziehungen erweitert. In Excalibur kann hingegen eine Anfrage nur um Synonymformen erweitert werden.
Excalibur: *Unterstützung* $\oplus$, *Anwendbarkeit* $\pm$

TextExtender: Unterstützung $\oplus$, *Anwendbarkeit* $+$

Boolesche Suche. Die Boolesche Suche wird durch beide Systeme unterstützt und die Anfrageergebnisse entsprachen den zu erwartenden Resultaten. Das Excalibur-System wird in der Bewertung der Anwendbarkeit jedoch etwas herabgesetzt, da aufgrund der fehlenden Schachtelung die Frageformulierung umfangreicher Anfragen schwierig ist.
Excalibur: Unterstützung $\oplus$, *Anwendbarkeit* $\pm$
TextExtender: Unterstützung $\oplus$, *Anwendbarkeit* $+$

Proximity-Suche. Funktionen zur Abstandssuche werden durch beide Systeme bereitgestellt. Der semanitsche Zusammenhang tritt durch die Funktionen des TextExtender (Abschnitt 4.3.2.6) deutlicher hervor, jedoch führt die Erkennung fehlerhafter Satzgrenzen zur Abwertung, da hierdurch das Retrieval-Ergebnis verfälscht wird. Im Fall des Excalibur kann durch Angabe eines zu großen Abstandswertes der gemeinsame Kontext der Terme nicht gewährt sein, so daß die Abstandssuche im schlechtesten Fall zur Stichwortsuche ausartet.
Excalibur: Unterstützung $\oplus$, *Anwendbarkeit* $\pm$
TextExtender: Unterstützung $\oplus$, *Anwendbarkeit* $\pm$

Sound-Suche. Diese Suchform steht nur im TextExtender zur Verfügung. Aufgrund der in Abschnitt 4.3.2.5 aufgeführten Schwächen im Retrieval wird die Anwendbarkeit dieser Funktion als gering bewertet.
Excalibur: Unterstützung $\ominus$
TextExtender: Unterstützung $\oplus$, *Anwendbarkeit* $\mp$

Phrasensuche. Eine exakte Phrasensuche ist in beiden Systemen möglich. Die Bewertung der Excalibur-Funktion wird aber geringfügig abgewertet, da trotz exakter Suche keine Differenzierung zwischen Groß- und Kleinbuchstaben erfolgt. Die Suche nach einer Phrase deren Terme eine andere Reihenfolgen besitzen dürfen, kann im TextExtender relativ gut über die Proximity-Suche simuliert werden. Im Excalibur existiert hiefür die Funktion der approximierten Phrasensuche, die jedoch die Phrasensuche zu größzügig aufgefaßt, wodurch ein relativ schlechtes Retrieval-Ergebnis die Folge ist.
Excalibur: Unterstützung $\oplus$, *Anwendbarkeit* $\pm$ / $-$
TextExtender: Unterstützung $\oplus$ / $\otimes$, *Anwendbarkeit* $+$ / $\pm$

Zeichenmaskierung. Der TextExtender bietet zwei Zeichen zur Maskierung an. Durch % wird eine beliebige Anzahl Zeichen und durch _ wird genau ein Zeichen an der entsprechenden Stelle maskiert. Die Zeichenmaskierung eignet sich jedoch nur für den präzisen und dualen Index, da im Fall des linguistischen Index ein maskiertes Zeichen nicht auf die Stammform reduziert werden kann. Durch Excalibur kann die Zeichemaskierung nur mit Hilfe des Fuzzy-Parameters zur Substitution simuliert werden. Auf diese Weise kann aber maximal ein Zeichen eines Terms maskiert werden.
Excalibur: Unterstützung $\otimes$, *Anwendbarkeit* $\mp$

TextExtender: Unterstützung $\oplus$, *Anwendbarkeit* $\pm$

Ranking. Beide Systeme stellen genau eine Ranking-Funktion (Abschnitt 4.3.2 und 4.5.2.7) zur Verfügung, jedoch ist keine der beiden Funktionen ideal. Die Ursache ist hier im verwendeten Retrieval-Modell zu suchen, das nur eine nachträgliche Bewertung der Ergebnisse erlaubt. Die exklusive Bewertung der Vorkommenshäufigkeiten des TextExtender hat z.B. zur Folge, daß eine Stammform eines Terms, die mehrfach in einem Dokument auftritt, in der Rangfolge besser bewertet wird als ein exakter Treffer, der genau einmal im Dokument enthalten ist (Voraussetzung ist, daß bei Dokumente ähnliche Größe aufweisen).
Excalibur: Unterstützung $\oplus$, *Anwendbarkeit* $\mp$
TextExtender: Unterstützung $\oplus$, *Anwendbarkeit* $\mp$

Relevance Feedback. Aufgrund des verwendeten Retrieval-Modell wird Relevance Feedback weder durch den TextExtender noch durch Excalibur unterstützt. Im TextExtender steht lediglich eine Funktion zur Verfügung, die eine optimierte Form der Anfrageverfeinerung vornimmt. Die Verfeinerung der Anfrage erfolgt auf dem Retrieval-Ergebnis der zuvor gestellten Anfrage. Eine Anfrageverfeinerung ist auch in Excalibur über die Datenbankfunktionalität zu realisieren. Betont werden muß, daß die Verfeinerung einer Anfrage nicht dem in 4.1 beschriebenen Relevance Feedback entspricht.
Excalibur: Unterstützung $\otimes$
TextExtender: Unterstützung $\otimes$

Result-Hightlighting. Die größte Einschränkung des TextExtender gegenüber Excalibur stellt das Fehlen einer Funktion dar, die Aufschluß über die Terme gibt, die zur Auswahl des entsprechenden Dokumentes geführt haben. Im Excalibur-System kann dies auf die in 4.5.2.8 beschriebene Weise erfolgen.
Excalibur: Unterstützung $\oplus$, *Anwendbarkeit* $+$
TextExtender: Unterstützung $\ominus$

Kombinierte Text/Attributanfragen. Die kombinierte Anfrage über Attribute der Datenbank und Text-Retrieval ist weniger ein Merkmal der Volltext-Erweiterungen als des Datenbanksystems. Durch beide Datenbanksysteme sind kombinierte Anfragen möglich, jedoch erlaubt Informix nicht die mehrfache Aufführung der selben nutzerdefinierten Funktion in der WHERE-Klausel.
Excalibur: Unterstützung $\oplus$, *Anwendbarkeit* $\pm$
TextExtender: Unterstützung $\oplus$, *Anwendbarkeit* $+$

Speicherung des Dokumentes. Beide Systeme ermöglichen die Speicherung der Dokumente sowohl im Datenbanksystem als auch im Dateisystem.
Excalibur: Unterstützung $\oplus$
TextExtender: Unterstützung $\oplus$

Unterstützung verschiedener Textformate. Die Filtermöglichkeit für diverse Dateiformate ist in beiden Systeme äußerst eingeschränkt.
Excalibur: Unterstützung $\oplus$, *Anwendbarkeit* $\mp$

TextExtender: Unterstützung $\oplus$, *Anwendbarkeit* $\mp$

Qunitessenz der Evaluierung der Volltext-Retrieval-Funktionalität ist, daß der TextExtender von der Funktionalität her wesentlich leistungsfähiger ist als das Excalibur DataBlade. Dies betrifft insbesondere die sprachspezifische linguistische Suche des TextExtenders, die Möglichkeit der gezielten Attributierung einzelner Terme und die Sucherweiterungen mit Hilfe eines Thesaurus, der über die Möglichkeiten einer Synonymwortliste hinausgeht. Sollte beispielsweise für die Konzeption einer Suchmaschine die Wahl getroffen werden, auf welches Volltext-Retrieval-System ihre Anfragen abgebildet werden sollen, so ist aufgrund der Analyse eindeutig der TextExtender zu bevorzugen, da das Excalibur-Datablade zuviele Restriktionen hinsichtlich der Funktionalität aufweist. Betont werden muß aber, daß sich die Bewertung allein auf die Volltexterweiterungen beziehen und nicht auf das jeweilige Datenbanksystem. Sollte eine Entscheidung hinsichtlich des Datenbanksystems gefällt werden, würde die Entscheidung auf Informix fallen.

Kapitel 5

Prototypische Datentyperweiterung

5.1 Konzeption für Informix Universal Server

5.2 Konzeption und Implementierung für DB2 UDB

Kapitel 6

Konzeption eines XML-Datentyps

Der Vorschlag diese XML-Datentypes beruht auf den Erfahrungen, die aufgrund der Implementierung der Erweiterung auf dem Volltext-Retrievaldatentyp gemacht wurden.

Die Methoden, die fuer diese Realisierung vorgeschlagen werden, sollen unabhaengig von einer Anwendungsdomaende sein. Daher wird hier kein Vorschlag fuer ein moegliches Datenbankschema gegeben.

Aufgaben: Globales Ranking, diverse Funktionen die die Strukturelemente als auch die Inhalte des Dokumentes betreffen Funktionen wie Union, Intersection, Difference etc auf XML-Dokumenten Einfuegen von XML-Dokumenten in die Datenbank (validierend, nichtvalidieren) Update von XML-Dokumenten (validierend, nichtvalidierend) Indexierungsvorschlag Moeglichkeiten zur partiellen Indexierung Strukturerkennung Restrukturierung von Dokumententeilen (Achtung, es muss ein gueltiges Dokument entstehen)

Kapitel 7

Schlußbetrachtung

7.1 Zusammenfassung

7.2 Ausblick

Literaturverzeichnis

[Abi97] Abiteboul, S.: Querying semi-structured data. In: *Proceedings of the International Conference on Database Theory*, 1997.

[Alm98] Alm, L.: Untersuchung zur Kopplung von Datenbanken und IR-Techniken. Diplomarbeit, Universität Rostock, 1998.

[AQMW97] Abiteboul, S.; Quas, D.; McHugh, J.; Widom, J.: The lorel query language for semistructured data. In: *International Journal on Digital Libraries*, S. 68–88, April 1997.

[BDFS97] Buneman, P.; Davidson, S.; Fernandez, M.; Suziu, D.: Adding structure to unstructured data. In: *Proceedings of the International Conference on Database Theory*, 1997.

[BDH96] Buneman, P.; Davidson, S.; Hillebrand, G.: A query language and optimization techniques for unstructired data. In: *Proc. of the ACM SIGMOD International Conference on Management of Data*, 1996.

[BDS95] Buneman, P.; Davidson, S.; Suciu, D.: Programming constructs for unstructured data. In: *Proc. Workshop on Database Programming Languages*, 1995.

[BPSM98] Bray, T.; Paoli, J.; Sperberg-McQueen, C.: Extensible Markup Language (XML) 1.0, Februar 1998. `http://www.w3.org/TR/1998/REC-xml-19980210`.

[Bra98] Bradley, N.: *The XML companion.* Addison Wesley, Harlow, England, 1. Auflage, 1998.

[Cha98] Chamberlin, D.: *A complete guide to DB2 Universal Database.* Morgan Kaufmann Publishers, San Francisco, 1. Auflage, 1998.

[Clu97] Cluet, S.: Modeling and querying semi-structured data. *Lecture Notes in Computer Science*, Band 1299, S. 192–213, 1997.

[Dao98] Dao, T.: An indexing model for structured documents to support queries on content, structure and attributes. In: *Proceedings of the IEEE ADL*, S. 88–97, 1998.

[FLM98] Florescu, D.; Levy, A.; Mendelzon, A.: Database techniques for the world wide web: A survey. *SIGMOD Record*, Band 27, S. 59–74, September 1998.

[Fuh97] Fuhr, N.: Information Retrieval, Skriptum zur Vorlesung, 1997. `http://ls6-www.informatik.uni-dortmund.de/ir/doc/courses/ir/irskall.ps.gz`.

[Gol90] Goldfarb, C. F.: *The SGML Handbook*. Clarendon Press, Oxford, 1990.

[Heu98] Heuer, A.: Global-Info Rahmenantrag auf eine Sonderfördermaßnahme im Schwerpunkt 4, 1998. `http://wwwdb.informatik.uni-rostock.de/global-info/bluerose/index.html`.

[IBM97] IBM Corporation: *DB2 Universal Database TextExtender Administration and Programming*, September 1997.

[Inf97] Informix Software Inc.: *Excalibur Text Search DataBlade Module*, Juli 1997.

[Kop96] Kopka, H.: *LATEX Einführung*. Addison Wesley, Bonn, 2. Auflage, 1996.

[Pet98] Petkovic, D.: *INFORMIX Universal Server*. Addison-Wesley, Bonn, 1998.

[Rah98] Rahm, E.: Evaluation of object-relational database systems for full-text retrieval, 1998. Report Nr. 6.

[Sal71] Salton, G.: *The SMART Retrieval System - Experiments in automatic document processing*. Prentice Hall, Englewood Cliff, New Jersey, 1971.

[Sch98] Schmiede, R.: Global-Info Ein neuer Fokus für Digital Library Entwicklungen, 1998. `http://www.global-info.org/doc/schmiede-9801.html`.

[Se98] Staab, S.; et.al.: A system for facilitating and enhancing web search, 1998. `http://wwwdb.informatik.uni-rostock.de/~duest/ueberblick.dvi.ps`.

[Sto96] Stonebraker, M.: *Object-Relational DBMSs The next great wave*. Morgan Kaufmann Publishers, San Francisco, 1996.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang A

Beispiel-DTD

A.1 Buch DTD

```
<!ELEMENT DOCTYPE (Buch) >

<!ENTITY % block ‘‘Liste | Tabelle | Graphic | Para‘‘>

<!ELEMENT Buch (Meta,Body,Ende?)>
<!ATTLIST Buch Sprache xml:lang NMTOKEN ‘‘de‘‘>

<!ELEMENT Meta (Titel, (Autor | Editor), Verlag, Jahr, ISBN,
                Edition?, Volume?, Note?, Copy?)  >

<!ELEMENT Body ( (Vorwort, (Teil | Kapitel+) ) |
                 (Teil | Kapitel+) )>

<!ELEMENT Ende (Anhang*, Literatur?)>

<!- Elemente des Tags ‘‘Meta‘‘->

<!ELEMENT Autor (Vorname, Middlename?, Nachname)+ >
<!ELEMENT Editor (Vorname, Middlename?, Nachname)+ >
<!ELEMENT Edition     (#PCDATA)>
<!ELEMENT Verlag      (verlagsname, adresse?)>
<!ELEMENT Verlagsname (#PCDATA)>
<!ELEMENT Adresse     (Strasse, Plz, Ort)>
<!ELEMENT Strasse     (#PCDATA)>
<!ELEMENT Plz         (#PCDATA)>
<!ELEMENT Ort         (#PCDATA)>
```

```
<!ELEMENT ISBN        (#PCDATA)>
<!ELEMENT Jahr        (#PCDATA)>
<!ELEMENT Volume      (#PCDATA)>
<!ELEMENT Note        (#PCDATA)>
<!ELEMENT Copy        (#PCDATA)>
<!ELEMENT Vorname     (#PCDATA)>
<!ELEMENT Middlename  (#PCDATA)>
<!ELEMENT Nachname    (#PCDATA)>

<!- Elemente des Tags ‘‘Body‘‘ ->

<!ELEMENT Vorwort (%Block;)+>
<!ELEMENT Teil ((Titel, (%Block;)*, Kapitel+)+)>
<!ATTLIST Teil ident ID #REQUIRED

<!ELEMENT Kapitel (Titel, (%Block;)*, Abschnitt*)>
<!ATTLIST Kapitel ident ID #REQUIRED>

<!ELEMENT Abschnitt (Titel, (%Block;)*, Unterabschnitt*))>
<!ATTLIST Abschnitt ident ID #REQUIRED>

<!ELEMENT Unterabschnitt (Titel, (%Block;), Unterabschnitt*)>
<!ATTLIST Unterabschnitt ident ID #REQUIRED>

<!- Elemente des Tags ‘‘Ende‘‘ ->

<!ELEMENT Anhang (Titel, (%Block;)* Abschnitt*)>
<!ATTLIST Anhang ident ID #REQUIRED>

<!ELEMENT Literatur (Eintrag+)>
<!ELEMENT Eintrag (Autor+, Titel )>
<!ATTLIST Eintrag ref ID #REQUIRED>

<!- Elemente des Entity ‘‘Block‘‘ ->

<!ELEMENT Para (#PCDATA, Citation*, Reference*)* >
<!ELEMENT Graphic EMPTY>
<!ATTLIST Graphic id ID #IMPLIED
          ident ENTITY #REQUIRED>

<!ELEMENT Liste (Item+)>
<!ELEMENT Item (#PCDATA, Citation*, Reference*)*>
<!ATTLIST Item listtype (number | random) random»
```

```
<!ELEMENT Tabelle (Ueberschrift?, Zeile*, Unterschrift?)
<!ELEMENT Ueberschrift (#PCDATA)>
<!ELEMENT Zeile (Spalte+)>
<!ATTLIST Zeile Nummer ID #REQUIRED>

<!ELEMENT spalte (#PCDATA, Citation*, Reference*)*>
<!ELEMENT Unterschrift (#PCDATA)>

<!- Elemente verschiedener Tags ->

<!ELEMENT Titel (Obertitel, Untertitel?)>
<!ELEMENT Obertitel (#PCDATA) >
<!ELEMENT Untertitel (#PCDATA) >
<!ELEMENT Citation (#PCDATA)>
<!ELEMENT Reference (#PCDATA)>
```

A.2 Diplomarbeit DTD

```
<!ELEMENT DOCTYPE (Diplomarbeit) >

<!ENTITY % block ‘‘Liste | Tabelle | Graphic | Para‘‘>

<!ELEMENT Diplomarbeit (Meta,Body,Ende?)>
<!ATTLIST Diplomarbeit Sprache xml:lang NMTOKEN ‘‘de‘‘>

<!ELEMENT Meta (Titel, Autor, Jahr, Schule, Fachbereich, Betreuer,
                Bearbeitungszeit?, Abstract?, Keywords?,
                Klassifikation?)>

<!ELEMENT Body ( (Vorwort, (Teil | Kapitel+) ) |
                 (Teil | Kapitel+) )>

<!ELEMENT Ende (Anhang*, Literatur?)>

<!- Elemente des Tags ‘‘Meta‘‘->

<!ELEMENT Autor (Vorname, Middlename?, Nachname)>
<!ELEMENT Jahr             (#PCDATA)>
<!ELEMENT Schule           (Schulname, Adresse?)>
```

```
<!ELEMENT Schulname        (#PCDATA)>
<!ELEMENT Adresse          (Strasse, Plz, Ort)>
<!ELEMENT Strasse          (#PCDATA)>
<!ELEMENT Plz              (#PCDATA)>
<!ELEMENT Ort              (#PCDATA)>
<!ElEMENT Fachbereich      (#PCDATA)>
<!ELEMENT Betreuer (PersTitel?, Vorname, Middlename?, Nachname)+>
<!Element PersTitel        (#PCDATA)>
<!ELEMENT Bearbeitungszeit (Vom, Bis)>
<!ELEMENT Vom              (#PCDATA) >
<!ELEMENT Bis              (#PCDATA) >
<!ELEMENT Abstract         (#PCDATA)>
<!ELEMENT Keywords         (Begriff+)>
<!ELEMENT Begriff          (#PCDATA)>
<!ELEMENT Klassifikation   (#PCDATA)>
<!ELEMENT Vorname          (#PCDATA)>
<!ELEMENT Middlename       (#PCDATA)>
<!ELEMENT Nachname         (#PCDATA)>

<!- Elemente des Tags ‘‘Body‘‘ ->

<!ELEMENT Vorwort (%Block;)+>
<!ELEMENT Teil ((Titel, (%Block;)*, Kapitel+)+)>
<!ATTLIST Teil ident ID #REQUIRED

<!ELEMENT Kapitel (Titel, (%Block;)*, Abschnitt*)>
<!ATTLIST Kapitel ident ID #REQUIRED>

<!ELEMENT Abschnitt (Titel, (%Block;)*, Unterabschnitt*))>
<!ATTLIST Abschnitt ident ID #REQUIRED>

<!ELEMENT Unterabschnitt (Titel, (%Block;), Unterabschnitt*)>
<!ATTLIST Unterabschnitt ident ID #REQUIRED>

<!- Elemente des Tags ‘‘Ende‘‘ ->

<!ELEMENT Anhang (Titel, (%Block;)* Abschnitt*)>
<!ATTLIST Anhang ident ID #REQUIRED>

<!ELEMENT Literatur (Eintrag+)>
<!ELEMENT Eintrag (Autor+, Titel )>
<!ATTLIST Eintrag ref ID #REQUIRED>
```

```
<!- Elemente des Entity “Block“ ->

<!ELEMENT Para (#PCDATA, Citation*, Reference*)* >
<!ELEMENT Graphic EMPTY>
<!ATTLIST Graphic id ID #IMPLIED
          ident ENTITY #REQUIRED>

<!ELEMENT Liste (Item+)>
<!ELEMENT Item (#PCDATA, Citation*, Reference*)*>
<!ATTLIST Item listtype (number | random) random»

<!ELEMENT Tabelle (Ueberschrift?, Zeile*, Unterschrift?)
<!ELEMENT Ueberschrift (#PCDATA)>
<!ELEMENT Zeile (Spalte+)>
<!ATTLIST Zeile Nummer ID #REQUIRED>

<!ELEMENT spalte (#PCDATA, Citation*, Reference*)*>
<!ELEMENT Unterschrift (#PCDATA)>

<!- Elemente verschiedener Tags ->

<!ELEMENT Titel (Obertitel, Untertitel?)>
<!ELEMENT Obertitel (#PCDATA) >
<!ELEMENT Untertitel (#PCDATA) >
<!ELEMENT Citation (#PCDATA)>
<!ELEMENT Reference (#PCDATA)>
```